



Guia do Desenvolvedor

AWS App Runner



AWS App Runner: Guia do Desenvolvedor

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

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que AWS App Runner é	1
Para quem é o App Runner?	1
Acessando o App Runner	1
Preços do App Runner	2
Próximas etapas	2
Configuração	3
Inscreva-se para uma Conta da AWS	3
Criar um usuário com acesso administrativo	4
Conceder acesso programático	5
Próximas etapas	7
Conceitos básicos	8
Pré-requisitos	8
Etapa 1: criar um serviço App Runner	10
Etapa 2: alterar seu código de serviço	20
Etapa 3: fazer uma alteração na configuração	21
Etapa 4: visualizar os registros do seu serviço	23
Etapa 5: limpar	26
Próximas etapas	26
Arquitetura e conceitos	28
Conceitos do App Runner	29
Configurações compatíveis com o App Runner	30
Recursos do App Runner	31
Cotas de recursos do App Runner	33
Serviço baseado em imagem	36
Provedores de repositórios de imagens	36
Usando uma imagem armazenada no Amazon ECR em sua conta AWS	37
Usando uma imagem armazenada no Amazon ECR em uma conta diferente AWS	37
Usando uma imagem armazenada no Amazon ECR Public	38
Exemplo de imagem	39
Serviço baseado em código	40
Provedores de repositórios de código-fonte	41
Implantação a partir do seu provedor de repositório de código-fonte	41
Diretório de origem	42
Plataformas gerenciadas do App Runner	43

Versões de tempo de execução gerenciadas e a compilação do App Runner	43
Saiba mais sobre as compilações e a migração do App Runner	45
Plataforma Python	49
Configuração de tempo de execução do Python	50
Explicações para versões específicas de tempo de execução	51
Exemplos de tempo de execução em Python	52
Informações de lançamento	56
Plataforma Node.js	58
Configuração de tempo de execução do Node.js	59
Explicações para versões específicas de tempo de execução	62
Exemplos de tempo de execução do Node.js	62
Informações de lançamento	67
Plataforma Java	69
Configuração do Java Runtime	70
Exemplos de tempo de execução em Java	71
Informações de lançamento	75
Plataforma.NET	77
Configuração do runtime do.NET	78
Exemplos de runtime do.NET	79
Informações de lançamento	81
Plataforma PHP	83
Configuração de tempo de execução do PHP	84
Compatibilidade	85
Exemplos de tempo de execução em PHP	86
Informações de lançamento	95
Plataforma Ruby	96
Configuração de tempo de execução do Ruby	97
Exemplos de tempo de execução do Ruby	98
Informações de lançamento	100
Plataforma Go	101
Configuração de tempo de execução do Go	102
Exemplos de tempo de execução do Go	102
Informações de lançamento	105
Desenvolvendo para o App Runner	106
Informações de tempo de execução	106
Diretrizes de desenvolvimento de código	108

Console do App Runner	110
Layout geral do console	110
A página de serviços	111
A página do painel de controle do serviço	111
A página Contas conectadas	113
A página de configurações de Auto Scaling	113
Gerenciando seu serviço	115
Criação	115
Pré-requisitos	116
Criar um serviço	116
Serviço com falha na reconstrução	132
Reconstruindo um serviço do App Runner com falha usando o console do App Runner	132
Reconstrução do serviço App Runner com falha usando a API App Runner ou AWS CLI	133
Implantação	134
Métodos de implantação	134
Implantação manual	136
Configuração	138
Configure seu serviço usando a API App Runner ou AWS CLI	139
Configure seu serviço usando o console do App Runner	140
Configure seu serviço usando um arquivo de configuração do App Runner	142
Configuração de observabilidade	142
Recursos de configuração	144
Configuração de verificação de integridade	146
Conexões	148
Gerenciar conexões	149
Ajuste de escala automático	150
Gerencie o escalonamento automático para um serviço	153
Gerencie recursos de configurações de auto scaling	154
Nomes de domínios personalizados	162
Associe (vincule) um domínio personalizado ao seu serviço	163
Desassociar (desvincular) um domínio personalizado	166
Gerenciar domínios personalizados	167
Configurar um registro de alias do Amazon Route 53	175
Pausando/ retomando	177
Comparação entre pausar e excluir	178
Quando seu serviço está pausado	178

Pausar e retomar seu serviço	179
Exclusão	181
Comparação entre pausar e excluir	181
O que o App Runner exclui?	182
Exclua seu serviço	182
Variáveis de ambiente de referência	184
Referenciando dados confidenciais como variáveis de ambiente	184
Considerações	186
Permissões	186
Gerenciar variáveis de ambiente	188
Console do App Runner	188
API App Runner ou AWS CLI	190
Redes	196
Terminologia	196
Termos gerais	196
Termo específico para configurar o tráfego de saída	197
Termos específicos para configurar o tráfego de entrada	197
Tráfego de entrada	198
Cabeçalhos	199
Habilitar endpoint privado	199
Ativar IPv6 para os endpoints públicos do App Runner	212
Tráfego de saída	217
Conector VPC	218
Sub-rede	218
Grupo de segurança	219
Gerenciar o acesso à VPC	220
Observabilidade	226
Atividade	226
Rastreie a atividade do serviço App Runner	226
Registros (CloudWatch Registros)	228
Grupos e streams de registros do App Runner	228
Visualizando os registros do App Runner no console	230
Métricas (CloudWatch)	232
Métricas do App Runner	232
Visualizando métricas do App Runner no console	234
Tratamento de eventos (EventBridge)	236

Criação de uma EventBridge regra para agir em eventos do App Runner	237
Exemplos de eventos do App Runner	237
Exemplos de padrões de eventos do App Runner	239
Referência do evento App Runner	240
Ações de API (CloudTrail)	242
Informações sobre o App Runner em CloudTrail	242
Entendendo as entradas do arquivo de log do App Runner	243
Rastreamento (X-Ray)	246
Instrumente seu aplicativo para rastreamento	247
Adicione permissões X-Ray à sua função de instância de serviço App Runner	250
Ative o rastreamento X-Ray para seu serviço App Runner	251
Visualize os dados de rastreamento do X-Ray para seu serviço App Runner	251
AWS WAF ACL da web	252
Fluxo de entrada de solicitações da web	252
Associando o WAF web ACLs ao seu serviço App Runner	253
Considerações	254
Permissões	255
Gerenciar a web ACLs	256
Console do App Runner	256
AWS CLI	260
Testando e registrando AWS WAF na web ACLs	265
Arquivo de configuração do App Runner	267
Exemplos	268
Exemplos de arquivos de configuração	268
Referência	271
Visão geral da estrutura	271
Seção superior	272
Seção de construção	273
Seção de execução	275
API do App Runner	279
Usando o AWS CLI para trabalhar com o App Runner	279
Usando AWS CloudShell	279
Obtendo permissões do IAM para AWS CloudShell	280
Interagindo com o App Runner usando AWS CloudShell	281
Verificando seu serviço App Runner usando AWS CloudShell	284
Solução de problemas	285

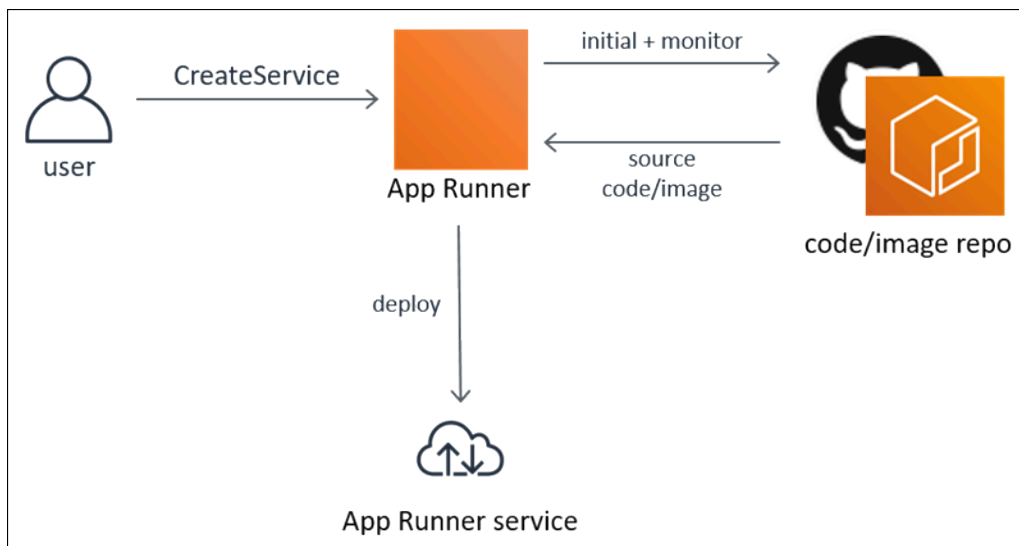
Falha na criação do serviço	285
Nomes de domínios personalizados	286
Obtendo o erro Create Fail para domínio personalizado	287
Erro pendente de validação do certificado DNS para domínio personalizado	288
Comandos básicos de solução de problemas	288
Renovação de certificado de domínio personalizado	289
Erro de roteamento da solicitação	290
Erro 404 Não encontrado ao enviar tráfego HTTP/HTTPS para endpoints do serviço App Runner	290
Falha na conexão com o Amazon RDS ou com o serviço downstream	291
Quando não há endereços IP suficientes para lançamento ou escalabilidade	294
Como atualizar seus serviços para ter mais serviços disponíveis IPs	294
Cálculo IPs necessário para seus serviços	295
Criar nova (s) sub-rede (s)	295
Anexando blocos CIDR secundários à sua VPC	296
Verificação	297
Armadilhas comuns	297
Recursos adicionais	298
Glossário	298
Segurança	299
Proteção de dados	300
Criptografia de dados	301
Privacidade entre redes	302
Gerenciamento de identidade e acesso	302
Público	303
Autenticar com identidades	304
Gerenciar o acesso usando políticas	307
App Runner e IAM	310
Exemplos de políticas baseadas em identidade	318
Uso de perfis vinculados ao serviço	323
AWS políticas gerenciadas	331
Solução de problemas	333
Registro em log e monitoramento	334
Validação de conformidade	335
Resiliência	337
Segurança da infraestrutura	337

Endpoints da VPC	338
Configurando um VPC endpoint para o App Runner	338
Considerações sobre privacidade da rede VPC	339
Usar políticas de endpoint para controlar o acesso com VPC endpoints	339
Integração com o endpoint de interface	340
Modelo de responsabilidade compartilhada	340
Imagens de contêineres de patches	340
Práticas recomendadas de segurança	340
Práticas recomendadas de segurança preventiva	340
Práticas recomendadas de segurança de detecção	341
AWS Glossário	342
.....	cccxliii

O que AWS App Runner é

AWS App Runner é um AWS serviço que fornece uma maneira rápida, simples e econômica de implantar a partir do código-fonte ou de uma imagem de contêiner diretamente em um aplicativo web escalável e seguro na AWS nuvem. Você não precisa aprender novas tecnologias, decidir qual serviço de computação usar ou saber como provisionar e configurar AWS recursos.

O App Runner se conecta diretamente ao seu repositório de código ou imagem. Ele fornece um pipeline automático de integração e entrega com operações totalmente gerenciadas, alto desempenho, escalabilidade e segurança.



Para quem é o App Runner?

Se você é desenvolvedor, pode usar o App Runner para simplificar o processo de implantação de uma nova versão do seu repositório de código ou imagem.

Para equipes de operações, o App Runner permite implantações automáticas sempre que uma confirmação é enviada para o repositório de código ou uma nova versão da imagem do contêiner é enviada para o repositório de imagens.

Acessando o App Runner

Você pode definir e configurar suas implantações de serviço App Runner usando qualquer uma das seguintes interfaces:

- Console do App Runner — fornece uma interface web para gerenciar seus serviços do App Runner.
- API do App Runner — fornece uma RESTful API para realizar ações do App Runner. Para obter mais informações, consulte [Referência de API do AWS App Runner](#).
- AWS Interface de linha de comando (AWS CLI) — Fornece comandos para um amplo conjunto de AWS serviços, incluindo Amazon VPC, e é compatível com Windows, macOS e Linux. Para obter mais informações, consulte [AWS Command Line Interface](#).
- AWS SDKs— fornece informações específicas para o idioma APIs e cuida de muitos detalhes da conexão, como calcular assinaturas, lidar com novas tentativas de solicitação e tratamento de erros. Para obter mais informações, consulte [AWS SDKs](#).

Preços do App Runner

O App Runner fornece uma maneira econômica de executar seu aplicativo. Você paga apenas pelos recursos que seu serviço App Runner consome. Seu serviço é reduzido para menos instâncias de computação quando o tráfego de solicitações é menor. Você tem controle sobre as configurações de escalabilidade: o menor e o maior número de instâncias provisionadas e a maior carga que uma instância processa.

Para obter mais informações sobre o escalonamento automático do App Runner, consulte [the section called “Ajuste de escala automático”](#)

Para obter informações sobre preços, consulte [AWS App Runner preços](#).

Próximas etapas

Saiba como começar a usar o App Runner nos seguintes tópicos:

- [Configuração](#)— Conclua as etapas de pré-requisito para usar o App Runner.
- [Conceitos básicos](#)— Implante seu primeiro aplicativo no App Runner.

Configuração para o App Runner

Se você for um AWS cliente novo, preencha os pré-requisitos de configuração listados nesta página antes de começar a usar. AWS App Runner

Para esses procedimentos de configuração, você usa o serviço AWS Identity and Access Management (IAM). Para obter mais informações sobre o IAM, consulte os seguintes materiais de referência:

- [AWS Identity and Access Management \(IAM\)](#)
- [Guia do usuário do IAM](#)

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra a <https://portal.aws.amazon.com/billing/inscrição>.
2. Siga as instruções online.

Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar [tarefas que exigem acesso de usuário-raiz](#).

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, você pode visualizar a atividade atual da sua conta e gerenciar sua conta acessando <https://aws.amazon.com/e> escolhendo Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login [AWS Management Console](#) como proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira a senha.

Para obter ajuda ao fazer login usando o usuário-raiz, consulte [Fazer login como usuário-raiz](#) no Guia do usuário do Início de Sessão da AWS .

2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

Para obter instruções, consulte [Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz \(console\) no Guia](#) do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

Para obter instruções, consulte [Habilitar o AWS IAM Identity Center](#) no Guia do usuário do AWS IAM Identity Center .

2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte [Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM](#) no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte [Como fazer login no portal de AWS acesso](#) no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

Para obter instruções, consulte [Criar um conjunto de permissões](#) no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

Para obter instruções, consulte [Adicionar grupos](#) no Guia do usuário do AWS IAM Identity Center .

Conceder acesso programático

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do AWS Management Console. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso AWS IAM Identity Center no Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referênci

Qual usuário precisa de acesso programático?	Para	Por
		a de ferramentas AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para o AWS CLI AWS SDKs, ou. AWS APIs	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para o AWS CLI, AWS SDKs, ou. AWS APIs	Siga as instruções da interface que deseja utilizar. • Para isso AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para ferramentas AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de ferramentas AWS SDKs e ferramentas. • Para isso AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Próximas etapas

Você concluiu as etapas de pré-requisito. Para implantar seu primeiro aplicativo no App Runner, consulte [Conceitos básicos](#).

Introdução ao App Runner

AWS App Runner é um AWS serviço que fornece uma maneira rápida, simples e econômica de transformar uma imagem de contêiner ou código-fonte existente diretamente em um serviço Web em execução no Nuvem AWS.

Este tutorial aborda como você pode usar AWS App Runner para implantar seu aplicativo em um serviço do App Runner. Ele explica a configuração do código-fonte e da implantação, da compilação do serviço e do tempo de execução do serviço. Também mostra como implantar uma versão de código, fazer uma alteração na configuração e visualizar registros. Por fim, o tutorial mostra como limpar os recursos que você criou enquanto segue os procedimentos do tutorial.

Tópicos

- [Pré-requisitos](#)
- [Etapa 1: criar um serviço App Runner](#)
- [Etapa 2: alterar seu código de serviço](#)
- [Etapa 3: fazer uma alteração na configuração](#)
- [Etapa 4: visualizar os registros do seu serviço](#)
- [Etapa 5: limpar](#)
- [Próximas etapas](#)

Pré-requisitos

Antes de iniciar o tutorial, certifique-se de realizar as seguintes ações:

1. Conclua as etapas de configuração em [Configuração](#).
2. Decida se você gostaria de trabalhar com um repositório ou um GitHub repositório Bitbucket.
 - Para trabalhar com um Bitbucket, primeiro crie uma conta do [Bitbucket](#), se você ainda não tiver uma. Se você é novo no Bitbucket, consulte [Introdução ao Bitbucket](#) na documentação do Bitbucket Cloud.
 - Para trabalhar com GitHub ela, crie uma [GitHub](#) conta, se você ainda não tiver uma. Se você é GitHub novato, consulte [Introdução ao GitHub](#) [GitHub Docs](#).

 Note

Você pode criar conexões com vários provedores de repositório a partir da sua conta. Portanto, se você quiser acompanhar a implantação a partir de um repositório Bitbucket GitHub e de um repositório do Bitbucket, repita esse procedimento. Na próxima vez, crie um novo serviço App Runner e crie uma nova conexão de conta para o outro provedor de repositório.

3. Crie um repositório na sua conta do provedor de repositórios. Este tutorial usa o nome `python-hello` do repositório. Crie arquivos no diretório raiz do repositório, com os nomes e o conteúdo especificados nos exemplos a seguir.

Arquivos para o repositório de **python-hello** exemplo

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)
```

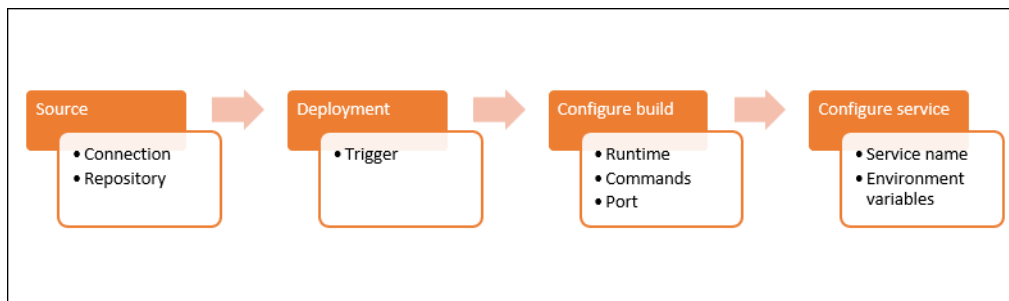
```
server.serve_forever()
```

Etapa 1: criar um serviço App Runner

Nesta etapa, você cria um serviço App Runner com base no exemplo de repositório de código-fonte que você criou no GitHub ou no Bitbucket como parte do. [the section called “Pré-requisitos”](#) O exemplo contém um site simples em Python. Estas são as principais etapas que você segue para criar um serviço:

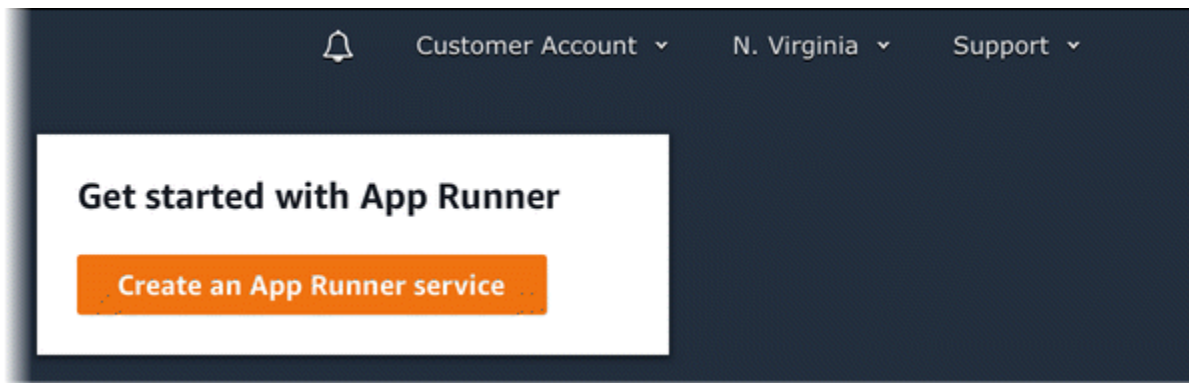
1. Configure seu código-fonte.
2. Configure a implantação da fonte.
3. Configure a criação do aplicativo.
4. Configure seu serviço.
5. Revise e confirme.

O diagrama a seguir descreve as etapas para criar um serviço App Runner:

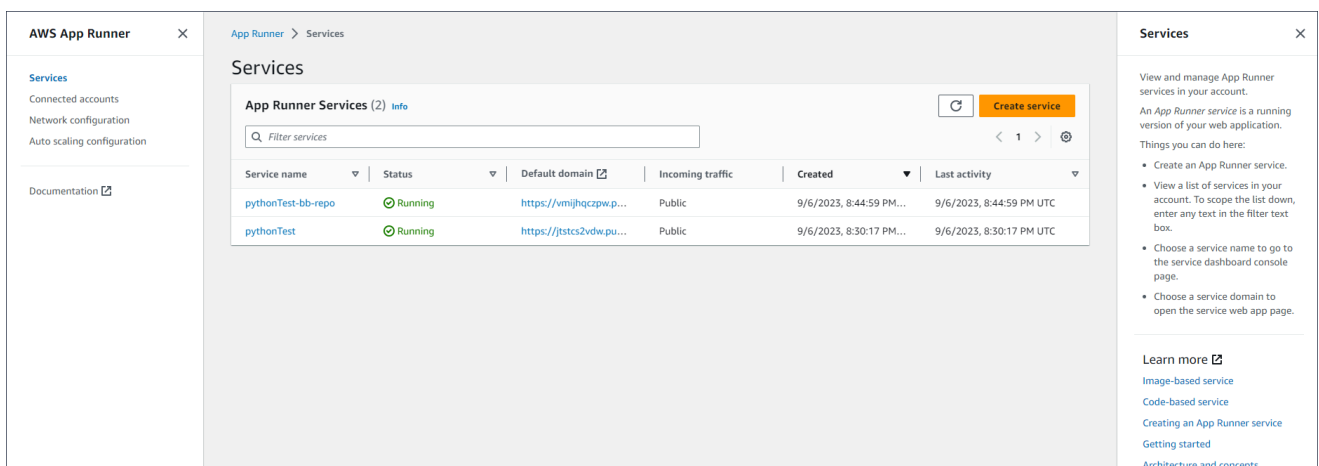


Para criar um serviço App Runner com base em um repositório de código-fonte

1. Configure seu código-fonte.
 - a. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
 - b. Se Conta da AWS ainda não tiver nenhum serviço do App Runner, a página inicial do console será exibida. Escolha Criar um serviço App Runner.



Se tiver serviços existentes, a página Serviços com uma lista de seus serviços será exibida. Conta da AWS Escolha Create service.



- Na página Fonte e implantação, na seção Fonte, em Tipo de repositório, escolha Repositório de código-fonte.
- Selecione um tipo de provedor. Escolha um GitHub ou o Bitbucket.
- Em seguida, escolha Adicionar novo. Se solicitado, forneça suas credenciais GitHub ou do Bitbucket.
- Escolha o próximo conjunto de etapas com base no tipo de provedor selecionado anteriormente.

Note

As etapas a seguir para instalar o conector AWS GitHub em sua conta são etapas únicas. GitHub Você pode reutilizar a conexão para criar vários serviços do App Runner com base nos repositórios dessa conta. Quando você tiver uma conexão existente, escolha-a e vá para a seleção do repositório.

O mesmo se aplica ao conector AWS para sua conta do Bitbucket. Se você estiver usando ambos GitHub e o Bitbucket como repositórios de código-fonte para seus serviços do App Runner, precisará instalar um AWS Connector para cada provedor. Em seguida, você pode reutilizar cada conector para criar mais serviços do App Runner.

- Para GitHub, siga estas etapas.
 - i. Na próxima tela, insira um nome de conexão.
 - ii. Se esta é sua primeira vez usando o GitHub App Runner, selecione Instalar outro.
 - iii. Na caixa de GitHub diálogo AWS Conector para, se solicitado, escolha o nome GitHub da sua conta.
 - iv. Se solicitado a autorizar o AWS Conector para GitHub, escolha Autorizar conexões da AWS.
 - v. Na caixa de GitHub diálogo Instalar AWS conector para, escolha Instalar.

O nome da sua conta aparece como a GitHub conta/organização selecionada. Agora você pode escolher um repositório na sua conta.
 - vi. Em Repositório, escolha o repositório de exemplo que você criou, `python-hello`. Para Branch, escolha o nome padrão do branch do seu repositório (por exemplo, `main`).
 - vii. Deixe o diretório de origem com o valor padrão. O diretório usa como padrão a raiz do repositório. Você armazenou seu código-fonte no diretório raiz do repositório nas etapas anteriores dos Pré-requisitos.
- Para o Bitbucket, siga estas etapas.
 - i. Na próxima tela, insira um nome de conexão.
 - ii. Se esta é sua primeira vez usando o Bitbucket com o App Runner, selecione Instalar outro.
 - iii. Na caixa de diálogo de AWS CodeStar solicitações de acesso, você pode selecionar seu espaço de trabalho e conceder acesso à integração com AWS CodeStar o Bitbucket. Selecione seu espaço de trabalho e, em seguida, selecione Conceder acesso.

- iv. Em seguida, você será redirecionado para o AWS console. Verifique se o aplicativo Bitbucket está configurado para o espaço de trabalho correto do Bitbucket e selecione Avançar.
 - v. Em Repositório, escolha o repositório de exemplo que você criou, `python-hello`. Para Branch, escolha o nome padrão do branch do seu repositório (por exemplo, `main`).
 - vi. Deixe o diretório de origem com o valor padrão. O diretório usa como padrão a raiz do repositório. Você armazenou seu código-fonte no diretório raiz do repositório nas etapas anteriores dos Pré-requisitos.
2. Configure suas implantações: na seção Configurações de implantação, escolha Automático e, em seguida, escolha Avançar.

 Note

Com a implantação automática, cada nova confirmação no diretório de origem do repositório implanta automaticamente uma nova versão do seu serviço.

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.

3. Configure a criação do aplicativo.

- a. Na página Configurar compilação, em Arquivo de configuração, escolha Definir todas as configurações aqui.
- b. Forneça as seguintes configurações de compilação:
 - Runtime — Escolha Python 3.
 - Comando de compilação — Enter **`pip install -r requirements.txt`**.
 - Comando Iniciar — Enter **`python server.py`**.
 - Porta — Entrar **8080**.
- c. Escolha Próximo.

Note

O tempo de execução do Python 3 cria uma imagem do Docker usando uma imagem básica do Python 3 e seu código Python de exemplo. Em seguida, ele inicia um serviço que executa uma instância de contêiner dessa imagem.

Configure build Info

Build settings

Configuration file

☒ **Configure all settings here**
Specify all settings for your service here in the App Runner console.

☐ **Use a configuration file**
Let App Runner read your configuration from the `apprunner.yaml` file in your source repository.

Runtime
Choose an App Runner runtime for your service.

Python 3 ▼

Build command
This command runs in the root directory of your repository when a new code version is deployed. Use it to install dependencies or compile your code.

`pip install -r requirements.txt`

Start command
This command runs in the root directory of your service to start the service processes. Use it to start a webserver for your service. The command can access environment variables that App Runner and you defined.

`python server.py`

Port
Your service uses this IP port.

8080 ▼

Cancel Previous **Next**

4. Configure seu serviço.

- Na página Configurar serviço, na seção Configurações do serviço, insira um nome de serviço.
- Em Variáveis de ambiente, selecione Adicionar variável de ambiente. Forneça os seguintes valores para a variável de ambiente.
 - Fonte — Escolha texto sem formatação
 - Nome da variável de ambiente — **NAME**
 - Valor da variável de ambiente — qualquer nome (por exemplo, seu primeiro nome).

 Note

O aplicativo de exemplo lê o nome que você definiu nessa variável de ambiente e exibe o nome em sua página da Web.

- c. Escolha Próximo.

Configure service [Info](#)

Service settings

Service name

Virtual CPU & memory

Environment variables — *optional* [Info](#)

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

No environment variables have been configured.

Add environment variable

You can add up to 50 more items.

► IAM policy templates

► Auto scaling [Info](#)

Configure automatic scaling behavior.

► Health check [Info](#)

Configure load balancer health checks.

► Security [Info](#)

Specify an Instance role and an AWS KMS encryption key

► Networking [Info](#)

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

► Observability

Configure observability tooling.

▼ Tags [Info](#)

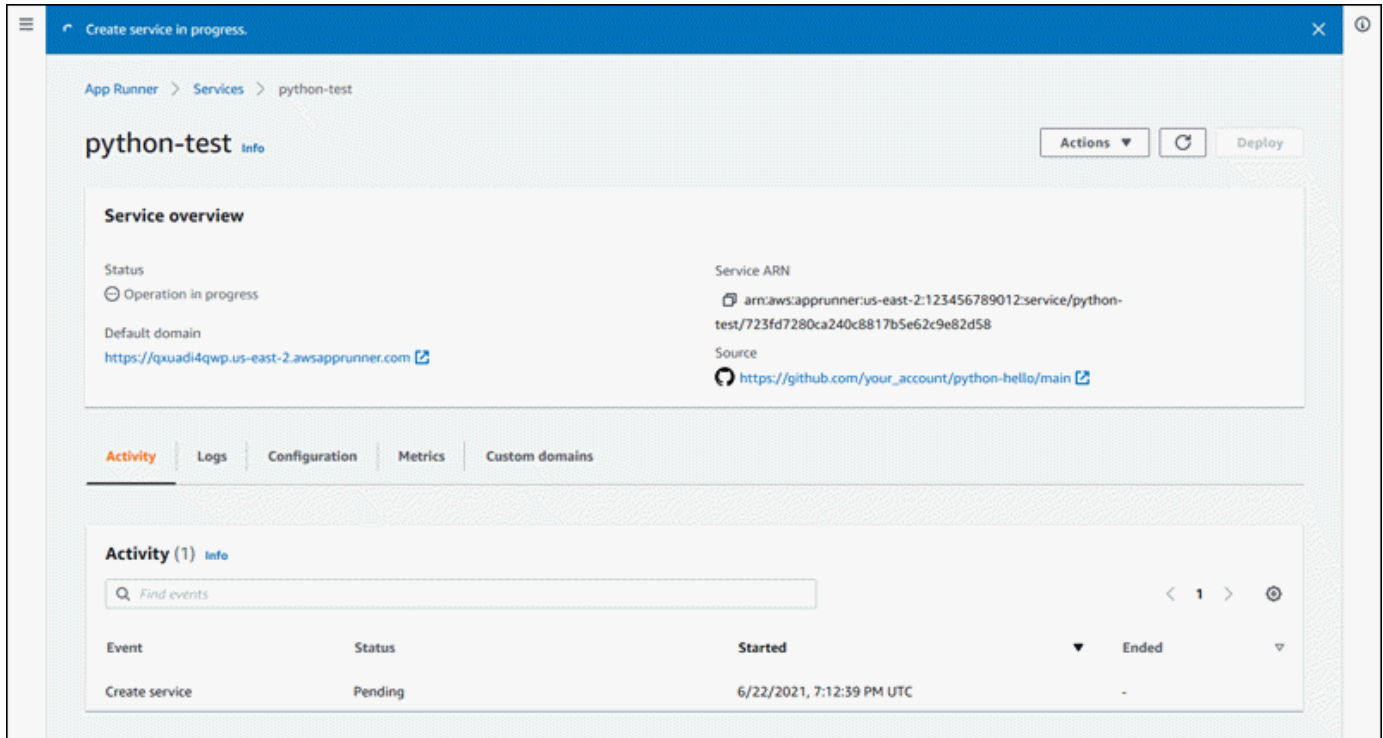
Use tags to search and filter your resources, track your AWS costs, and control access permissions.

Tags — *optional*

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

5. Na página Revisar e criar, verifique todos os detalhes que você inseriu e escolha Criar e implantar.

Se o serviço for criado com sucesso, o console mostrará o painel do serviço, com uma visão geral do novo serviço.



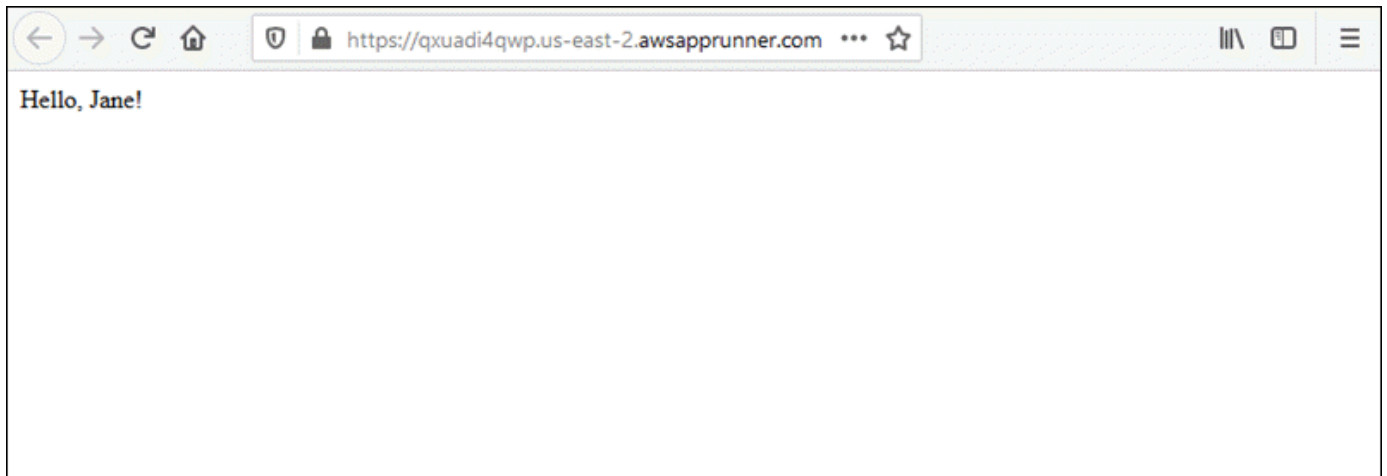
6. Verifique se o serviço está em execução.
 - a. Na página do painel do serviço, aguarde até que o status do serviço esteja em execução.
 - b. Escolha o valor do domínio padrão — é o URL do site do seu serviço.

Note

Para aumentar a segurança de seus aplicativos App Runner, o domínio*.awsapprunner.com é registrado na Lista Pública de Sufixos (PSL).

Para maior segurança, recomendamos que você use cookies com um __Host- prefixo se precisar definir cookies confidenciais no nome de domínio padrão para seus aplicativos App Runner. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

Uma página da web exibe: Olá,*your name*!

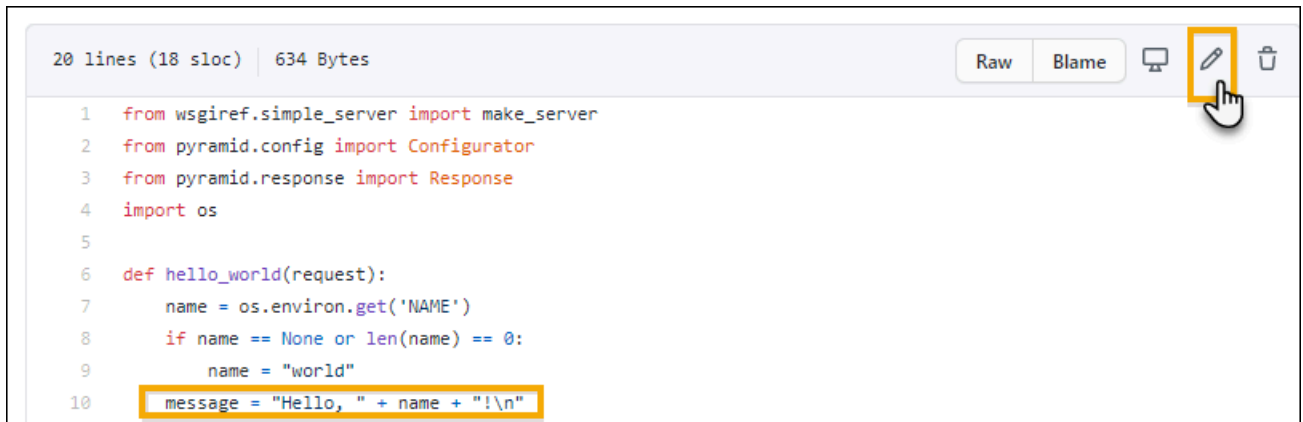


Etapa 2: alterar seu código de serviço

Nesta etapa, você faz uma alteração no seu código no diretório de origem do repositório. O recurso de CI/CD do App Runner cria e implanta automaticamente a alteração em seu serviço.

Para fazer uma alteração no seu código de serviço

1. Navegue até seu repositório de exemplo.
2. Edite o arquivo chamado `server.py`.
3. Na expressão atribuída à variável `message`, altere o texto `Hello` para `Good morning`.
4. Salve e confirme suas alterações no repositório.
5. As etapas a seguir ilustram a alteração do código de serviço em um GitHub repositório.
 - a. Navegue até seu GitHub repositório de exemplo.
 - b. Escolha o nome do arquivo `server.py` para navegar até esse arquivo.
 - c. Escolha Editar este arquivo (o ícone de lápis).
 - d. Na expressão atribuída à variável `message`, altere o texto `Hello` para `Good morning`.



```
20 lines (18 sloc) | 634 Bytes
1  from wsgiref.simple_server import make_server
2  from pyramid.config import Configurator
3  from pyramid.response import Response
4  import os
5
6  def hello_world(request):
7      name = os.environ.get('NAME')
8      if name == None or len(name) == 0:
9          name = "world"
10     message = "Hello, " + name + "!\\n"
```

e. Escolha Commit changes (Confirmar alterações).

6. O novo commit começa a ser implantado em seu serviço App Runner. Na página do painel do serviço, o status do serviço muda para Operação em andamento.

Aguarde o término da implantação. Na página do painel do serviço, o status do serviço deve voltar para Em execução.

7. Verifique se a implantação foi bem-sucedida: atualize a guia do navegador na qual a página da web do seu serviço é exibida.

A página agora exibe a mensagem modificada: Bom dia,**your name**!

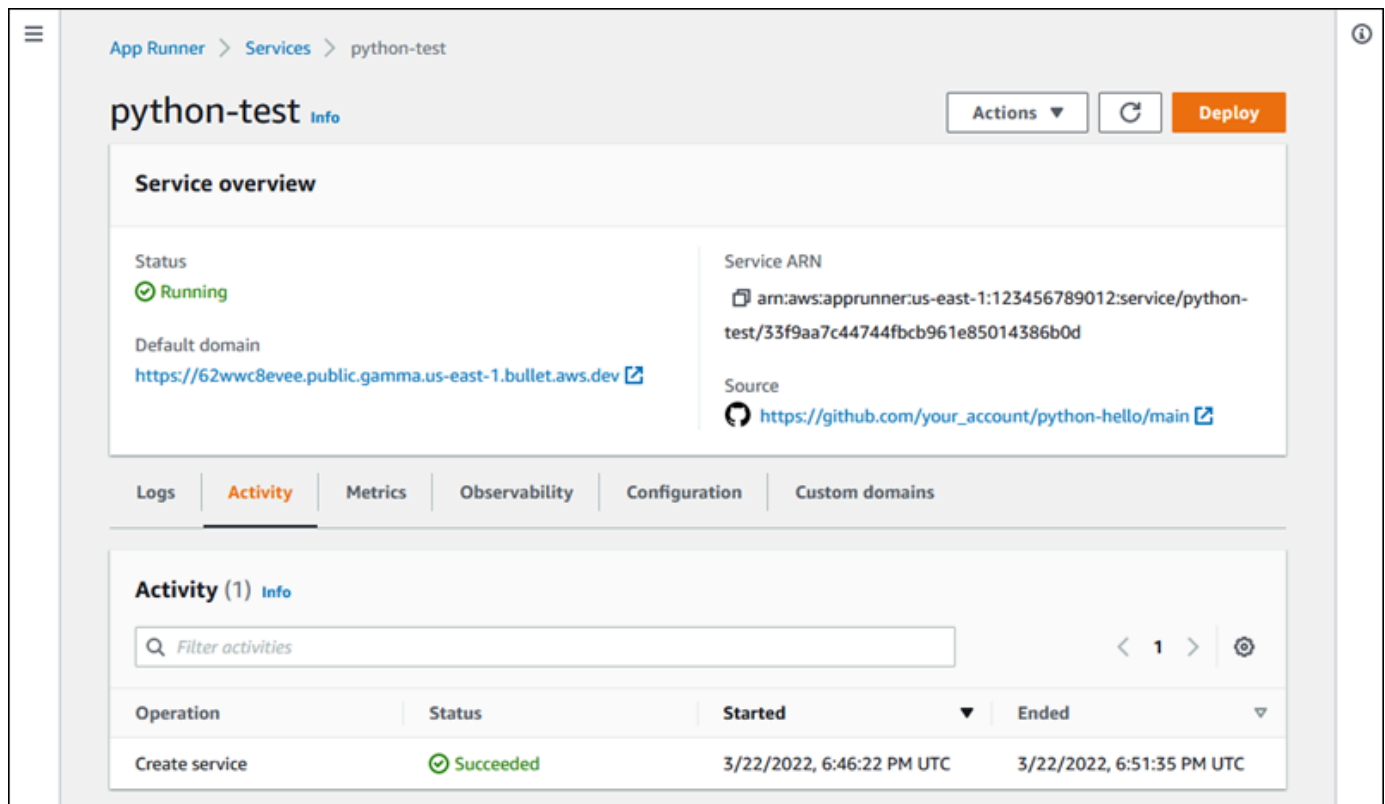
Etapa 3: fazer uma alteração na configuração

Nesta etapa, você faz uma alteração no valor da variável de **NAME** ambiente para demonstrar uma alteração na configuração do serviço.

Para alterar o valor de uma variável de ambiente

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

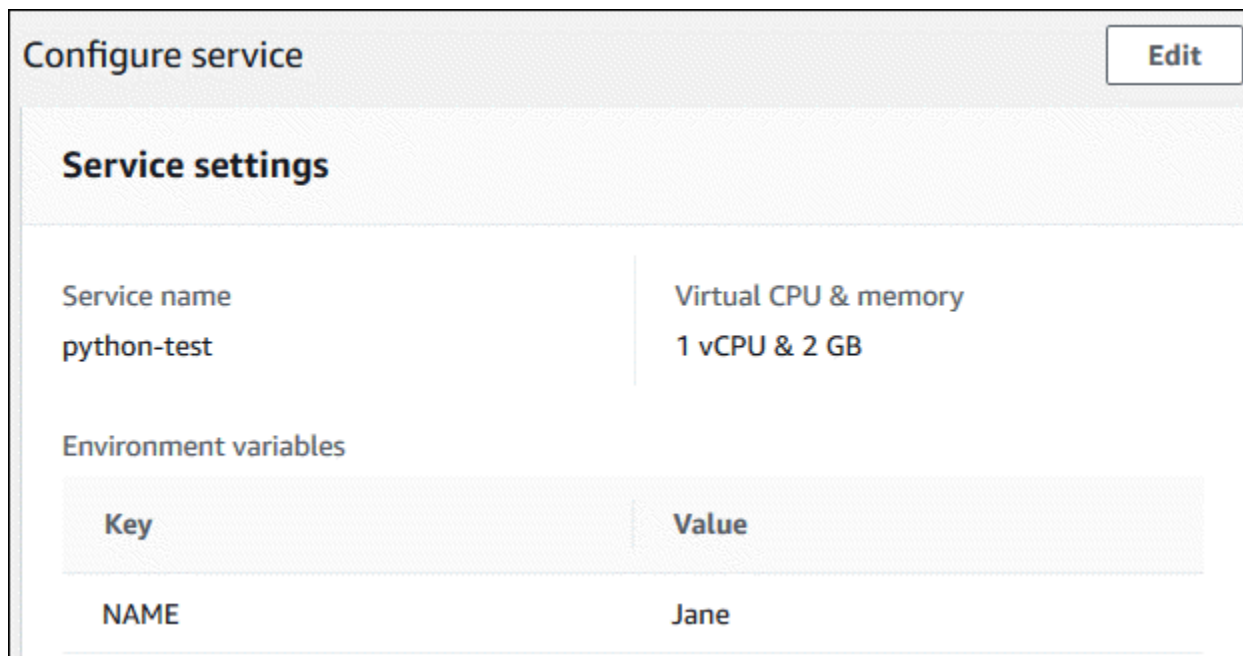
O console exibe o painel do serviço com uma visão geral do serviço.



3. Na página do painel do serviço, escolha a guia Configuração.

O console exibe suas configurações de serviço em várias seções.

4. Na seção Configurar serviço, escolha Editar.



5. Para a variável de ambiente com a chave **NAME**, altere o valor para um nome diferente.

6. Selecione Aplicar alterações.

O App Runner inicia o processo de atualização. Na página do painel do serviço, o status do serviço muda para Operação em andamento.

7. Aguarde até que a atualização termine. Na página do painel do serviço, o status do serviço deve voltar para Em execução.
8. Verifique se a atualização foi bem-sucedida: atualize a guia do navegador onde a página da web do seu serviço é exibida.

A página agora exibe o nome modificado: Bom dia,**new name**!

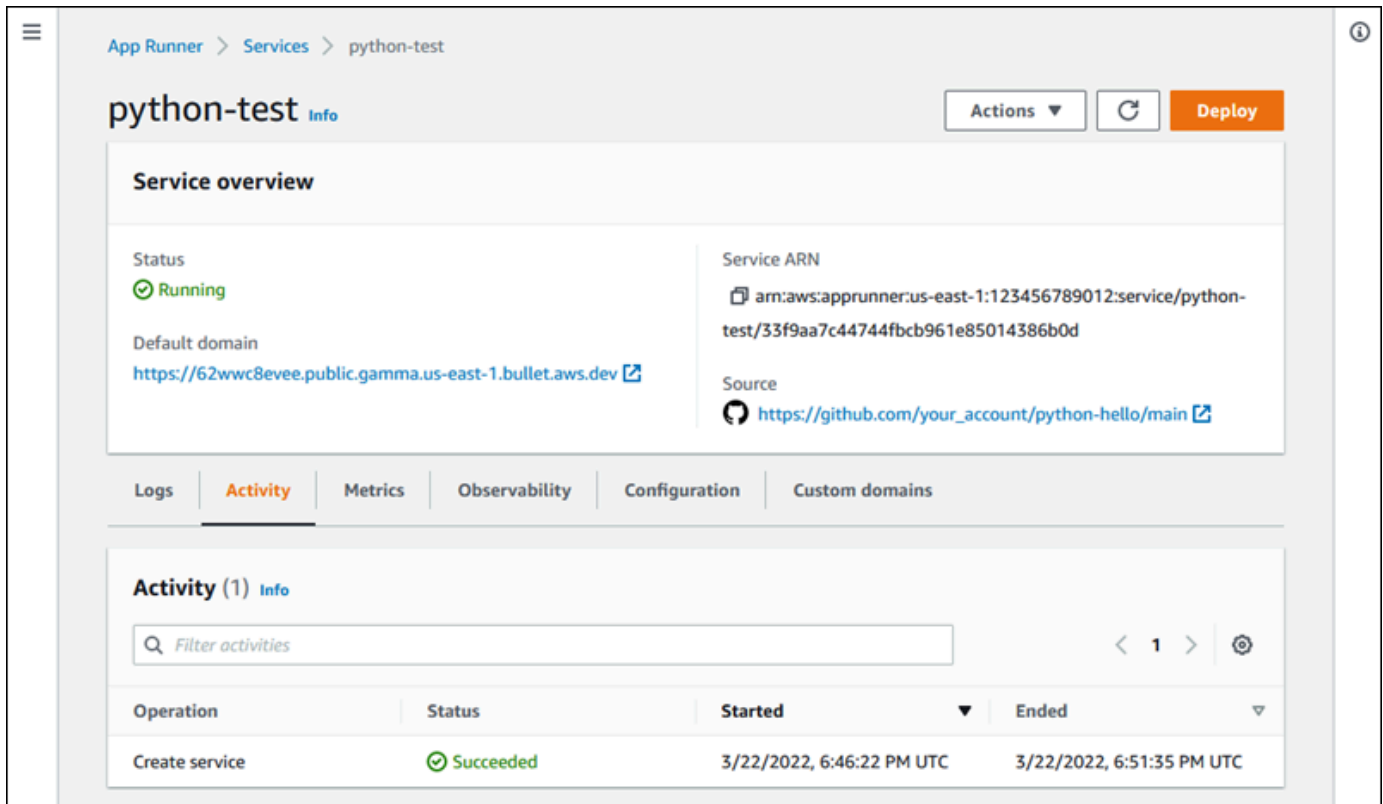
Etapa 4: visualizar os registros do seu serviço

Nesta etapa, você usa o console do App Runner para visualizar os registros do seu serviço App Runner. O App Runner transmite registros para o Amazon CloudWatch Logs (CloudWatch Logs) e os exibe no painel do seu serviço. Para obter informações sobre os registros do App Runner, consulte [the section called “Registros \(CloudWatch Registros\)”](#).

Para ver os registros do seu serviço

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Na página do painel do serviço, escolha a guia Registros.

O console exibe alguns tipos de registros em várias seções:

- Registro de eventos — Atividade no ciclo de vida do seu serviço App Runner. O console exibe os eventos mais recentes.
- Registros de implantação — forneça implantações de repositórios para seu serviço App Runner. O console exibe um fluxo de log separado para cada implantação.
- Registros do aplicativo — A saída do aplicativo web que é implantado no seu serviço App Runner. O console combina a saída de todas as instâncias em execução em um único fluxo de registros.

The screenshot displays the AWS App Runner console interface. At the top, the 'Event log' section shows a list of events with timestamps and descriptions, such as 'Build service started', 'Build service completed', 'my-web-service1 server running', and 'Deploying service'. Below this, the 'Deployment logs (1)' section includes a search bar and a table with columns for Operation, Status, Started, and Ended. The table shows one entry: 'Automatic deployment' with a status of 'In progress' and a start time of '12/21/2020, 2:30:31 PM UTC'. At the bottom, the 'Application logs' section shows a table with columns for Name and Last written, with one entry: 'Application logs' with a last written time of '12/21/2020, 2:30:31 PM UTC'. Navigation buttons like 'View in CloudWatch', 'View full log', and 'Download' are visible throughout the interface.

4. Para encontrar implantações específicas, examine a lista de registros de implantação inserindo um termo de pesquisa. Você pode pesquisar qualquer valor que apareça na tabela.
5. Para visualizar o conteúdo de um registro, escolha Exibir registro completo (registro de eventos) ou o nome do fluxo de registros (registros de implantação e aplicativos).
6. Escolha Baixar para baixar um registro. Para um fluxo de registros de implantação, selecione primeiro um fluxo de registros.
7. Escolha Visualizar em CloudWatch para abrir o CloudWatch console e usar seus recursos completos para explorar seus registros de serviço do App Runner. Para um fluxo de registros de implantação, selecione primeiro um fluxo de registros.

Note

O CloudWatch console é particularmente útil se você quiser visualizar os registros do aplicativo de instâncias específicas em vez do registro combinado do aplicativo.

Etapa 5: limpar

Agora você aprendeu como criar um serviço App Runner, visualizar registros e fazer algumas alterações. Nesta etapa, você exclui o serviço para remover recursos dos quais não precisa mais.

Para excluir seu serviço

1. Na página do painel do serviço, escolha Ações e, em seguida, escolha Excluir serviço.
2. Na caixa de diálogo de confirmação, insira o texto solicitado e escolha Excluir.

Resultado: o console navega até a página Serviços. O serviço que você acabou de excluir mostra o status EXCLUINDO. Pouco tempo depois, ele desaparece da lista.

Considere também excluir as conexões GitHub e o Bitbucket que você criou como parte deste tutorial. Para obter mais informações, consulte [the section called “Conexões”](#).

Próximas etapas

Agora que você implantou seu primeiro serviço App Runner, saiba mais nos tópicos a seguir:

- [Arquitetura e conceitos](#)— A arquitetura, os principais conceitos e AWS recursos relacionados ao App Runner.
- [Serviço baseado em imagem](#) e [Serviço baseado em código](#) — Os dois tipos de fonte de aplicativo que o App Runner pode implantar.
- [Desenvolvendo para o App Runner](#)— Coisas que você deve saber ao desenvolver ou migrar o código do aplicativo para implantação no App Runner.
- [Console do App Runner](#)— Gerencie e monitore seu serviço usando o console do App Runner.
- [Gerenciando seu serviço](#)— Gerencie o ciclo de vida do seu serviço App Runner.
- [Observabilidade](#)— Obtenha visibilidade das operações do serviço App Runner monitorando métricas, lendo registros, gerenciando eventos, rastreando chamadas de ação de serviço e rastreando eventos de aplicativos, como chamadas HTTP.
- [Arquivo de configuração do App Runner](#)— Uma forma baseada em configuração de especificar opções para o comportamento de compilação e tempo de execução do seu serviço App Runner.
- [API do App Runner](#)— Use a interface de programação de aplicativos (API) do App Runner para criar, ler, atualizar e excluir recursos do App Runner.

- [Segurança](#)— As diferentes maneiras pelas quais você garante a segurança na nuvem enquanto usa o App Runner e outros serviços. AWS

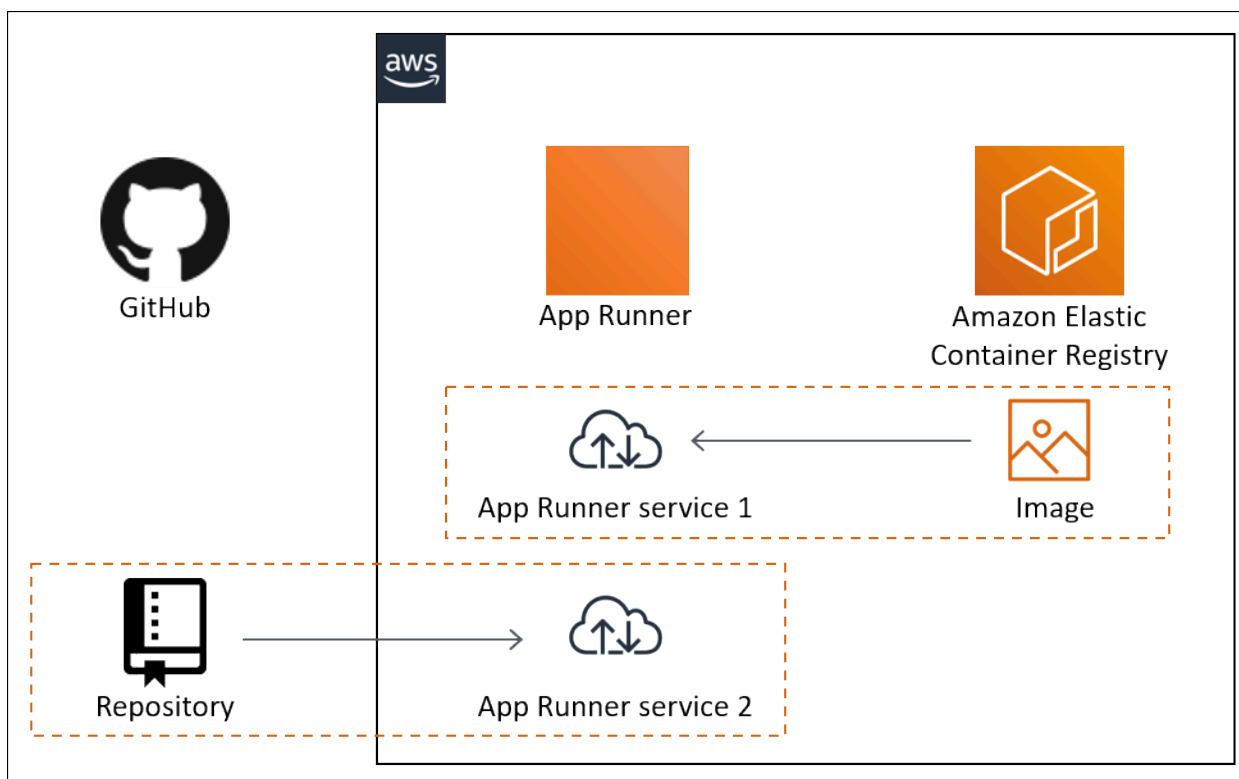
Arquitetura e conceitos do App Runner

AWS App Runner pega seu código-fonte ou imagem fonte de um repositório e cria e mantém um serviço web em execução para você no Nuvem AWS. Normalmente, você precisa chamar apenas uma ação do App Runner, [CreateService](#), para criar seu serviço.

Com um repositório de imagens de origem, você fornece uma imagem de ready-to-use contêiner que o App Runner pode implantar para executar seu serviço web. Com um repositório de código-fonte, você fornece seu código e instruções para criar e executar um serviço web e tem como alvo um ambiente de tempo de execução específico. O App Runner oferece suporte a várias plataformas de programação, cada uma com um ou mais tempos de execução gerenciados para as versões principais da plataforma.

No momento, o App Runner pode recuperar seu código-fonte de um [Bitbucket](#) ou [GitHub](#) repositório, ou pode recuperar sua imagem de origem do Amazon [Elastic Container Registry \(Amazon ECR\)](#) em seu. Conta da AWS

O diagrama a seguir mostra uma visão geral da arquitetura do serviço App Runner. No diagrama, há dois exemplos de serviços: um implanta o código-fonte do GitHub Amazon ECR e o outro implanta uma imagem fonte do Amazon ECR. O mesmo fluxo se aplica ao repositório Bitbucket.



Conceitos do App Runner

A seguir estão os principais conceitos relacionados ao seu serviço web que está sendo executado no App Runner:

- **Serviço App Runner** — Um AWS recurso que o App Runner usa para implantar e gerenciar seu aplicativo com base em seu repositório de código-fonte ou imagem de contêiner. Um serviço App Runner é uma versão em execução do seu aplicativo. Para obter mais informações sobre a criação de um serviço, consulte [the section called “Criação”](#).
- **Tipo de fonte** — [O tipo de repositório de origem que você fornece para implantar seu serviço App Runner: código-fonte ou imagem de origem](#).
- **Provedor de repositório** — O serviço de repositório que contém a fonte do seu aplicativo (por exemplo [GitHub](#), [Bitbucket](#) ou [Amazon ECR](#)).
- **Conexão App Runner** — Um AWS recurso que permite que o App Runner acesse uma conta de provedor de repositório (por exemplo, uma GitHub conta ou organização). Para obter mais informações sobre conexões, consulte [the section called “Conexões”](#).
- **Runtime** — Uma imagem base para implantar um repositório de código-fonte. O App Runner fornece uma variedade de tempos de execução gerenciados para diferentes plataformas e versões de programação. Para obter mais informações, consulte [Serviço baseado em código](#).
- **Implantação** — Uma ação que aplica uma versão do seu repositório de origem (código ou imagem) a um serviço do App Runner. A primeira implantação do serviço ocorre como parte da criação do serviço. Implantações posteriores podem ocorrer de duas maneiras:
 - **Implantação automática** — um recurso de CI/CD. Você pode configurar um serviço App Runner para criar automaticamente (para código-fonte) e implantar cada versão do seu aplicativo conforme ela aparece no repositório. Isso pode ser um novo commit em um repositório de código-fonte ou uma nova versão de imagem em um repositório de imagens de origem.
 - **Implantação manual** — Uma implantação em seu serviço App Runner que você inicia explicitamente.
- **Domínio personalizado** — Um domínio que você associa ao seu serviço App Runner. Os usuários do seu aplicativo web podem usar esse domínio para acessar seu serviço web em vez do subdomínio padrão do App Runner. Para obter mais informações, consulte [the section called “Nomes de domínios personalizados”](#).

 Note

Para aumentar a segurança de seus aplicativos App Runner, o [domínio*.awsapprunner.com é registrado na Lista Pública de Sufixos \(PSL\)](#). Para maior segurança, recomendamos que você use cookies com um `__Host-` prefixo se precisar definir cookies confidenciais no nome de domínio padrão para seus aplicativos App Runner. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

- **Manutenção** — Uma atividade que o App Runner executa ocasionalmente na infraestrutura que executa seu serviço App Runner. Quando a manutenção está em andamento, o status do serviço muda temporariamente para `OPERATION_IN_PROGRESS` (Operação em andamento no console) por alguns minutos. As ações em seu serviço (por exemplo, implantação, atualização de configuração, pausa/retomada ou exclusão) são bloqueadas durante esse período. Tente a ação novamente alguns minutos depois, quando o status do serviço retornar para `RUNNING`.

 Note

Se sua ação falhar, isso não significa que o serviço App Runner está inativo. Seu aplicativo está ativo e continua processando solicitações. É improvável que seu serviço sofra algum tempo de inatividade.

Em particular, o App Runner migra seu serviço se detectar problemas no hardware subjacente que hospeda o serviço. Para evitar qualquer tempo de inatividade do serviço, o App Runner implanta seu serviço em um novo conjunto de instâncias e transfere o tráfego para elas (uma implantação azul esverdeada). Ocasionalmente, você pode ver um ligeiro aumento temporário nas cobranças.

Configurações compatíveis com o App Runner

Ao configurar um serviço App Runner, você especifica a configuração virtual de CPU e memória a ser alocada ao seu serviço. Você paga com base na configuração computacional selecionada. Para obter mais informações sobre a definição de preço, consulte [Definição de preço do AWS Resource Groups](#).

A tabela a seguir fornece informações sobre as configurações de vCPU e memória compatíveis com o App Runner:

CPU	Memória
0.25 vCPU	0,5 GB
0.25 vCPU	1 GB
0.5 vCPU	1 GB
1 vCPU	2 GB
1 vCPU	3 GB
1 vCPU	4 GB
2 vCPU	4 GB
2 vCPU	6 GB
4 vCPU	8 GB
4 vCPU	10 GB
4 vCPU	12 GB

Recursos do App Runner

Ao usar o App Runner, você cria e gerencia alguns tipos de recursos no seu Conta da AWS. Esses recursos são usados para acessar seu código e gerenciar seus serviços.

A tabela a seguir fornece uma visão geral desses recursos:

Nome do recurso	Descrição
Service	Representa uma versão em execução do seu aplicativo. A maior parte do restante deste guia descreve os tipos de serviços, o gerenciamento, a configuração e o monitoramento.

Nome do recurso	Descrição
	ARN: <code>arn:aws:apprunner: <i>region</i>:<i>account-id</i> :service/<i>service-name</i> [/<i>service-id</i>]</code>
Connection	Fornece aos serviços do App Runner acesso a repositórios privados armazenados com fornecedores terceirizados. Existe como um recurso separado para compartilhamento em vários serviços. Para obter mais informações sobre conexões, consulte the section called “Conexões”. ARN: <code>arn:aws:apprunner: <i>region</i>:<i>account-id</i> :connection/<i>connection-name</i> [/<i>connection-id</i>]</code>
AutoScalingConfiguration	Fornece aos serviços do App Runner configurações que controlam o escalonamento automático do seu aplicativo. Existe como um recurso separado para compartilhamento em vários serviços. Para obter mais informações sobre a escalabilidade automática, consulte the section called “Ajuste de escala automático”. ARN: <code>arn:aws:apprunner: <i>region</i>:<i>account-id</i> :autoscalingconfiguration/<i>config-name</i> [/<i>config-revision</i> [/<i>config-id</i>]]</code>
ObservabilityConfiguration	Configura recursos adicionais de observabilidade do aplicativo para seus serviços do App Runner. Existe como um recurso separado para compartilhamento em vários serviços. Para obter mais informações sobre a configuração de observabilidade, consulte the section called “Configuração de observabilidade”. ARN: <code>arn:aws:apprunner: <i>region</i>:<i>account-id</i> :observabilityconfiguration/<i>config-name</i> [/<i>config-revision</i> [/<i>config-id</i>]]</code>

Nome do recurso	Descrição
VpcConnector	Define as configurações de VPC para seus serviços do App Runner. Existe como um recurso separado para compartilhamento em vários serviços. Para obter mais informações sobre a funcionalidade da VPC, consulte the section called “Tráfego de saída”. ARN: <code>arn:aws:apprunner: <i>region</i>:<i>account-id</i> :vpcconnector/ <i>connector-name</i> [/<i>connector-revision</i> [/<i>connector-id</i>]]</code>
VpcIngressConnection	É um AWS App Runner recurso usado para configurar o tráfego de entrada. Ele estabelece uma conexão entre um endpoint de interface VPC e o serviço App Runner, para tornar seu serviço App Runner acessível somente de dentro de uma Amazon VPC. Para obter mais informações sobre a funcionalidade do VPCIngress Connection, consulte the section called “Habilitar endpoint privado”. ARN: <code>arn:aws:apprunner: <i>region</i>:<i>account-id</i> :vpcingressconnection/ <i>vpc-ingress-connection-name</i> [/<i>connector-id</i>]]</code>

Cotas de recursos do App Runner

AWS impõe algumas cotas (também conhecidas como limites) em sua conta para o uso de AWS recursos em cada uma. Região da AWS A tabela a seguir lista as cotas relacionadas aos recursos do App Runner. As cotas também são listadas nos [AWS App Runner endpoints e as cotas](#) no. Referência geral da AWS

Cota de recursos	Descrição	Valor padrão	Ajustável?
Services	O número máximo de serviços que você pode criar em sua conta para cada um Região da AWS.	30	✓ Sim

Cota de recursos		Descrição	Valor padrão	Ajustável?
Connections		O número máximo de conexões que você pode criar em sua conta para cada uma Região da AWS. Você pode usar uma única conexão em vários serviços.	10	✓ Sim
Auto scaling configurações	Nomes	O número máximo de nomes exclusivos que você pode ter nas configurações de escalonamento automático criadas em sua conta para cada um. Região da AWSÉ possível usar uma única configuração de ajuste de escala automático em vários serviços.	10	✓ Sim
	revisões por nome	O número máximo de revisões de configuração de auto scaling que você pode criar em sua conta Região da AWS para cada nome exclusivo. Você pode usar uma única revisão de configuração de auto scaling em vários serviços.	5	× Não
Observability configurações	Nomes	O número máximo de nomes exclusivos que você pode ter nas configurações de observabilidade que você cria em sua conta para cada um. Região da AWSÉ possível usar uma única configuração de observabilidade em vários serviços.	10	✓ Sim
	revisões por nome	O número máximo de revisões de configuração de observabilidade que você pode criar em sua conta Região da AWS para cada nome exclusivo. Você pode usar uma única revisão de configuração de observabilidade em vários serviços.	10	× Não

Cota de recursos	Descrição	Valor padrão	Ajustável?
VPC connectors	O número máximo de conectores VPC que você pode criar na sua conta para cada um. Região da AWSÉ possível usar um único conector VPC em vários serviços.	10	✓ Sim
VPC Ingress Connection	O número máximo de conexões de entrada de VPC que você pode criar em sua conta para cada uma. Região da AWS Você pode usar uma única conexão de entrada de VPC para acessar vários serviços do App Runner.	1	× Não

A maioria das cotas é ajustável e você pode solicitar um aumento de cota para elas. Para obter mais informações, consulte [Solicitar um aumento da cota](#) no Guia do usuário do Service Quotas.

Serviço App Runner baseado em uma imagem de origem

Você pode usar AWS App Runner para criar e gerenciar serviços com base em dois tipos fundamentalmente diferentes de fonte de serviço: código-fonte e imagem de origem.

Independentemente do tipo de fonte, o App Runner se encarrega de iniciar, executar, escalar e balancear a carga do seu serviço. Você pode usar o recurso de CI/CD do App Runner para rastrear alterações em sua imagem ou código fonte. Quando o App Runner descobre uma alteração, ele cria automaticamente (para o código-fonte) e implanta a nova versão no seu serviço App Runner.

Este capítulo discute serviços com base em uma imagem de origem. Para obter informações sobre serviços baseados no código-fonte, consulte [Serviço baseado em código](#).

Uma imagem de origem é uma imagem de contêiner pública ou privada armazenada em um repositório de imagens. Você aponta o App Runner para uma imagem e ele inicia um serviço executando um contêiner com base nessa imagem. Nenhum estágio de construção é necessário. Em vez disso, você fornece uma ready-to-deploy imagem.

Note

Ao fornecer imagens de contêiner, você é responsável por atualizar e corrigir essas imagens regularmente. Enquanto o App Runner gerencia a infraestrutura, você deve garantir a segurança e o up-to-date status das imagens de contêiner fornecidas. Para obter mais informações, consulte a documentação do [AWS App Runner](#)

Provedores de repositórios de imagens

O App Runner é compatível com os seguintes provedores de repositórios de imagens:

- Amazon Elastic Container Registry (Amazon ECR) — Armazena imagens que são privadas para um. Conta da AWS
- Amazon Elastic Container Registry Public (Amazon ECR Public) — Armazena imagens que podem ser lidas publicamente.

Casos de uso do provedor

- [Usando uma imagem armazenada no Amazon ECR em sua conta AWS](#)

- [Usando uma imagem armazenada no Amazon ECR em uma conta diferente AWS](#)
- [Usando uma imagem armazenada no Amazon ECR Public](#)

Usando uma imagem armazenada no Amazon ECR em sua conta AWS

O [Amazon ECR](#) armazena imagens em repositórios. Existem repositórios públicos e privados.

Para implantar sua imagem em um serviço App Runner a partir de um repositório privado, o App Runner precisa de permissão para ler sua imagem do Amazon ECR. Para dar essa permissão ao App Runner, você precisa fornecer ao App Runner uma função de acesso. Essa é uma função AWS Identity and Access Management (IAM) que tem as permissões de ação necessárias do Amazon ECR. Ao usar o console do App Runner para criar o serviço, você pode escolher uma função existente na sua conta. Como alternativa, você pode usar o console do IAM para criar um novo papel personalizado. Ou você pode escolher que o console do App Runner crie uma função para você com base nas políticas gerenciadas.

Ao usar a API App Runner ou a AWS CLI, você conclui um processo de duas etapas. Primeiro, você usa o console do IAM para criar uma função de acesso. Você pode usar uma política gerenciada fornecida pelo App Runner ou inserir suas próprias permissões personalizadas. Em seguida, você fornece a função de acesso durante a criação do serviço usando a ação [CreateService](#) da API.

Para obter informações sobre a criação do serviço App Runner, consulte [the section called “Criação”](#).

Usando uma imagem armazenada no Amazon ECR em uma conta diferente AWS

Ao criar um serviço App Runner, você pode usar uma imagem armazenada em um repositório Amazon ECR que pertence a uma AWS conta diferente daquela em que seu serviço está. Há algumas considerações adicionais que você deve ter em mente ao usar uma imagem de várias contas, além das listadas na seção anterior sobre a imagem da mesma conta.

- O repositório de várias contas deve ter uma política anexada a ele. A política do repositório fornece à sua função de acesso permissões para ler imagens no repositório. Use a política a seguir para essa finalidade. *access-role-arn* Substitua pelo Amazon Resource Name (ARN) da sua função de acesso.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

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

Para obter informações sobre como anexar uma política de repositório a um repositório do Amazon ECR, consulte [Definir uma declaração de política de repositório no](#) Guia do usuário do Amazon Elastic Container Registry.

- O App Runner não oferece suporte à implantação automática de imagens do Amazon ECR em uma conta diferente daquela em que seu serviço está.

Usando uma imagem armazenada no Amazon ECR Public

[O Amazon ECR Public](#) armazena imagens legíveis publicamente. Essas são as principais diferenças entre o Amazon ECR e o Amazon ECR Public que você deve conhecer no contexto dos serviços do App Runner:

- As imagens públicas do Amazon ECR podem ser lidas publicamente. Você não precisa fornecer uma função de acesso ao criar um serviço baseado em uma imagem pública do Amazon ECR. O repositório não precisa de nenhuma política anexada a ele.
- O App Runner não oferece suporte à implantação automática (contínua) de imagens públicas do Amazon ECR.

Inicie um serviço diretamente do Amazon ECR Public

Você pode iniciar diretamente imagens de contêiner de aplicativos web compatíveis que estão hospedados na [Galeria Pública do Amazon ECR](#) como serviços web executados no App Runner. Ao navegar na galeria, procure Launch with App Runner na página da galeria para ver uma imagem. Uma imagem com essa opção é compatível com o App Runner. Para obter mais informações sobre

a galeria, consulte [Usando a Galeria Pública do Amazon ECR](#) no guia do usuário do Amazon ECR Public.



Para iniciar uma imagem da galeria como um serviço App Runner

1. Na página da galeria de uma imagem, escolha Iniciar com o App Runner.

Resultado: o console do App Runner é aberto em uma nova guia do navegador. O console exibe o assistente de criação de serviço, com a maioria dos novos detalhes de serviço necessários pré-preenchidos.

2. Se você quiser criar seu serviço em uma AWS região diferente daquela que o console está mostrando, escolha a região exibida no cabeçalho do console. Em seguida, selecione outra região.
3. Em Porta, insira o número da porta que o aplicativo de imagem escuta. Normalmente, você pode encontrá-la na página da galeria da imagem.
4. Opcionalmente, altere quaisquer outros detalhes de configuração.
5. Escolha Avançar, revise as configurações e escolha Criar e implantar.

Exemplo de imagem

A equipe do App Runner mantém a imagem de hello-app-runner exemplo em uma galeria pública do Amazon ECR. Você pode usar esse exemplo para começar a criar um serviço App Runner baseado em imagens. Para obter mais informações, consulte [hello-app-runner](#).

Serviço App Runner baseado no código-fonte

Você pode usar AWS App Runner para criar e gerenciar serviços com base em dois tipos fundamentalmente diferentes de fonte de serviço: código-fonte e imagem de origem.

Independentemente do tipo de fonte, o App Runner se encarrega de iniciar, executar, escalar e balancear a carga do seu serviço. Você pode usar o recurso de CI/CD do App Runner para rastrear alterações em sua imagem ou código fonte. Quando o App Runner descobre uma alteração, ele cria automaticamente (para o código-fonte) e implanta a nova versão no seu serviço App Runner.

Este capítulo discute serviços baseados no código-fonte. Para obter informações sobre serviços baseados em uma imagem de origem, consulte [Serviço baseado em imagem](#).

O código-fonte é o código do aplicativo que o App Runner cria e implanta para você. Você aponta o App Runner para um [diretório de origem](#) em um repositório de código e escolhe um tempo de execução adequado que corresponda a uma versão da plataforma de programação. O App Runner cria uma imagem baseada na imagem base do tempo de execução e no código do seu aplicativo. Em seguida, ele inicia um serviço que executa um contêiner com base nessa imagem.

O App Runner fornece tempos de execução gerenciados convenientes e específicos da plataforma. Cada um desses tempos de execução cria uma imagem de contêiner a partir do código-fonte e adiciona dependências de tempo de execução da linguagem à sua imagem. Você não precisa fornecer instruções de configuração e criação de contêineres, como um Dockerfile.

Os subtópicos deste capítulo discutem as várias plataformas suportadas pelo App Runner — plataformas gerenciadas que fornecem tempos de execução gerenciados para diferentes ambientes e versões de programação.

Tópicos

- [Provedores de repositórios de código-fonte](#)
- [Diretório de origem](#)
- [Plataformas gerenciadas do App Runner](#)
- [Versões de tempo de execução gerenciadas e a compilação do App Runner](#)
- [Usar a plataforma Python do](#)
- [Usar a plataforma Node.js do](#)
- [Usando a plataforma Java](#)

- [Usar a plataforma .NET do](#)
- [Usar a plataforma PHP do](#)
- [Usar a plataforma Ruby do](#)
- [Usar a plataforma Go do](#)

Provedores de repositórios de código-fonte

O App Runner implanta seu código-fonte lendo-o em um repositório de código-fonte. [O App Runner é compatível com dois provedores de repositórios de código-fonte: GitHub e o Bitbucket.](#)

Implantação a partir do seu provedor de repositório de código-fonte

Para implantar seu código-fonte em um serviço do App Runner a partir de um repositório de código-fonte, o App Runner estabelece uma conexão com ele. Ao usar o console do App Runner para [criar um serviço](#), você fornece detalhes da conexão e um diretório de origem para o App Runner implantar seu código-fonte.

Conexões

Você fornece detalhes da conexão como parte do procedimento de criação do serviço. Quando você usa a API App Runner ou a AWS CLI, uma conexão é um recurso separado. Primeiro, você cria a conexão usando a ação [CreateConnection](#) da API. Em seguida, você fornece o ARN da conexão durante a criação do serviço usando a ação da [CreateService](#) API.

Diretório de origem

Ao criar um serviço, você também fornece um diretório de origem. Por padrão, o App Runner usa o diretório raiz do seu repositório como o diretório de origem. O diretório de origem é o local no seu repositório de código-fonte que armazena o código-fonte e os arquivos de configuração do seu aplicativo. Os comandos build e start também são executados a partir do diretório de origem. Ao usar a API App Runner ou a AWS CLI para criar ou atualizar um serviço, você fornece o diretório de origem nas ações da [UpdateService](#) API [CreateService](#)ee. Para obter mais informações, consulte a seção [Diretório de origem](#) abaixo.

Para obter mais informações sobre a criação do serviço App Runner, consulte [the section called “Criação”](#). Para obter mais informações sobre as conexões do App Runner, consulte [the section called “Conexões”](#).

Diretório de origem

Ao criar um serviço App Runner, você pode fornecer o diretório de origem, junto com o repositório e a ramificação. Defina o valor do campo Diretório de origem para o caminho do diretório do repositório que armazena o código-fonte e os arquivos de configuração do aplicativo. O App Runner executa os comandos build e start a partir do caminho do diretório de origem fornecido por você.

Insira o valor do caminho do diretório de origem como absoluto a partir do diretório raiz do repositório. Se você não especificar um valor, o padrão é o diretório de nível superior do repositório, também conhecido como diretório raiz do repositório.

Você também tem a opção de fornecer caminhos de diretório de origem diferentes além do diretório de repositório de nível superior. Isso suporta uma arquitetura de repositório monorepo, o que significa que o código-fonte de vários aplicativos é armazenado em um repositório. Para criar e oferecer suporte a vários serviços do App Runner a partir de um único monorepo, especifique diretórios de origem diferentes ao criar cada serviço.

Note

Se você especificar o mesmo diretório de origem para vários serviços do App Runner, os dois serviços serão implantados e operados individualmente.

Se você optar por usar um arquivo de `apprunner.yaml` configuração para definir seus parâmetros de serviço, coloque-o na pasta do diretório de origem do repositório.

Se a opção de gatilho de implantação estiver definida como Automática, as alterações confirmadas no diretório de origem acionarão uma implantação automática. Somente as alterações no caminho do diretório de origem acionarão uma implantação automática. É importante entender como a localização do diretório de origem afeta o escopo de uma implantação automática. Para obter mais informações, consulte Implantações automatizadas em [Métodos de implantação](#).

Note

Se seu serviço App Runner usa os tempos de execução gerenciados do PHP e você gostaria de designar um diretório de origem diferente do repositório raiz padrão, é importante usar a versão correta do tempo de execução do PHP. Para obter mais informações, consulte [Usar a plataforma PHP do](#).

Plataformas gerenciadas do App Runner

As plataformas gerenciadas do App Runner fornecem tempos de execução gerenciados para vários ambientes de programação. Cada tempo de execução gerenciado facilita a criação e a execução de contêineres com base em uma versão de uma linguagem de programação ou ambiente de execução. Quando você usa um tempo de execução gerenciado, o App Runner começa com uma imagem de tempo de execução gerenciada. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém um pacote de execução de linguagem, bem como algumas ferramentas e pacotes de dependências populares. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Versões de tempo de execução gerenciadas e a compilação do App Runner

O App Runner oferece um processo de compilação atualizado para aplicativos que são executados nos tempos de execução da versão principal mais recente. Esse processo de construção revisado é mais rápido e eficiente. Ele também cria uma imagem final com um espaço menor que contém apenas o código-fonte, os artefatos de construção e os tempos de execução necessários para executar seu aplicativo.

Nós nos referimos ao processo de compilação mais recente como compilação revisada do App Runner e ao processo de compilação original como compilação original do App Runner. Para evitar alterações nas versões anteriores das plataformas de tempo de execução, o App Runner só aplica

a compilação revisada a versões específicas de tempo de execução, normalmente lançamentos principais recém-lançados.

Introduzimos um novo componente no arquivo de `apprunner.yaml` configuração para tornar a compilação revisada compatível com versões anteriores para um caso de uso muito específico e também para fornecer mais flexibilidade para configurar a compilação do seu aplicativo. Esse é o `pre-run` parâmetro opcional. Explicamos quando usar esse parâmetro junto com outras informações úteis sobre as compilações nas seções a seguir.

A tabela a seguir mostra qual versão da compilação do App Runner se aplica a versões específicas de tempo de execução gerenciado. Continuaremos atualizando este documento para mantê-lo informado sobre nossos tempos de execução atuais.

Plataforma	Construção original	Construção revisada
Python – Informações de lançamento	• Python 3.8 • Python 3.7	• Python 3.11 (!)
Node.js: Informações de lançamento	• Node.js 16 • Node.js 14 • Node.js 12	• Node.js 22 • Node.js 18
Correto — Informações de lançamento	• Corretto 11 • Corretto 8	
.NET: Informações de lançamento	• .NET 6	
PHP: Informações de lançamento	• PHP 8.1	
Ruby: Informações de lançamento	• Rubi 3.1	
Go: Informações de lançamento o	• Go 1	

⚠ Important

Python 3.11 — Temos recomendações específicas para a configuração de compilação de serviços que usam o tempo de execução gerenciado do Python 3.11. Para obter mais informações, consulte [Explicações para versões específicas de tempo de execução](#) o tópico da plataforma Python.

Saiba mais sobre as compilações e a migração do App Runner

Ao migrar seu aplicativo para um novo tempo de execução que usa a compilação revisada, talvez seja necessário modificar um pouco sua configuração de compilação.

Para contextualizar as considerações de migração, primeiro descreveremos os processos de alto nível da compilação original do App Runner e da versão revisada. Seguiremos com uma seção que descreve atributos específicos sobre seu serviço que podem exigir algumas atualizações de configuração.

A versão original do App Runner

O processo original de criação do aplicativo App Runner aproveita o AWS CodeBuild serviço. As etapas iniciais são baseadas em imagens selecionadas pelo CodeBuild serviço. Segue um processo de criação do Docker que usa a imagem de tempo de execução gerenciada aplicável do App Runner como imagem base.

As etapas gerais são as seguintes:

1. Execute pre-build comandos em uma imagem CodeBuild com curadoria.

Os pre-build comandos são opcionais. Eles só podem ser especificados no arquivo `apprunner.yaml` de configuração.

2. Execute os build CodeBuild comandos usando a mesma imagem da etapa anterior.

Os build comandos são obrigatórios. Eles podem ser especificados no console do App Runner, na API do App Runner ou no arquivo de `apprunner.yaml` configuração.

3. Execute uma compilação do Docker para gerar uma imagem com base na imagem de tempo de execução gerenciada do App Runner para sua plataforma e versão de tempo de execução específicas.

4. Copie o `/app` diretório da imagem que geramos na Etapa 2. O destino é a imagem baseada na imagem de tempo de execução gerenciada do App Runner, que geramos na Etapa 3.
5. Execute os `build` comandos novamente na imagem de tempo de execução gerenciada do App Runner gerada. Executamos os comandos de construção novamente para gerar artefatos de construção a partir do código-fonte no `/app` diretório que copiamos para ele na Etapa 4. Essa imagem será posteriormente implantada pelo App Runner para executar seu serviço web em um contêiner.

Os `build` comandos são obrigatórios. Eles podem ser especificados no console do App Runner, na API do App Runner ou no arquivo de `apprunner.yaml` configuração.

6. Execute `post-build` comandos na CodeBuild imagem a partir da Etapa 2.

Os `post-build` comandos são opcionais. Eles só podem ser especificados no arquivo `apprunner.yaml` de configuração.

Após a conclusão da compilação, o App Runner implanta a imagem de tempo de execução gerenciada do App Runner gerada na Etapa 5 para executar seu serviço web em um contêiner.

A versão revisada do App Runner

O processo de construção revisado é mais rápido e eficiente do que o processo de construção original descrito na seção anterior. Ele elimina a duplicação dos comandos de compilação que ocorre na compilação da versão anterior. Ele também cria uma imagem final com um espaço menor que contém apenas o código-fonte, os artefatos de construção e os tempos de execução necessários para executar seu aplicativo.

Esse processo de compilação usa uma compilação de vários estágios do Docker. As etapas gerais do processo são as seguintes:

1. Etapa de compilação — inicie um processo de compilação do docker que executa `pre-build` e `build` comanda sobre as imagens de compilação do App Runner.
 - a. Copie o código-fonte do aplicativo para o `/app` diretório.

Note

Esse `/app` diretório é designado como o diretório de trabalho em cada estágio da compilação do Docker.

b. Execute comandos da `pre-build`.

Os `pre-build` comandos são opcionais. Eles só podem ser especificados no arquivo `apprunner.yaml` de configuração.

c. Execute os `build` comandos.

Os `build` comandos são obrigatórios. Eles podem ser especificados no console do App Runner, na API do App Runner ou no arquivo de `apprunner.yaml` configuração.

2. Etapa de empacotamento — gera a imagem final do contêiner do cliente, que também se baseia na imagem de execução do App Runner.

- a. Copie o `/app` diretório do estágio anterior de compilação para a nova imagem Executar. Isso inclui o código-fonte do seu aplicativo e os artefatos de construção do estágio anterior.
- b. Execute os `pre-run` comandos. Se você precisar modificar a imagem de tempo de execução fora do `/app` diretório usando os `build` comandos, adicione os mesmos comandos ou os obrigatórios a esse segmento do arquivo de `apprunner.yaml` configuração.

Esse é um novo parâmetro que foi introduzido para dar suporte à versão revisada do App Runner.

Os `pre-run` comandos são opcionais. Eles só podem ser especificados no arquivo `apprunner.yaml` de configuração.

 Observações

- Os `pre-run` comandos são suportados somente pela compilação revisada. Não os adicione ao arquivo de configuração se seu serviço usar versões de tempo de execução que usam a compilação original.
- Se você não precisar modificar nada fora do `/app` diretório com os `build` comandos, não precisará especificar `pre-run` comandos.

3. Estágio pós-construção — Esse estágio é retomado a partir do estágio de compilação e `post-build` executa comandos.

- a. Execute os `post-build` comandos dentro do `/app` diretório.

Os `post-build` comandos são opcionais. Eles só podem ser especificados no arquivo `apprunner.yaml` de configuração.

Após a conclusão da compilação, o App Runner implanta a imagem Run para executar seu serviço Web em um contêiner.

Note

Não se deixe enganar pelas env entradas na seção Executar `apprunner.yaml` ao configurar o processo de compilação. Mesmo que o parâmetro do `pre-run` comando, referenciado na Etapa 2 (b), resida na seção Executar, não use o env parâmetro na seção Executar para configurar sua compilação. Os `pre-run` comandos fazem referência apenas às env variáveis definidas na seção Construir do arquivo de configuração. Para obter mais informações, consulte o [Seção de execução](#) capítulo do arquivo de configuração do App Runner.

Requisitos de serviço para consideração da migração

Se o ambiente do seu aplicativo tiver um desses dois requisitos, você precisará revisar sua configuração de compilação adicionando `pre-run` comandos.

- Se você precisar modificar qualquer coisa fora do `/app` diretório com os `build` comandos.
- Se você precisar executar os `build` comandos duas vezes para criar o ambiente necessário. Esse é um requisito muito incomum. A grande maioria das construções não fará isso.

Modificações fora do `/app` diretório

- A [versão revisada do App Runner](#) pressupõe que seu aplicativo não tenha dependências fora do diretório. `/app`
- Os comandos que você fornece com o `apprunner.yaml` arquivo, a API App Runner ou o console do App Runner devem gerar artefatos de construção no diretório. `/app`
- Você pode modificar os `post-build` comandos `pre-build`, e para garantir que todos os artefatos de construção estejam no `/app` diretório.
- Se seu aplicativo exigir que a compilação modifique ainda mais a imagem gerada para seu serviço, fora do `/app` diretório, você poderá usar os novos `pre-run` comandos no `apprunner.yaml`. Para obter mais informações, consulte [Definindo as opções do serviço App Runner usando um arquivo de configuração](#).

Executando os **build** comandos duas vezes

- A [compilação original do App Runner](#) executa build os comandos duas vezes, primeiro na Etapa 2 e depois novamente na Etapa 5. A versão revisada do App Runner corrige essa redundância e só executa os build comandos uma vez. Se seu aplicativo tiver um requisito incomum para que os build comandos sejam executados duas vezes, a versão revisada do App Runner oferece a opção de especificar e executar os mesmos comandos novamente usando o `pre-run` parâmetro. Isso mantém o mesmo comportamento de construção dupla.

Usar a plataforma Python do

A plataforma AWS App Runner Python fornece tempos de execução gerenciados. Cada tempo de execução facilita a criação e a execução de contêineres com aplicativos web baseados em uma versão do Python. Quando você usa um tempo de execução do Python, o App Runner começa com uma imagem gerenciada do tempo de execução do Python. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do Python e algumas ferramentas e pacotes de dependências populares. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do seu código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

Para nomes e versões válidos do tempo de execução do Python, consulte [the section called “Informações de lançamento”](#)

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe da versão para tempos de execução do Python: `major[.minor[.patch]]`

Por exemplo: 3.8.5

Os exemplos a seguir demonstram o bloqueio de versões:

- 3.8— Bloqueie as versões principais e secundárias. O App Runner atualiza somente as versões de patch.
- 3.8.5— Bloqueie uma versão específica do patch. O App Runner não atualiza sua versão de tempo de execução.

Tópicos

- [Configuração de tempo de execução do Python](#)
- [Explicações para versões específicas de tempo de execução](#)
- [Exemplos de tempo de execução em Python](#)
- [Informações sobre o lançamento do Python Runtime](#)

Configuração de tempo de execução do Python

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os StartCommand membros BuildCommand e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Explicações para versões específicas de tempo de execução

Note

O App Runner agora executa um processo de compilação atualizado para aplicativos com base nas seguintes versões de tempo de execução: Python 3.11, Node.js 22 e Node.js 18. Se seu aplicativo for executado em qualquer uma dessas versões de tempo de execução, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#) para obter mais informações sobre o processo de compilação revisado. Os aplicativos que usam todas as outras versões de tempo de execução não são afetados e continuam usando o processo de criação original.

Python 3.11 (versão revisada do App Runner)

Use as configurações a seguir no `apprunner.yaml` para o tempo de execução gerenciado do Python 3.11.

- Defina a `runtime` chave na seção superior como `python311`

Example

```
runtime: python311
```

- Use o `pip3` em vez de `pip` para instalar dependências.
- Use o `python3` intérprete em vez de `python`
- Execute o `pip3` instalador como um `pre-run` comando. O Python instala dependências fora do diretório. `/app` Como o App Runner executa a compilação revisada do App Runner para Python 3.11, qualquer coisa instalada fora do `/app` diretório por meio de comandos na seção `Build` do arquivo será perdida. `apprunner.yaml` Para obter mais informações, consulte [A versão revisada do 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 -
```

Para obter mais informações, consulte também o [exemplo de um arquivo de configuração estendido para Python 3.11](#) posteriormente neste tópico.

Exemplos de tempo de execução em Python

Os exemplos a seguir mostram os arquivos de configuração do App Runner para criar e executar um serviço Python. O último exemplo é o código-fonte de um aplicativo Python completo que você pode implantar em um serviço de tempo de execução do Python.

Note

A versão de tempo de execução usada nesses exemplos é **3.7.7 3.11** e. Você pode substituí-lo por uma versão que você deseja usar. Para ver a versão mais recente de tempo de execução do Python compatível, consulte [the section called “Informações de lançamento”](#)

Arquivo mínimo de configuração do Python

Este exemplo mostra um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado em Python. Para as suposições que o App Runner faz com um arquivo de configuração mínimo, consulte [the section called “Exemplos de arquivos de configuração”](#)

O Python 3.11 usa os comandos `pip3` e `python3`. Para obter mais informações, consulte o [exemplo de um arquivo de configuração estendido para Python 3.11](#) posteriormente neste tópico.

Example apprunner.yaml

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

Arquivo de configuração estendido do Python

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado em Python.

Note

A versão de tempo de execução usada nesses exemplos é **3.7.7**. Você pode substituí-lo por uma versão que você deseja usar. Para ver a versão mais recente de tempo de execução do Python compatível, consulte [the section called “Informações de lançamento”](#)

O Python 3.11 usa os comandos `pip3` e `python3`. Para obter mais informações, consulte o exemplo de um arquivo de configuração estendido para Python 3.11 posteriormente neste tópico.

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"

```

Arquivo de configuração estendido do Python — Python 3.11 (usa a versão revisada)

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado do Python 3.11 no `apprunner.yaml`. Esse exemplo inclui uma `pre-run` seção, já que essa versão do Python usa a versão revisada do App Runner.

O `pre-run` parâmetro só é compatível com a versão revisada do App Runner. Não insira esse parâmetro no arquivo de configuração se o aplicativo usar versões de tempo de execução compatíveis com a compilação original do App Runner. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Note

A versão de tempo de execução usada nesses exemplos é **3.11**. Você pode substituí-lo por uma versão que você deseja usar. Para ver a versão mais recente de tempo de execução do Python compatível, consulte [the section called “Informações de lançamento”](#)

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"

```

Fonte completa do aplicativo Python

Este exemplo mostra o código-fonte de um aplicativo Python completo que você pode implantar em um serviço de tempo de execução do 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
```

Informações sobre o lançamento do Python Runtime

Este tópico lista os detalhes completos das versões de tempo de execução do Python compatíveis com o App Runner.

Versões de tempo de execução suportadas — versão revisada do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Python 3.11 (python311)	3.11.12	SQLite 3.50.0
	3.11.11	SQLite 3.49.1
	3.11.10	SQLite 3.46.1

Nome do runtime	Versões secundárias	Pacotes incluídos
	3.11.9	SQLite 3.46.1
	3.11.8	SQLite 3.45.2
	3.11.7	SQLite 3.4.2

Observações

- Python 3.11 — Temos recomendações específicas para a configuração de compilação de serviços que usam o tempo de execução gerenciado do Python 3.11. Para obter mais informações, consulte [Explicações para versões específicas de tempo de execução](#) o tópico da plataforma Python.
- O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Python 3 (python3)	3.8.20	SQLite 3.50,0
	3.8.16	SQLite 3.46.1
	3.8.15	SQLite 3.40,0
	3.7.16	SQLite 3.50,0
	3.7.15	SQLite 3.40,0
	3.7.10	SQLite 3.40,0

Note

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Usar a plataforma Node.js do

A plataforma AWS App Runner Node.js fornece tempos de execução gerenciados. Cada tempo de execução facilita a criação e a execução de contêineres com aplicativos da Web baseados em uma versão do Node.js. Quando você usa um tempo de execução do Node.js, o App Runner começa com uma imagem de tempo de execução do Node.js gerenciada. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do Node.js e algumas ferramentas. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

Para obter nomes e versões válidos do tempo de execução do Node.js, consulte [the section called “Informações de lançamento”](#).

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe da versão para tempos de execução do Node.js: `major[.minor[.patch]]`

Por exemplo: `22.14.0`

Os exemplos a seguir demonstram o bloqueio de versões:

- 22.14— Bloqueie as versões principais e secundárias. O App Runner atualiza somente as versões de patch.
- 22.14.0— Bloqueie uma versão específica do patch. O App Runner não atualiza sua versão de tempo de execução.

Tópicos

- [Configuração de tempo de execução do Node.js](#)
- [Explicações para versões específicas de tempo de execução](#)
- [Exemplos de tempo de execução do Node.js](#)
- [Informações sobre a versão de execução do Node.js](#)

Configuração de tempo de execução do Node.js

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os StartCommand membros BuildCommand e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Especificamente com os tempos de execução do Node.js, você também pode configurar a compilação e o tempo de execução usando um arquivo JSON nomeado `package.json` na raiz do seu repositório de origem. Usando esse arquivo, você pode configurar a versão do mecanismo

Node.js, os pacotes de dependências e vários comandos (aplicativos de linha de comando). Gerenciadores de pacotes como npm ou yarn interpretam esse arquivo como entrada para seus comandos.

Por exemplo:

- npm install instala pacotes definidos pelo devDependencies no dependencies e em package.json.
- npm start ou npm run start executa o comando definido pelo scripts/start no package.json.

Veja a seguir um exemplo de arquivo 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"
  }
}
```

Para obter mais informações sobre `package.json`, consulte [Criação de um arquivo package.json](#) no site do npm Docs.

Dicas

- Se seu `package.json` arquivo definir um `start` comando, você poderá usá-lo como um `run` comando no arquivo de configuração do App Runner, conforme mostra o exemplo a seguir.

Example

`package.json`

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

`apprunner.yaml`

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

- Quando você executa `npm install` em seu ambiente de desenvolvimento, o `npm` cria o `package-lock.json` arquivo. Esse arquivo contém um instantâneo das versões do pacote `npm` recém-instaladas. Depois disso, quando o `npm` instala dependências, ele usa essas versões exatas. Se você instalar o `yarn`, ele criará um `yarn.lock` arquivo. Confirme esses arquivos em seu repositório de código-fonte para garantir que seu aplicativo seja instalado com as versões das dependências com as quais você o desenvolveu e testou.
- Você também pode usar um arquivo de configuração do App Runner para configurar a versão do `Node.js` e o comando `start`. Quando você faz isso, essas definições substituem `package.json` as existentes. Um conflito entre a `node` versão `package.json` e o `runtime-version` valor no arquivo de configuração do App Runner faz com que a fase de construção do App Runner falhe.

Explicações para versões específicas de tempo de execução

Node.js 22 e Node.js 18 (versão revisada do App Runner)

O App Runner agora executa um processo de compilação atualizado para aplicativos com base nas seguintes versões de tempo de execução: Python 3.11, Node.js 22 e Node.js 18. Se seu aplicativo for executado em qualquer uma dessas versões de tempo de execução, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#) para obter mais informações sobre o processo de compilação revisado. Os aplicativos que usam todas as outras versões de tempo de execução não são afetados e continuam usando o processo de criação original.

Exemplos de tempo de execução do Node.js

Os exemplos a seguir mostram os arquivos de configuração do App Runner para criar e executar um serviço Node.js.

Note

A versão de tempo de execução usada nesses exemplos é **22.14.0**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão de tempo de execução mais recente compatível do Node.js, consulte [the section called “Informações de lançamento”](#).

Arquivo mínimo de configuração Node.js

Este exemplo mostra um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado pelo Node.js. Para as suposições que o App Runner faz com um arquivo de configuração mínimo, consulte [the section called “Exemplos de arquivos de configuração”](#)

Example apprunner.yaml

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

Arquivo de configuração Node.js estendido

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado pelo Node.js.

Note

A versão de tempo de execução usada nesses exemplos é **22.14.0**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão de tempo de execução mais recente compatível do Node.js, consulte [the section called “Informações de lançamento”](#).

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"
```


Arquivo de configuração Node.js estendido — Node.js 22 (usa compilação revisada)

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado pelo Node.js no `apprunner.yaml`. Esse exemplo inclui uma `pre-run` seção, já que essa versão do Node.js usa a versão revisada do App Runner.

O `pre-run` parâmetro só é compatível com a versão revisada do App Runner. Não insira esse parâmetro no arquivo de configuração se o aplicativo usar versões de tempo de execução compatíveis com a compilação original do App Runner. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Note

A versão de tempo de execução usada nesses exemplos é **22.14.0**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão de tempo de execução mais recente compatível do Node.js, consulte [the section called “Informações de lançamento”](#).

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"
```

Aplicativo Node.js com Grunt

Este exemplo mostra como configurar um aplicativo Node.js desenvolvido com o Grunt. [O Grunt](#) é um executor de JavaScript tarefas de linha de comando. Ele executa tarefas repetitivas e gerencia a automação de processos para reduzir o erro humano. Os plug-ins Grunt e Grunt são instalados e gerenciados usando o npm. Você configura o Grunt incluindo o `Gruntfile.js` arquivo na raiz do seu repositório de origem.

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

A versão de tempo de execução usada nesses exemplos é **22.14.0**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão de tempo de execução mais recente compatível do Node.js, consulte [the section called “Informações de lançamento”](#).

```
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
```

Informações sobre a versão de execução do Node.js

Note

A política de suspensão de uso padrão do App Runner é descontinuar um tempo de execução quando qualquer componente importante do tempo de execução chegar ao fim do suporte de longo prazo da comunidade (LTS) e as atualizações de segurança não estiverem mais disponíveis. Em alguns casos, o App Runner pode atrasar a suspensão de uso de um tempo de execução por um período limitado, além da end-of-support data da versão do idioma suportada pelo tempo de execução. Um exemplo desse caso pode ser estender o suporte a um tempo de execução para permitir que os clientes tenham tempo para a migração.

Este tópico lista os detalhes completos das versões de tempo de execução do Node.js suportadas pelo App Runner.

Versões de tempo de execução suportadas — versão revisada do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Node.js 22 (nodejs22)	22.16.0	npm 10.9.2, fio 1.22.22
Node.js 22 (nodejs22)	22.14.0	npm 10.9.2, fio 1.22.22

Note

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Versões de tempo de execução suportadas — versão revisada do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Node.js 18 (nodejs18)	18.20.8	npm 10.8.2, fio 1.22.22
	18.20.7	npm 10.8.2, fio 1.22.22
	18.20.6	npm 10.8.2, fio 1.22.22
	18.20.5	npm 10.8.2, fio 1.22.22
	18.20.4	npm 10.7.0, fio 1.22.22
	18.20.3	npm 10.7.0, fio 1.22.22
	18.20.2	npm 10, fio *
	18.19.1	npm 10, fio *
	18.19.0	npm 10, fio *

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Node.js 16 (nodejs16)	16.20.2	npm 8.19.4, fio 1.22.22
	16.20.1	npm 8.19.4, fio *
	16.20.0	npm 8.19.4, fio *
	16.19.1	npm 8.19.4, fio *
	16.19.0	npm 8.19.4, fio *
	16.18.1	npm 8.19.4, fio *
	16.17.1	npm 8.19.4, fio *
	16.17.0	npm 8.19.4, fio *

Nome do runtime	Versões secundárias	Pacotes incluídos
Node.js 14 (nodejs14)	14.21.3	npm 6.14.18, fio 1.22.22
	14.21.2	npm 6.14.18, fio *
	14.21.1	npm 6.14.18, fio *
	14.20.1	npm 6.14.18, fio *
	14.19.0	npm 6.14.18, fio *
Node.js 12 (nodejs12)	12.22.12	npm 6.14.16, fio 1.22.22
	12.21.0	npm 6.14.16, fio *

Usando a plataforma Java

A plataforma AWS App Runner Java fornece tempos de execução gerenciados. Cada tempo de execução facilita a criação e a execução de contêineres com aplicativos web baseados em uma versão Java. Quando você usa um Java Runner, o App Runner começa com uma imagem de tempo de execução Java gerenciada. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do Java e algumas ferramentas. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do seu código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

No momento, todos os tempos de execução Java compatíveis são baseados no Amazon Corretto. Para nomes e versões válidos do Java Runtime, consulte [the section called “Informações de lançamento”](#).

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo

de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe de versão para tempos de execução do Amazon Corretto:

Runtime	Sintaxe	Exemplo
corretto11	<code>11.0[.openjdk-update [.openjdk-build [.corretto-specific-revision]]]</code>	11.0.13.08.1
corretto8	<code>8[.openjdk-update [.openjdk-build [.corretto-specific-revision]]]</code>	8.312.07.1

Os exemplos a seguir demonstram o bloqueio de versões:

- 11.0.13— Bloqueie a versão de atualização do Open JDK. O App Runner atualiza somente compilações de nível inferior do Open JDK e do Amazon Corretto.
- 11.0.13.08.1— Bloqueie uma versão específica. O App Runner não atualiza sua versão de tempo de execução.

Tópicos

- [Configuração do Java Runtime](#)
- [Exemplos de tempo de execução em Java](#)
- [Informações sobre a versão do Java Runtime](#)

Configuração do Java Runtime

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os `StartCommand` membros `BuildCommand` e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Exemplos de tempo de execução em Java

Os exemplos a seguir mostram os arquivos de configuração do App Runner para criar e executar um serviço Java. O último exemplo é o código-fonte de um aplicativo Java completo que você pode implantar em um serviço de tempo de execução do Corretto 11.

Note

A versão de tempo de execução usada nesses exemplos é **11.0.13.08.1**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão mais recente do Java runtime compatível, consulte [the section called “Informações de lançamento”](#).

Arquivo de configuração do Minimal Corretto 11

Este exemplo mostra um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado do Corretto 11. Para as suposições que o App Runner faz com um arquivo de configuração mínimo, consulte.

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 .
```

Arquivo de configuração do Corretto 11 estendido

Este exemplo mostra como você pode usar todas as chaves de configuração com um tempo de execução gerenciado do Corretto 11.

Note

A versão de tempo de execução usada nesses exemplos é **11.0.13.08.1**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão mais recente do Java runtime compatível, consulte [the section called “Informações de lançamento”](#).

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"
```

Fonte completa do aplicativo Corretto 11

Este exemplo mostra o código-fonte de um aplicativo Java completo que você pode implantar em um serviço de tempo de execução do 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>

```

Informações sobre a versão do Java Runtime

Este tópico lista os detalhes completos das versões de tempo de execução do Java suportadas pelo App Runner.

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Corretto 11 (Corretto 11)	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

Nome do runtime	Versões secundárias	Pacotes incluídos
	11.0.23.9.1	Maven 3.9.8, Gradle 6.9.4
	11.0.22.7.1	Maven 3.9.6, Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.6, Gradle 6.9.4
	11.0.21.9.1	Maven 3.9.5, Gradle 6.9.4
	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)	8.452.09.2	Maven 3.9.9, Gradle 6.9.4
	8.452.09.1	Maven 3.9.9, Gradle 6.9.4
	8.442.06.1	Maven 3.9.9, Gradle 6.9.4
	8.432.06.1	Maven 3.9.9, Gradle 6.9.4
	8.422.05.1	Maven 3.9.9, Gradle 6.9.4
	8.412.08.1	Maven 3.9.8, Gradle 6.9.4
	8.402.08.1	Maven 3.9.6, Gradle 6.9.4
	8.392.08.1	Maven 3.9.6, Gradle 6.9.4
	8.382.05.1	Maven 3.9.4, Gradle 6.9.4
	8.372.07.1	Maven 3.9.3, Gradle 6.9.4

Nome do runtime	Versões secundárias	Pacotes incluídos
	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

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Usar a plataforma .NET do

A plataforma AWS App Runner do.NET fornece tempos de execução gerenciados. Cada tempo de execução facilita a criação e a execução de contêineres com aplicativos web baseados em uma versão.NET. Quando você usa um runtime do.NET, o App Runner começa com uma imagem de tempo de execução do.NET gerenciada. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do.NET e algumas ferramentas e pacotes de dependências populares. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateServiceAPI](#). Você também pode especificar um tempo de execução como parte do seu código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

Para nomes e versões válidos do runtime do.NET, consulte [the section called “Informações de lançamento”](#).

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe da versão para tempos de execução do.NET: `major[.minor[.patch]]`

Por exemplo: `6.0.9`

Os exemplos a seguir demonstram o bloqueio de versões:

- `6.0`— Bloqueie as versões principais e secundárias. O App Runner atualiza somente as versões de patch.
- `6.0.9`— Bloqueie uma versão específica do patch. O App Runner não atualiza sua versão de tempo de execução.

Tópicos

- [Configuração do runtime do.NET](#)
- [Exemplos de runtime do.NET](#)
- [Informações sobre a versão do runtime do.NET](#)

Configuração do runtime do.NET

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os `StartCommand` membros `BuildCommand` e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Exemplos de runtime do.NET

Os exemplos a seguir mostram os arquivos de configuração do App Runner para criar e executar um serviço.NET. O último exemplo é o código-fonte de um aplicativo.NET completo que você pode implantar em um serviço de runtime do.NET.

Note

A versão de tempo de execução usada nesses exemplos é **6.0.9**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão mais recente do runtime do.NET compatível, consulte [the section called “Informações de lançamento”](#).

Arquivo mínimo de configuração.NET

Este exemplo mostra um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado.NET. Para as suposições que o App Runner faz com um arquivo de configuração mínimo, consulte [the section called “Exemplos de arquivos de configuração”](#)

Example apprunner.yaml

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

Arquivo de configuração estendido.NET

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado.NET.

 Note

A versão de tempo de execução usada nesses exemplos é **6.0.9**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão mais recente do runtime do .NET compatível, consulte [the section called “Informações de lançamento”](#).

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"
```

Fonte completa do aplicativo.NET

Este exemplo mostra o código-fonte de um aplicativo.NET completo que você pode implantar em um serviço de runtime do .NET.

 Note

- Execute o comando a seguir para criar um aplicativo web do .NET 6 simples: `dotnet new web --name HelloWorldDotNetApp -f net6.0`

- Adicione o `apprunner.yaml` ao aplicativo web do.NET 6 criado.

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"
```

Informações sobre a versão do runtime do.NET

Este tópico lista os detalhes completos das versões de tempo de execução do.NET suportadas pelo App Runner.

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
.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

Nome do runtime	Versões secundárias	Pacotes incluídos
	6.0.26	.NET SDK 6.0.418
	6.0.25	.NET SDK 6.0.417
	6.0.24	.NET SDK 6.0.416
	6.0.22	.NET SDK 6.0.414
	6.0.21	.NET SDK 6.0.413
	6.0.20	.NET SDK 6.0.412
	6.0.19	.NET SDK 6.0.411
	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

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Usar a plataforma PHP do

A plataforma AWS App Runner PHP fornece tempos de execução gerenciados. Você pode usar cada tempo de execução para criar e executar contêineres com aplicativos web baseados em uma versão do PHP. Quando você usa um tempo de execução PHP, o App Runner começa com uma imagem de tempo de execução PHP gerenciada. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do PHP e algumas ferramentas. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do seu código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

Para nomes e versões válidos do tempo de execução do PHP, consulte [the section called “Informações de lançamento”](#).

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe da versão para tempos de execução do PHP: `major[.minor[.patch]]`

Por exemplo: `8.1.10`

Veja a seguir exemplos de bloqueio de versão:

- `8.1`— Bloqueie as versões principais e secundárias. O App Runner atualiza somente as versões de patch.
- `8.1.10`— Bloqueie uma versão específica do patch. O App Runner não atualiza sua versão de tempo de execução.

⚠ Important

Se você quiser especificar o [diretório de origem](#) do repositório de código para seu serviço App Runner em um local diferente do diretório raiz padrão do repositório, sua versão de tempo de execução gerenciada do PHP deve ser PHP 8.1.22 ou posterior. As versões de tempo de execução do PHP anteriores só 8.1.22 podem usar o diretório raiz padrão do código-fonte.

Tópicos

- [Configuração de tempo de execução do PHP](#)
- [Compatibilidade](#)
- [Exemplos de tempo de execução em PHP](#)
- [Informações sobre a versão do PHP runtime](#)

Configuração de tempo de execução do PHP

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os StartCommand membros BuildCommand e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Compatibilidade

Você pode executar seus serviços do App Runner na plataforma PHP usando um dos seguintes servidores web:

- Apache HTTP Server
- NGINX

Apache HTTP Server and NGINX são compatíveis com PHP-FPM. Você pode iniciar o Apache HTTP Server e NGINX usando uma das seguintes opções:

- [Supervisord](#) - Para obter mais informações sobre como executar um supervisord, consulte [Supervisores de execução](#).
- Script de inicialização

Para obter exemplos de como configurar seu serviço App Runner com a plataforma PHP usando o Apache HTTP Server ou NGINX, consulte [the section called “Fonte completa do aplicativo PHP”](#)

Estrutura do arquivo

O `index.php` deve ser instalado na `public` pasta abaixo do `root` diretório do servidor web.

Note

Recomendamos que os `supervisord.conf` arquivos `startup.sh` ou sejam armazenados no diretório raiz do servidor web. Certifique-se de que o `start` comando aponte para o local em que os `supervisord.conf` arquivos `startup.sh` ou estão armazenados.

A seguir está um exemplo da estrutura do arquivo se você estiver usando supervisord.

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

Veja a seguir um exemplo da estrutura do arquivo se você estiver usando um script de inicialização.

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

Recomendamos que você armazene essas estruturas de arquivo no [diretório de origem](#) do repositório de código designado para o serviço App Runner.

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

Important

Se você quiser especificar o [diretório de origem](#) do repositório de código para seu serviço App Runner em um local diferente do diretório raiz padrão do repositório, sua versão de tempo de execução gerenciada do PHP deve ser PHP 8.1.22 ou posterior. As versões de tempo de execução do PHP anteriores só 8.1.22 podem usar o diretório raiz padrão do código-fonte.

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Seu serviço usará os tempos de execução mais recentes por padrão, a menos que você tenha especificado o bloqueio de versão usando a `runtime-version` palavra-chave no arquivo de configuração do [App Runner](#).

Exemplos de tempo de execução em PHP

Veja a seguir exemplos de arquivos de configuração do App Runner usados para criar e executar um serviço PHP.

Arquivo mínimo de configuração PHP

O exemplo a seguir é um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado por PHP. Para obter mais informações sobre um arquivo de configuração mínimo, consulte [the section called “Exemplos de arquivos de configuração”](#).

Example apprunner.yaml

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

Arquivo de configuração PHP estendido

O exemplo a seguir usa todas as chaves de configuração com um tempo de execução gerenciado por PHP.

Note

A versão de tempo de execução usada nesses exemplos é **8.1.10**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão mais recente do PHP em tempo de execução compatível, consulte [the section called “Informações de lançamento”](#).

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"
```

Fonte completa do aplicativo PHP

Os exemplos a seguir são do código-fonte do aplicativo PHP que você pode usar para implantar em um serviço de tempo de execução PHP usando Apache HTTP Server ou NGINX. Esses exemplos pressupõem que você use a estrutura de arquivo padrão.

Executando a plataforma PHP com Apache HTTP Server usar supervisord

Example Estrutura do arquivo

Note

- O `supervisord.conf` arquivo pode ser armazenado em qualquer lugar do repositório. Certifique-se de que o `start` comando aponte para onde o `supervisord.conf` arquivo está armazenado.
- O `index.php` deve ser instalado na `public` pasta abaixo do `root` diretório.

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

Example supervisor.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>
```

Executando a plataforma PHP com Apache HTTP Server usar startup script

Example Estrutura do arquivo

Note

- O `startup.sh` arquivo pode ser armazenado em qualquer lugar do repositório. Certifique-se de que o `start` comando aponte para onde o `startup.sh` arquivo está armazenado.
- O `index.php` deve ser instalado na `public` pasta abaixo do `root` diretório.

```
/
## 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

- Certifique-se de salvar o `startup.sh` arquivo como executável antes de confirmá-lo em um repositório Git. Use `chmod +x startup.sh` para definir a permissão de execução em seu `startup.sh` arquivo.

- Se você não salvar o `startup.sh` arquivo como executável, insira `chmod +x startup.sh` como build comando no seu `apprunner.yaml` arquivo.

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>
```

Executando a plataforma PHP com NGINX usar supervisord

Example Estrutura do arquivo

Note

- O `supervisord.conf` arquivo pode ser armazenado em qualquer lugar do repositório. Certifique-se de que o `start` comando aponte para onde o `supervisord.conf` arquivo está armazenado.

- O `index.php` deve ser instalado na `public` pasta abaixo do `root` diretório.

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

Example supervisor.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>
```

Executando a plataforma PHP com NGINX usar startup script

Example Estrutura do arquivo

Note

- O `startup.sh` arquivo pode ser armazenado em qualquer lugar do repositório. Certifique-se de que o `start` comando aponte para onde o `startup.sh` arquivo está armazenado.
- O `index.php` deve ser instalado na `public` pasta abaixo do `root` diretório.

```
/
## 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

- Certifique-se de salvar o `startup.sh` arquivo como executável antes de confirmá-lo em um repositório Git. Use `chmod +x startup.sh` para definir a permissão de execução em seu `startup.sh` arquivo.
- Se você não salvar o `startup.sh` arquivo como executável, insira `chmod +x startup.sh` como build comando no seu `apprunner.yaml` arquivo.

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>
```

Informações sobre a versão do PHP runtime

Este tópico lista os detalhes completos das versões de tempo de execução do PHP suportadas pelo App Runner.

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
PHP 8.1 (php81)	8.1.32	
	8.1.31	
	8.1.29	
	8.1.28	
	8.1.27	
	8.1.26	
	8.1.24	
	8.1.22	
	8.1.21	
	8.1.20	
	8.1.19	
	8.1.17	
	8.1.16	
	8.1.14	

Nome do runtime	Versões secundárias	Pacotes incluídos
	8.1.13	
	8.1.12	
	8.1.10	

Note

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Usar a plataforma Ruby do

A plataforma AWS App Runner Ruby fornece tempos de execução gerenciados. Cada tempo de execução facilita a criação e a execução de contêineres com aplicativos web baseados em uma versão Ruby. Quando você usa um tempo de execução do Ruby, o App Runner começa com uma imagem gerenciada do tempo de execução do Ruby. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do Ruby e algumas ferramentas. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do seu código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

Para nomes e versões válidos do tempo de execução do Ruby, consulte [the section called “Informações de lançamento”](#).

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe da versão para tempos de execução do Ruby: `major[.minor[.patch]]`

Por exemplo: `3.1.2`

Os exemplos a seguir demonstram o bloqueio de versões:

- `3.1`— Bloqueie as versões principais e secundárias. O App Runner atualiza somente as versões de patch.
- `3.1.2`— Bloqueie uma versão específica do patch. O App Runner não atualiza sua versão de tempo de execução.

Tópicos

- [Configuração de tempo de execução do Ruby](#)
- [Exemplos de tempo de execução do Ruby](#)
- [Informações de lançamento do Ruby Runtime](#)

Configuração de tempo de execução do Ruby

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os `StartCommand` membros `BuildCommand` e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Exemplos de tempo de execução do Ruby

Os exemplos a seguir mostram os arquivos de configuração do App Runner para criar e executar um serviço Ruby.

Arquivo de configuração mínimo do Ruby

Este exemplo mostra um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado por Ruby. Para as suposições que o App Runner faz com um arquivo de configuração mínimo, consulte [the section called “Exemplos de arquivos de configuração”](#)

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
```

Arquivo de configuração Ruby estendido

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado por Ruby.

Note

A versão de tempo de execução usada nesses exemplos é **3.1.2**. Você pode substituí-lo por uma versão que você deseja usar. Para obter a versão mais recente de tempo de execução do Ruby compatível, consulte [the section called “Informações de lançamento”](#).

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"
```

Fonte completa do aplicativo Ruby

Esses exemplos mostram o código-fonte de um aplicativo Ruby completo que você pode implantar em um serviço de tempo de execução do Ruby.

Example servidor.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
```

Informações de lançamento do Ruby Runtime

Este tópico lista os detalhes completos das versões de tempo de execução do Ruby suportadas pelo App Runner.

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Ruby 3.1 (rubi 31)	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

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Usar a plataforma Go do

A plataforma AWS App Runner Go fornece tempos de execução gerenciados. Cada tempo de execução facilita a criação e a execução de contêineres com aplicativos web baseados em uma versão Go. Quando você usa um tempo de execução em Go, o App Runner começa com uma imagem gerenciada de tempo de execução em Go. Essa imagem é baseada na [imagem Docker do Amazon Linux](#) e contém o pacote de tempo de execução para uma versão do Go e algumas ferramentas. O App Runner usa essa imagem de tempo de execução gerenciada como imagem base e adiciona o código do aplicativo para criar uma imagem do Docker. Em seguida, ele implanta essa imagem para executar seu serviço web em um contêiner.

Você especifica um tempo de execução para seu serviço App Runner ao [criar um serviço](#) usando o console do App Runner ou a operação da [CreateService](#) API. Você também pode especificar um tempo de execução como parte do seu código-fonte. Use a `runtime` palavra-chave em um [arquivo de configuração do App Runner](#) que você inclui no seu repositório de código. A convenção de nomenclatura de um tempo de execução gerenciado é `<language-name><major-version>`.

Para nomes e versões válidos do tempo de execução do Go, consulte [the section called “Informações de lançamento”](#).

O App Runner atualiza o tempo de execução do seu serviço para a versão mais recente em cada implantação ou atualização de serviço. Se seu aplicativo exigir uma versão específica de um tempo de execução gerenciado, você poderá especificá-la usando a `runtime-version` palavra-chave no [arquivo de configuração do App Runner](#). Você pode bloquear qualquer nível de versão, incluindo uma versão principal ou secundária. O App Runner só faz atualizações de nível inferior no tempo de execução do seu serviço.

Sintaxe da versão para tempos de execução do Go: `major[.minor[.patch]]`

Por exemplo: `1.18.7`

Os exemplos a seguir demonstram o bloqueio de versões:

- 1.18— Bloqueie as versões principais e secundárias. O App Runner atualiza somente as versões de patch.
- 1.18.7— Bloqueie uma versão específica do patch. O App Runner não atualiza sua versão de tempo de execução.

Tópicos

- [Configuração de tempo de execução do Go](#)
- [Exemplos de tempo de execução do Go](#)
- [Informações sobre a versão do Go Runtime](#)

Configuração de tempo de execução do Go

Ao escolher um tempo de execução gerenciado, você também deve configurar, no mínimo, comandos de compilação e execução. Você os configura ao [criar](#) ou [atualizar](#) seu serviço App Runner. Você pode fazer isso usando um dos seguintes métodos:

- Usando o console do App Runner — Especifique os comandos na seção Configurar compilação do processo de criação ou da guia de configuração.
- Usando a API App Runner — chame a operação [CreateService](#) ou [UpdateService](#) da API. Especifique os comandos usando os `StartCommand` membros `BuildCommand` e do tipo de [CodeConfigurationValues](#) dados.
- Usando um [arquivo de configuração](#) — especifique um ou mais comandos de compilação em até três fases de compilação e um único comando de execução que serve para iniciar seu aplicativo. Há configurações opcionais adicionais.

Fornecer um arquivo de configuração é opcional. Ao criar um serviço App Runner usando o console ou a API, você especifica se o App Runner obtém suas configurações diretamente quando é criado ou de um arquivo de configuração.

Exemplos de tempo de execução do Go

Os exemplos a seguir mostram os arquivos de configuração do App Runner para criar e executar um serviço Go.

Arquivo de configuração Minimal Go

Este exemplo mostra um arquivo de configuração mínimo que você pode usar com um tempo de execução gerenciado em Go. Para as suposições que o App Runner faz com um arquivo de configuração mínimo, consulte [the section called “Exemplos de arquivos de configuração”](#)

Example apprunner.yaml

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

Arquivo de configuração Extended Go

Este exemplo mostra o uso de todas as chaves de configuração com um tempo de execução gerenciado em Go.

Note

A versão de tempo de execução usada nesses exemplos é **1.18.7**. Você pode substituí-lo por uma versão que você deseja usar. Para ver a versão mais recente do Go runtime compatível, consulte [the section called “Informações de lançamento”](#).

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"
```

Fonte completa do aplicativo Go

Esses exemplos mostram o código-fonte de um aplicativo Go completo que você pode implantar em um serviço de tempo de execução 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
```

Informações sobre a versão do Go Runtime

Este tópico lista os detalhes completos das versões de tempo de execução do Go compatíveis com o App Runner.

Versões de tempo de execução suportadas — versão original do App Runner

Nome do runtime	Versões secundárias	Pacotes incluídos
Go 1 (Go 1)	1.18.10	
	1.18.9	
	1.18.8	
	1.18.7	

Note

O App Runner fornece um processo de criação revisado para os principais tempos de execução específicos que foram lançados mais recentemente. Por isso, você verá referências à versão revisada do App Runner e à versão original do App Runner em determinadas seções deste documento. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Desenvolvendo código de aplicativo para o App Runner

Este capítulo discute as informações de tempo de execução e as diretrizes de desenvolvimento que você deve considerar ao desenvolver ou migrar o código do aplicativo para implantação em AWS App Runner.

Informações de tempo de execução

Se você fornecer uma imagem de contêiner ou se o App Runner criar uma para você, o App Runner executa o código do aplicativo em uma instância de contêiner. Aqui estão alguns aspectos principais do ambiente de tempo de execução da instância de contêiner.

- **Suporte de estrutura** — O App Runner oferece suporte a qualquer imagem que implemente um aplicativo web. É independente da linguagem de programação que você escolher e do servidor de aplicativos web ou estrutura que você usa, se você usar algum. Para sua conveniência, fornecemos tempos de execução gerenciados específicos da plataforma para várias plataformas de programação, a fim de agilizar o processo de criação de aplicativos e a criação de imagens abstratas.
- **Solicitações da Web** — o App Runner fornece suporte para HTTP 1.0 e HTTP 1.1 para as instâncias do contêiner. Para obter mais informações sobre como configurar seu serviço, consulte [the section called “Configuração”](#). Você não precisa implementar o tratamento do tráfego seguro HTTPS. O App Runner redireciona todas as solicitações HTTP recebidas para os endpoints HTTPS correspondentes. Você não precisa definir nenhuma configuração para habilitar o redirecionamento das solicitações HTTP da web. O App Runner encerra o TLS antes de passar solicitações para a instância do contêiner do aplicativo.

Note

- Há um limite total de tempo limite de solicitação de 120 segundos nas solicitações HTTP. Os 120 segundos incluem o tempo que o aplicativo leva para ler a solicitação, incluindo o corpo, e concluir a gravação da resposta HTTP.
- O limite de tempo limite de leitura e resposta da solicitação depende dos aplicativos que você usa. Esses aplicativos podem ter seus próprios tempos limite internos, como o servidor HTTP para Python, o Gunicorn, que tem um limite de tempo limite padrão de 30 segundos. Nesses casos, o limite de tempo limite do aplicativo substitui o limite de 120 segundos do App Runner.

- Você não precisa configurar conjuntos de criptografia TLS ou quaisquer outros parâmetros, pois o App Runner, sendo um serviço totalmente gerenciado, gerencia a terminação de TLS para você.
- Aplicativos sem estado — Atualmente, o App Runner não oferece suporte a um aplicativo com estado. Portanto, o App Runner não garante a persistência do estado além da duração do processamento de uma única solicitação da web recebida.
- Armazenamento — O App Runner aumenta ou diminui automaticamente as instâncias do seu aplicativo App Runner de acordo com o volume de tráfego de entrada. Você pode configurar as [opções de escalonamento automático](#) para seu aplicativo App Runner. Como o número de instâncias atualmente ativas que processam as solicitações da web é baseado no volume de tráfego de entrada, o App Runner não pode garantir que os arquivos possam persistir além do processamento de uma única solicitação. Portanto, o App Runner implementa o sistema de arquivos em sua instância de contêiner como armazenamento efêmero, o que significa que os arquivos são transitórios. Por exemplo, os arquivos não persistem quando você pausa e retoma o serviço App Runner.

O App Runner fornece 3 GB de armazenamento efêmero e usa uma parte dos 3 GB de armazenamento efêmero para sua imagem de contêiner extraída, compactada e não compactada na instância. O armazenamento efêmero restante pode ser usado pelo serviço App Runner. No entanto, este não é um armazenamento permanente devido à sua natureza apátrida.

Note

Pode haver cenários em que os arquivos de armazenamento persistam em todas as solicitações. Por exemplo, se a próxima solicitação chegar à mesma instância, os arquivos de armazenamento persistirão. A persistência dos arquivos de armazenamento nas solicitações pode ser útil em determinadas situações. Por exemplo, ao lidar com uma solicitação, você pode armazenar em cache os arquivos que seu aplicativo baixa se futuras solicitações precisarem deles. Isso pode acelerar o processamento futuro de solicitações, mas não pode garantir os ganhos de velocidade. Seu código não deve presumir que um arquivo que foi baixado em uma solicitação anterior ainda existe.

[Para garantir o armazenamento em cache usando um armazenamento de dados na memória de alta taxa de transferência e baixa latência, use um serviço como o Amazon ElastiCache](#)

- **Variáveis de ambiente** — Por padrão, o App Runner disponibiliza a variável de PORT ambiente em sua instância de contêiner. Você pode configurar o valor da variável com informações da porta e adicionar variáveis e valores de ambiente personalizados. Você também pode referenciar dados confidenciais armazenados no Parameter Store AWS Secrets Manager ou no AWS Systems Manager Parameter Store como variáveis de ambiente. Para obter mais informações sobre a criação de variáveis de ambiente, consulte [Variáveis de ambiente de referência](#).
- **Função da instância** — Se o código do seu aplicativo fizer chamadas para qualquer AWS serviço, usando o serviço APIs ou um deles AWS SDKs, crie uma função de instância usando AWS Identity and Access Management (IAM). Em seguida, anexe-o ao seu serviço App Runner ao criá-lo. Inclua todas as permissões de ação de AWS serviço que seu código exige em sua função de instância. Para obter mais informações, consulte [the section called “Função da instância”](#).

Diretrizes de desenvolvimento de código

Considere essas diretrizes ao desenvolver código para um aplicativo web do App Runner.

- **Correção de imagens de contêiner** — Ao fornecer imagens de contêiner, você é responsável por atualizar e corrigir regularmente essas imagens. Enquanto o App Runner gerencia a infraestrutura, você deve garantir a segurança e o up-to-date status das imagens de contêiner fornecidas. Para obter mais informações, consulte a documentação do [AWS App Runner](#).
- **Crie código sem estado** — Projete o aplicativo web que você implanta no serviço App Runner para ser sem estado. Seu código deve presumir que nenhum estado persiste além da duração do processamento de uma única solicitação da web recebida.
- **Excluir arquivos temporários** — Quando você cria arquivos, eles são armazenados em um sistema de arquivos e ocupam parte da alocação de armazenamento do seu serviço. Para evitar out-of-storage erros, não mantenha arquivos temporários por longos períodos. Equilibre o tamanho do armazenamento com a velocidade de tratamento de solicitações ao tomar decisões de armazenamento em cache de arquivos.
- **Inicialização da instância** — O App Runner fornece cinco minutos de tempo de inicialização da instância. Sua instância deve escutar as solicitações nas portas de escuta configuradas e estar íntegra dentro de cinco minutos após a inicialização. Durante o tempo de inicialização, as instâncias do App Runner recebem uma CPU virtual (vCPU) com base na configuração da sua vCPU. Para obter mais informações sobre a configuração de vCPU disponível, consulte [the section called “Configurações compatíveis com o App Runner”](#).

Depois que a instância é inicializada com sucesso, ela entra em um estado ocioso e aguarda as solicitações. Você paga com base na duração da inicialização da instância, com a cobrança mínima de um minuto por início da instância. Para obter mais informações sobre preços, consulte [Preços do AWS App Runner](#).

Usando o console do App Runner

Use o AWS App Runner console para criar, gerenciar e monitorar seus serviços do App Runner e recursos relacionados, como contas conectadas. Você pode visualizar os serviços existentes, criar novos e configurar um serviço. Você pode ver o status de um serviço do App Runner, bem como visualizar registros, monitorar atividades e monitorar métricas. Você também pode navegar até o site do seu serviço ou até o repositório de origem.

As seções a seguir descrevem o layout e a funcionalidade do console e indicam informações relacionadas.

Layout geral do console

O console do App Runner tem três áreas. Da esquerda para a direita:

- Painel de navegação — Um painel lateral que pode ser reduzido ou expandido. Use-o para escolher a página de console de nível superior que você deseja usar.
- Painel de conteúdo — A parte principal da página do console. Use-o para visualizar informações e realizar suas tarefas.
- Painel de ajuda — Um painel lateral para obter mais informações. Expanda-o para obter ajuda sobre a página em que você está. Ou escolha qualquer link de informações em uma página do console para obter ajuda contextual.

The screenshot shows the AWS App Runner console interface. On the left is a navigation sidebar with links for Services, Connected accounts, Network configuration, Auto scaling configuration, and Documentation. The main content area is titled 'Services' and shows a list of 'App Runner Services (2)'. A table lists the services with columns for Service name, Status, Default domain, Incoming traffic, Created, and Last activity. Two services are shown: 'pythonTest-bb-repo' and 'pythonTest', both with a 'Running' status. On the right, there is a 'Services' panel with a 'Create service' button and a list of instructions on how to manage services. Below this, there is a 'Learn more' section with links to 'Image-based service', 'Code-based service', 'Creating an App Runner service', 'Getting started', and 'Architecture and concepts'.

Service name	Status	Default domain	Incoming traffic	Created	Last activity
pythonTest-bb-repo	Running	https://vmijhqcpw.p...	Public	9/6/2023, 8:44:59 PM...	9/6/2023, 8:44:59 PM UTC
pythonTest	Running	https://jstcs2vdw.pu...	Public	9/6/2023, 8:30:17 PM...	9/6/2023, 8:30:17 PM UTC

A página de serviços

A página Serviços lista os serviços do App Runner em sua conta. Você pode reduzir o escopo da lista usando a caixa de texto do filtro.

Para acessar a página de Serviços

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
2. No painel de navegação, escolha Serviços.

Coisas que você pode fazer aqui:

- Crie um serviço App Runner. Para obter mais informações, consulte [the section called “Criação”](#).
- Escolha um nome de serviço para acessar a página do console do painel de serviços.
- Escolha um domínio de serviço para abrir a página do aplicativo web do serviço.

A página do painel de controle do serviço

Você pode visualizar informações sobre um serviço do App Runner e gerenciá-lo na página do painel do serviço. Na parte superior da página, você pode ver o nome do serviço.

Para acessar o painel de serviços, navegue até a página Serviços (consulte a seção anterior) e escolha seu serviço App Runner.

The screenshot displays the AWS App Runner console for a service named 'python-test'. The breadcrumb navigation shows 'App Runner > Services > python-test'. The service name 'python-test' is prominently displayed with an 'Info' link. To the right are buttons for 'Actions', a refresh icon, and a 'Deploy' button.

The 'Service overview' section contains the following details:

- Status:** Running (indicated by a green checkmark icon).
- Default domain:** <https://62wwc8evee.public.gamma.us-east-1.bullet.aws.dev>
- Service ARN:** `arn:aws:apprunner:us-east-1:123456789012:service/python-test/33f9aa7c44744fbc961e85014386b0d`
- Source:** https://github.com/your_account/python-hello/main

Below the overview is a horizontal menu with tabs: 'Logs', 'Activity' (selected), 'Metrics', 'Observability', 'Configuration', and 'Custom domains'.

The 'Activity (1)' section includes a search bar labeled 'Filter activities' and a table of recent activities:

Operation	Status	Started	Ended
Create service	✓ Succeeded	3/22/2022, 6:46:22 PM UTC	3/22/2022, 6:51:35 PM UTC

A seção Visão geral do serviço fornece detalhes básicos sobre o serviço App Runner e seu aplicativo. Coisas que você pode fazer aqui:

- Visualize detalhes do serviço, como status, integridade e ARN.
- Navegue até o domínio padrão — o domínio que o App Runner fornece para o aplicativo web em execução no seu serviço. Este é um subdomínio no domínio de propriedade do `awsapprunner.com` App Runner.
- Navegue até o repositório de origem implantado no serviço.
- Inicie a implantação de um repositório de origem em seu serviço.
- Pausa, retome e exclua seu serviço.

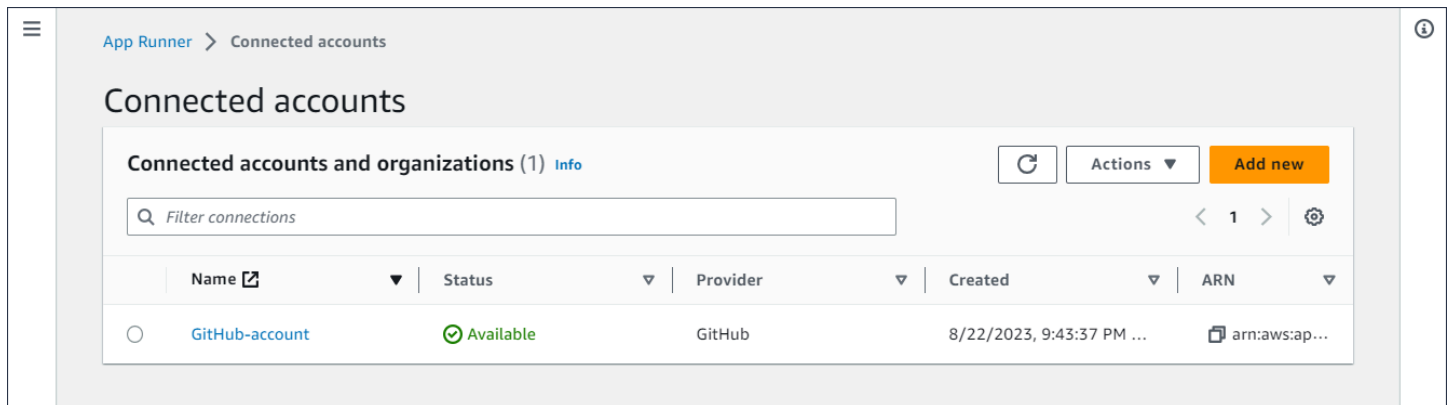
As guias abaixo da visão geral do serviço são para [gerenciamento](#) e [observabilidade](#) do serviço.

A página Contas conectadas

A página Contas conectadas lista as conexões do App Runner com os provedores de repositórios de código-fonte em sua conta. Você pode reduzir o escopo da lista usando a caixa de texto do filtro. Para obter mais informações sobre contas conectadas, consulte [the section called “Conexões”](#).

Para acessar a página Contas conectadas

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Contas conectadas.



Coisas que você pode fazer aqui:

- Veja uma lista de conexões de provedores de repositórios em sua conta. Para reduzir o escopo da lista, insira qualquer texto na caixa de texto do filtro.
- Escolha um nome de conexão para acessar a conta ou organização do provedor relacionado.
- Selecione uma conexão para concluir o handshake de uma conexão que você acabou de estabelecer (como parte da criação de um serviço) ou para excluir a conexão.

A página de configurações de Auto Scaling

A página Configurações do Auto Scaling lista as configurações do Auto Scaling que você configurou na sua conta. Você pode configurar alguns parâmetros para ajustar o comportamento do escalonamento automático e salvá-los em configurações diferentes que podem ser atribuídas posteriormente a um ou mais serviços do App Runner. Você pode reduzir o escopo da lista usando a caixa de texto do filtro. Para obter mais informações sobre configurações de escalonamento automático, consulte [Gerencie o escalonamento automático para um serviço](#)

Para acessar a página de configuração do Auto Scaling

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Configuração de escalonamento automático.

The screenshot displays the 'Auto scaling configuration' page in the AWS App Runner console. At the top, there's a breadcrumb 'App Runner > Auto scaling configuration'. Below it, the title 'Auto scaling configuration' is followed by a sub-header 'Auto scaling configurations (4) Info'. A description states: 'List of the available auto scaling configurations. Auto scaling configuration defines settings for instances used to process the web requests'. There are buttons for 'Refresh', 'Actions', and 'Create'. A search bar with the placeholder 'Filter configuration by name' is present. Below the search bar is a table with the following data:

	Configuration name ▲	Status ▼	Revisions ▼	Date created ▼	Date updated ▼
☐	DefaultConfiguration default	⊖ Not-in-use	1	5/18/2021, 12:00:00 AM UTC	–
☐	High-capacity	⊖ Not-in-use	2	9/7/2023, 10:21:03 PM UTC	9/7/2023, 11:24:13 PM UTC
☐	Low-capacity	✅ In-use	2	9/8/2023, 10:35:54 PM UTC	9/8/2023, 10:36:44 PM UTC
☐	Medium-capacity	✅ In-use	2	9/7/2023, 10:32:49 PM UTC	9/7/2023, 10:33:46 PM UTC

Coisas que você pode fazer aqui:

- Veja a lista de configurações de auto scaling existentes em sua conta.
- Crie uma nova configuração de auto scaling ou uma revisão para uma existente.
- Defina uma configuração de auto scaling como padrão para os novos serviços que você criar.
- Exclua uma configuração.
- Selecione o nome de uma configuração para navegar até o painel de revisões do Auto Scaling para [gerenciar](#) as revisões.

Gerenciando seu serviço App Runner

Este capítulo descreve como gerenciar seu AWS App Runner serviço. Neste capítulo, você aprende a gerenciar o ciclo de vida do seu serviço: criar, configurar e excluir um serviço, implantar novas versões do aplicativo em seu serviço e controlar a disponibilidade do seu serviço Web pausando e retomando seu serviço. Você também aprenderá a gerenciar outros aspectos do seu serviço, como conexões e escalonamento automático.

Tópicos

- [Criação de um serviço App Runner](#)
- [Reconstruindo um serviço do App Runner com falha](#)
- [Implantando uma nova versão do aplicativo no App Runner](#)
- [Configurando um serviço App Runner](#)
- [Gerenciando conexões do App Runner](#)
- [Gerenciando o escalonamento automático do App Runner](#)
- [Gerenciando nomes de domínio personalizados para um serviço App Runner](#)
- [Pausar e retomar um serviço do App Runner](#)
- [Excluindo um serviço do App Runner](#)

Criação de um serviço App Runner

AWS App Runner automatiza a transição de uma imagem de contêiner ou de um repositório de código-fonte para um serviço web em execução que se expande automaticamente. Você aponta o App Runner para sua imagem ou código de origem, especificando apenas um pequeno número de configurações necessárias. O App Runner cria seu aplicativo, se necessário, provisiona recursos computacionais e implanta seu aplicativo para ser executado neles.

Quando você cria um serviço, o App Runner cria um recurso de serviço. Em alguns casos, talvez seja necessário fornecer um recurso de conexão. Se você usa o console do App Runner, o console cria implicitamente o recurso de conexão. Para obter mais informações sobre os tipos de recursos do App Runner, consulte [the section called “Recursos do App Runner”](#). Esses tipos de recursos têm cotas associadas à sua conta em cada Região da AWS. Para obter mais informações, consulte [the section called “Cotas de recursos do App Runner”](#).

Há diferenças sutis no procedimento de criação de um serviço, dependendo do tipo de fonte e do provedor. Este tópico aborda diferentes procedimentos para criar esses tipos de fonte para que você possa seguir o que for adequado à sua situação. Para iniciar um procedimento básico com um exemplo de código, consulte [Conceitos básicos](#).

Pré-requisitos

Antes de criar seu serviço App Runner, certifique-se de concluir as seguintes ações:

- Conclua as etapas de configuração em [Configuração](#).
- Certifique-se de que a fonte do aplicativo esteja pronta. Você pode usar um repositório de código no [GitHub](#) ou [Bitbucket](#) ou uma imagem de contêiner no Amazon [Elastic Container Registry \(Amazon ECR\)](#) para criar um serviço App Runner.

Criar um serviço

Esta seção mostra o processo de criação dos dois tipos de serviço do App Runner: com base no código-fonte e com base em uma imagem de contêiner.

Note

Se você criar um conector VPC de tráfego de saída para um serviço, o processo de inicialização do serviço a seguir terá uma latência única. Você pode definir essa configuração para um novo serviço ao criá-lo, ou posteriormente, com uma atualização de serviço. Para obter mais informações, consulte [Latência única](#) o capítulo Networking with App Runner deste guia.

Crie um serviço a partir de um repositório de código

[As seções a seguir mostram como criar um serviço App Runner quando sua fonte é um repositório de código no Bitbucket, GitHub ou no Bitbucket](#). Quando você usa um repositório de código, o App Runner precisa se conectar à organização ou conta do provedor. Portanto, você precisa ajudar a estabelecer essa conexão. Para obter mais informações sobre as conexões do App Runner, consulte [the section called “Conexões”](#).

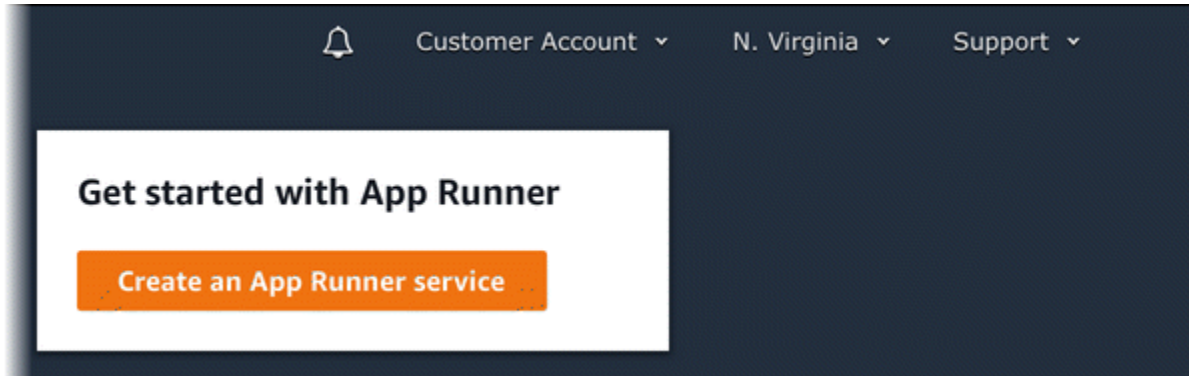
Quando você cria o serviço, o App Runner cria uma imagem do Docker que contém o código e as dependências do seu aplicativo. Em seguida, ele inicia um serviço que executa uma instância de contêiner dessa imagem.

Criação de um serviço a partir do código usando o console do App Runner

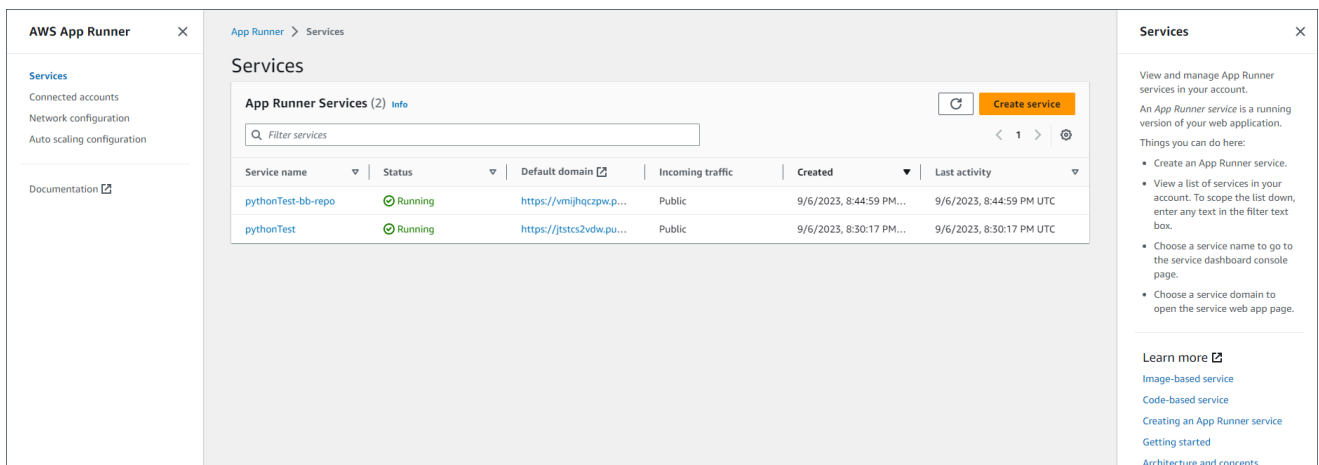
Para criar um serviço App Runner usando o console

1. Configure seu código-fonte.

- Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
- Se Conta da AWS ainda não tiver nenhum serviço do App Runner, a página inicial do console será exibida. Escolha Criar um serviço App Runner.



Se tiver serviços existentes, a página Serviços com uma lista de seus serviços será exibida. Conta da AWS Escolha Create service.



- Na página Fonte e implantação, na seção Fonte, em Tipo de repositório, escolha Repositório de código-fonte.
- Selecione um tipo de provedor. Escolha um GitHub ou o Bitbucket.
- Em seguida, selecione uma conta ou organização para o provedor que você já usou antes ou escolha Adicionar novo. Em seguida, passe pelo processo de fornecer suas credenciais de repositório de código e escolher uma conta ou organização à qual se conectar.

- f. Em Repositório, selecione o repositório que contém o código do seu aplicativo.
- g. Em Branch, selecione a ramificação que você deseja implantar.
- h. Em Diretório de origem, insira o diretório no repositório de origem que armazena o código do aplicativo e os arquivos de configuração.

 Note

Os comandos build e start são executados no diretório de origem que você especificar. O App Runner trata o caminho como absoluto a partir da raiz. Se você não especificar um valor aqui, o diretório usará como padrão a raiz do repositório.

2. Configure suas implantações.

- a. Na seção Configurações de implantação, escolha Manual ou Automático.

Para obter mais informações sobre métodos de implantação, consulte [the section called “Métodos de implantação”](#).

- b. Escolha Próximo.

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.

3. Configure a compilação do aplicativo.

- a. Na página Configurar compilação, em Arquivo de configuração, escolha Definir todas as configurações aqui se o seu repositório não contiver um arquivo de configuração do App Runner ou Usar um arquivo de configuração se tiver.

 Note

Um arquivo de configuração do App Runner é uma forma de manter sua configuração de compilação como parte da fonte do aplicativo. Quando você fornece um, o App Runner lê alguns valores do arquivo e não permite que você os defina no console.

- b. Forneça as seguintes configurações de compilação:
 - Tempo de execução — Escolha um tempo de execução gerenciado específico para seu aplicativo.
 - Comando de compilação — insira um comando que compila seu aplicativo a partir do código-fonte. Isso pode ser uma ferramenta específica do idioma ou um script fornecido com seu código.
 - Comando Iniciar — Insira o comando que inicia seu serviço web.
 - Porta — Insira a porta IP que seu serviço web escuta.
- c. Escolha Próximo.

Configure build Info

Build settings

Configuration file

☒ **Configure all settings here**
Specify all settings for your service here in the App Runner console.

☐ **Use a configuration file**
Let App Runner read your configuration from the `apprunner.yaml` file in your source repository.

Runtime
Choose an App Runner runtime for your service.

Python 3 ▼

Build command
This command runs in the root directory of your repository when a new code version is deployed. Use it to install dependencies or compile your code.

`pip install -r requirements.txt`

Start command
This command runs in the root directory of your service to start the service processes. Use it to start a webserver for your service. The command can access environment variables that App Runner and you defined.

`python server.py`

Port
Your service uses this IP port.

8080 ▼

Cancel Previous **Next**

4. Configure seu serviço.

- Na página Configurar serviço, na seção Configurações do serviço, insira um nome de serviço.

Note

Todas as outras configurações de serviço são opcionais ou têm padrões fornecidos pelo console.

- b. Opcionalmente, altere ou adicione outras configurações para atender aos requisitos do seu aplicativo.
- c. Escolha Próximo.

Configure service [Info](#)

Service settings

Service name

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

Virtual CPU & memory

Environment variables — optional
Key-value pairs that you can use to store custom configuration values.
No environment variables have been configured.

[Add environment variable](#)

► **Additional configuration**

► **Auto scaling** [Info](#)
Configure automatic scaling behavior.

► **Health check** [Info](#)
Configure load balancer health checks.

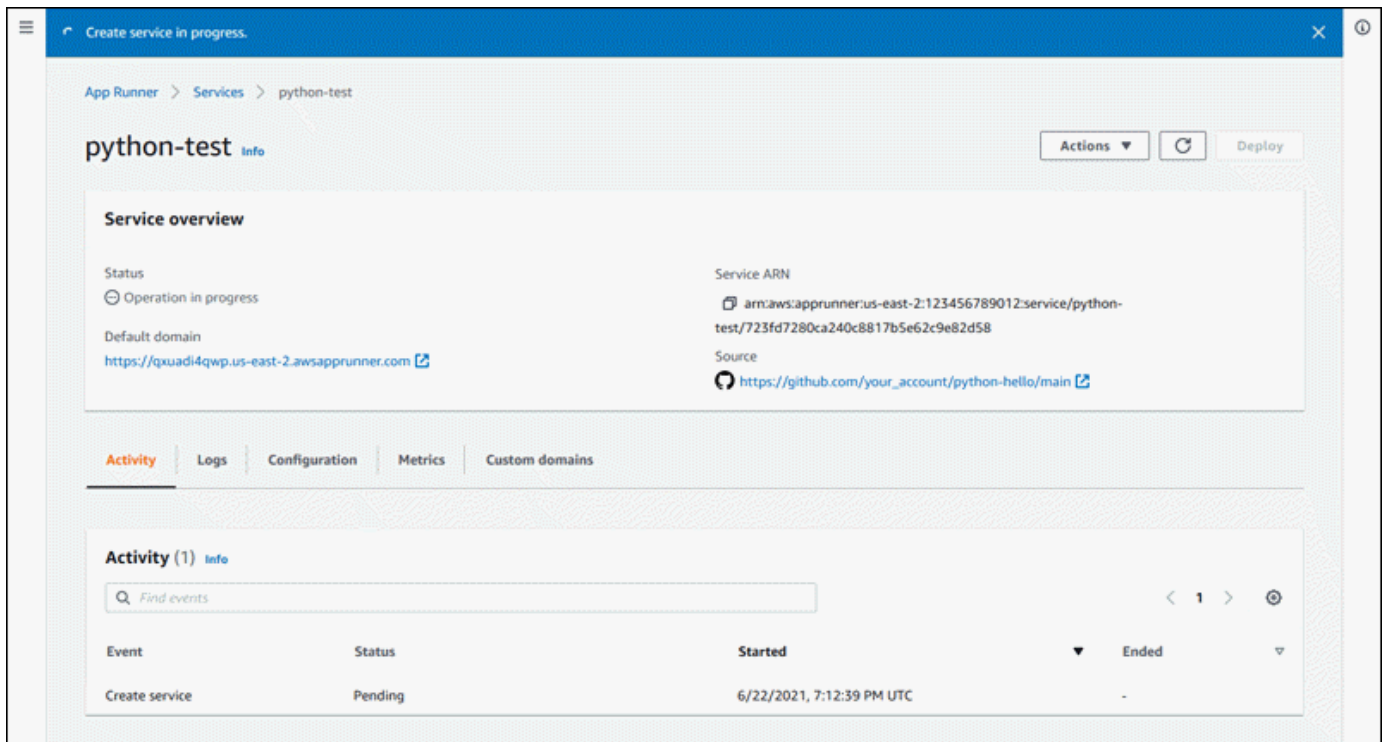
► **Security** [Info](#)
Specify an Instance role and an AWS KMS encryption key

► **Tags** [Info](#)
Use tags to search and filter your resources, track your AWS costs, and control access permissions.

[Cancel](#) [Previous](#) [Next](#)

5. Na página Revisar e criar, verifique todos os detalhes inseridos e escolha Criar e implantar.

Resultado: se o serviço for criado com êxito, o console exibirá o painel de serviços com uma visão geral do novo serviço.



6. Verifique se o serviço está em execução.

- Na página do painel do serviço, aguarde até que o status do serviço esteja em execução.
- Escolha o valor do domínio padrão. É a URL do site do seu serviço.
- Use seu site e verifique se ele está funcionando corretamente.

Criar um serviço a partir do código usando a API App Runner ou AWS CLI

Para criar um serviço usando a API App Runner ou AWS CLI chame a ação da `CreateService` API. Para obter mais informações e um exemplo, consulte [CreateService](#). Se for a primeira vez que você está criando um serviço usando uma organização ou conta específica para um repositório de código-fonte (GitHub ou Bitbucket), comece ligando. [CreateConnection](#) Isso estabelece uma conexão entre o App Runner e a organização ou conta do provedor do repositório. Para obter mais informações sobre as conexões do App Runner, consulte [the section called "Conexões"](#).

Se a chamada retornar uma resposta bem-sucedida com a exibição [de um objeto Service](#) "Status": "CREATING", seu serviço começará a ser criado.

Para ver um exemplo de chamada, consulte [Criar um serviço de repositório de código-fonte](#) na Referência da AWS App Runner API

Crie um serviço a partir de uma imagem do Amazon ECR

As seções a seguir mostram como criar um serviço App Runner quando sua fonte é uma imagem de contêiner armazenada no [Amazon ECR](#). O Amazon ECR é um AWS service (Serviço da AWS). Portanto, para criar um serviço baseado em uma imagem do Amazon ECR, você fornece ao App Runner uma função de acesso contendo as permissões de ação do Amazon ECR necessárias.

Note

As imagens armazenadas no Amazon ECR Public estão disponíveis publicamente. Portanto, se sua imagem estiver armazenada no Amazon ECR Public, não é necessária uma função de acesso.

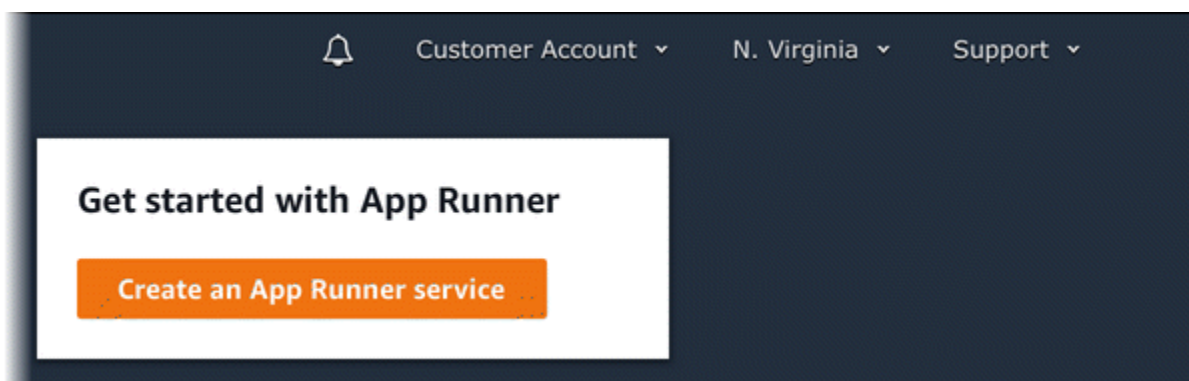
Quando seu serviço está sendo criado, o App Runner inicia um serviço que executa uma instância de contêiner da imagem que você fornece. Não há fase de construção neste caso.

Para obter mais informações, consulte [Serviço baseado em imagem](#).

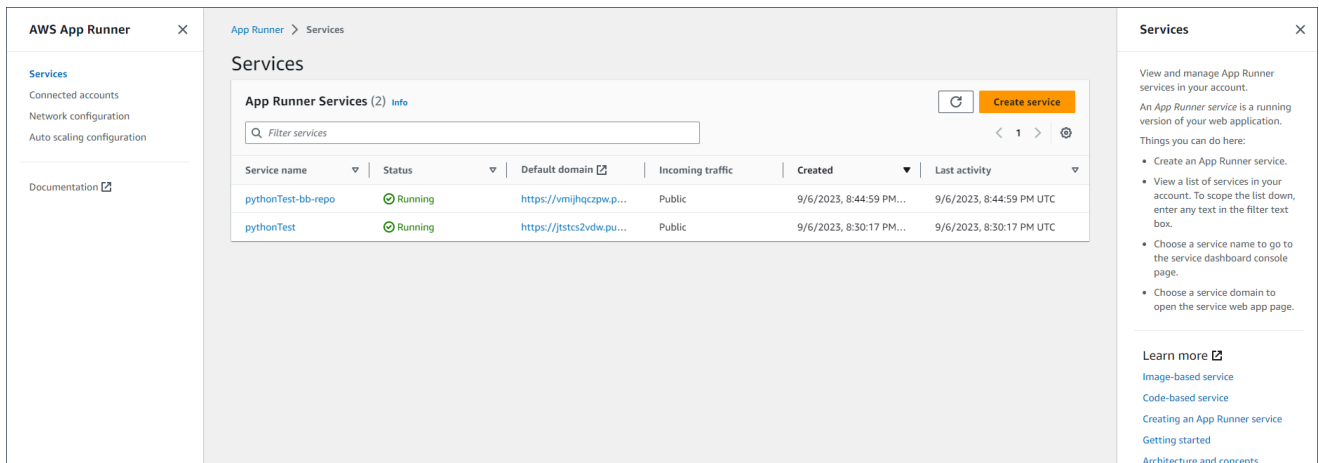
Criação de um serviço a partir de uma imagem usando o console do App Runner

Para criar um serviço App Runner usando o console

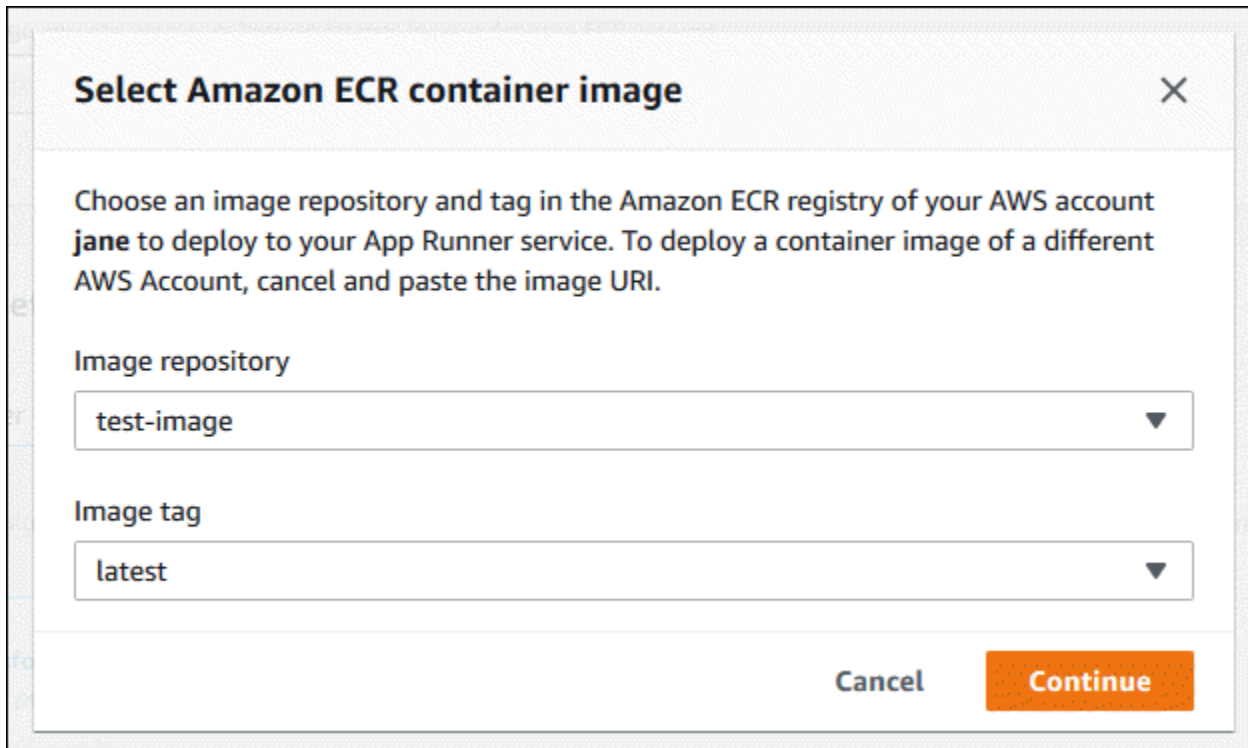
1. Configure seu código-fonte.
 - a. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
 - b. Se Conta da AWS ainda não tiver nenhum serviço do App Runner, a página inicial do console será exibida. Escolha Criar um serviço App Runner.



Se tiver serviços existentes, a página Serviços com uma lista de seus serviços será exibida. Conta da AWS Escolha Create service.



- c. Na página Origem e implantação, na seção Fonte, em Tipo de repositório, escolha Registro de contêiner.
- d. Para Provedor, escolha o provedor em que sua imagem está armazenada:
 - Amazon ECR — Uma imagem privada armazenada no Amazon ECR.
 - Amazon ECR Public — Uma imagem publicamente legível que é armazenada no Amazon ECR Public.
- e. Em URI da imagem do contêiner, escolha Procurar.
- f. Na caixa de diálogo Selecionar imagem de contêiner do Amazon ECR, em Repositório de imagens, selecione o repositório que contém sua imagem.
- g. Em Tag de imagem, selecione a tag de imagem específica que você deseja implantar (por exemplo, a mais recente) e escolha Continuar.



Select Amazon ECR container image

Choose an image repository and tag in the Amazon ECR registry of your AWS account **jane** to deploy to your App Runner service. To deploy a container image of a different AWS Account, cancel and paste the image URI.

Image repository

test-image ▼

Image tag

latest ▼

Cancel Continue

2. Configure suas implantações.

- a. Na seção Configurações de implantação, escolha Manual ou Automático.

Note

O App Runner não oferece suporte à implantação automática para imagens públicas do Amazon ECR e para imagens em um repositório do Amazon ECR que pertença a uma AWS conta diferente daquela em que seu serviço está.

Para obter mais informações sobre métodos de implantação, consulte [the section called “Métodos de implantação”](#).

- b. [Provedor Amazon ECR] Para a função de acesso ao ECR, escolha uma função de serviço existente em sua conta ou escolha criar uma nova função. Se você estiver usando a implantação manual, também poderá optar por usar a função de usuário do IAM no momento da implantação.
- c. Escolha Próximo.

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.

[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.

[Cancel](#)[Next](#)

3. Configure seu serviço.

- a. Na página Configurar serviço, na seção Configurações do serviço, insira o nome do serviço e a porta IP que o site do serviço escuta.

 Note

Todas as outras configurações de serviço são opcionais ou têm padrões fornecidos pelo console.

- b. (Opcional) Altere ou adicione outras configurações para atender às necessidades do seu aplicativo.
- c. Escolha Próximo.

Configure service [Info](#)

Service settings

Service name

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

Virtual CPU & memory

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.

► 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

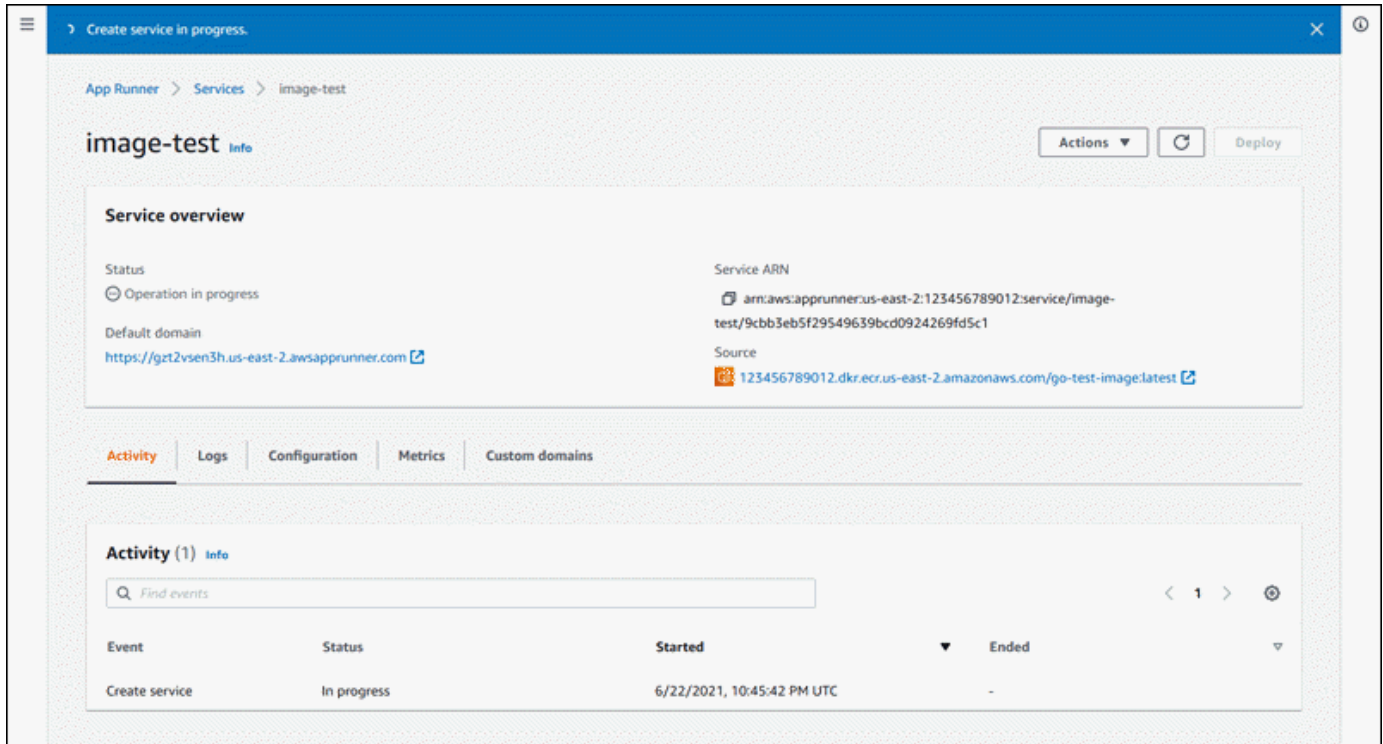
► Tags [Info](#)

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

[Cancel](#)[Previous](#)[Next](#)

4. Na página Revisar e criar, verifique todos os detalhes que você inseriu e escolha Criar e implantar.

Resultado: se o serviço for criado com êxito, o console mostrará o painel do serviço, com uma visão geral do novo serviço.



5. Verifique se o serviço está em execução.
 - a. Na página do painel do serviço, aguarde até que o status do serviço esteja em execução.
 - b. Escolha o valor do domínio padrão. É a URL do site do seu serviço.
 - c. Use seu site e verifique se ele está funcionando corretamente.

Criar um serviço a partir de uma imagem usando a API App Runner ou AWS CLI

Para criar um serviço usando a API App Runner ou AWS CLI chame a ação da [CreateService](#) API.

A criação do serviço começa se a chamada retornar uma resposta bem-sucedida com a exibição [de um objeto Service](#) "Status": "CREATING".

Para ver um exemplo de chamada, consulte [Criar um serviço de repositório de imagens de origem](#) na Referência da AWS App Runner API

Reconstruindo um serviço do App Runner com falha

Se você receber um erro de Falha na criação ao criar um serviço do App Runner, você pode fazer o seguinte.

- Siga as etapas [the section called “Falha na criação do serviço”](#) para identificar a causa do erro.
- Se você encontrar um erro na fonte ou na configuração, faça as alterações necessárias e reconstrua seu serviço.
- Se um problema temporário com o App Runner fez com que seu serviço falhasse, reconstrua o serviço com falha sem fazer nenhuma alteração na fonte ou na configuração.

Você pode reconstruir seu serviço com falha por meio do [console do App Runner](#) ou da API do [App Runner](#) ou. AWS CLI

Reconstruindo um serviço do App Runner com falha usando o console do App Runner

Rebuild with updates

A criação de um serviço pode falhar por vários motivos. Quando isso acontece, é importante identificar e corrigir a causa raiz do problema antes de reconstruir seu serviço. Para obter mais informações, consulte [the section called “Falha na criação do serviço”](#).

Para reconstruir um serviço com falhas com atualizações

1. Vá até a guia Configurações na sua página de serviço e escolha Editar.

A página abre um painel de resumo que exibe uma lista de todas as suas atualizações.

2. Faça as alterações necessárias e revise-as no painel de resumo.
3. Escolha Salvar e reconstruir.

Você pode monitorar o progresso na guia Registros da sua página de serviço.

Rebuild without updates

Se um problema temporário fizer com que a criação do serviço falhe, você poderá reconstruí-lo sem modificar a fonte ou as configurações.

Para reconstruir um serviço com falha sem atualizações

- Escolha Reconstruir no canto superior direito da sua página de serviço.

Você pode monitorar o progresso na guia Registros da sua página de serviço.

- Se o serviço não for criado novamente, siga as instruções de solução de problemas em [the section called “Falha na criação do serviço”](#). Faça as alterações necessárias e, em seguida, reconstrua seu serviço.

Reconstrução do serviço App Runner com falha usando a API App Runner ou AWS CLI

Rebuild with updates

Para reconstruir um serviço com falha:

1. Siga as instruções em [the section called “Falha na criação do serviço”](#) para encontrar a causa do erro.
2. Faça as alterações necessárias na ramificação ou na imagem do repositório de origem ou na configuração que causou o erro.
3. Reconstrua chamando a ação da [UpdateService](#) API com os novos parâmetros do repositório de código-fonte ou do repositório de imagens de origem. O App Runner recupera a confirmação mais recente do repositório do código-fonte.

Example Reconstruindo com atualizações

No exemplo a seguir, a configuração de origem de um serviço baseado em imagem está sendo atualizada. O valor do Port é alterado para 80.

Atualização do `input.json` arquivo para o serviço App Runner baseado em imagem

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



```
}  
  }  
}   
}
```

Chamando a ação `UpdateService` da API.

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

Rebuild without updates

Para reconstruir seu serviço com falha usando a API App Runner ou AWS CLI chame a ação da [UpdateService](#) API sem fazer nenhuma alteração na fonte ou na configuração do seu serviço. Escolha reconstruir sem fazer atualizações somente se a criação do serviço falhar devido a um problema temporário com o App Runner.

Implantando uma nova versão do aplicativo no App Runner

Ao [criar um serviço](#) em AWS App Runner, você configura uma fonte de aplicativo — uma imagem de contêiner ou um repositório de origem. O App Runner provisiona recursos para executar seu serviço e implanta seu aplicativo neles.

Este tópico descreve maneiras de reimplantar a fonte do aplicativo no serviço App Runner quando uma nova versão estiver disponível. Isso pode ser uma nova versão de imagem no repositório de imagens ou um novo commit no repositório de código. O App Runner fornece dois métodos de implantação em um serviço: automático e manual.

Métodos de implantação

O App Runner fornece os seguintes métodos para você controlar como as implantações de aplicativos são iniciadas.

Implantação automática

Use a implantação automática quando quiser um comportamento de integração e implantação contínuas (CI/CD) para seu serviço. O App Runner monitora alterações em seu repositório de imagens ou códigos.

Repositório de imagens — Sempre que você envia uma nova versão de imagem para seu repositório de imagens ou uma nova confirmação para seu repositório de código, o App Runner a implanta automaticamente em seu serviço sem nenhuma ação adicional de sua parte.

Repositório de código — Sempre que você envia um novo commit para seu repositório de código que faz alterações no [diretório de origem](#), o App Runner implanta todo o repositório. Como somente as alterações no diretório de origem acionam uma implantação automática, é importante entender como a localização do diretório de origem afeta o escopo de uma implantação automatizada.

- **Diretório de nível superior (raiz do repositório)** — Esse é o valor padrão definido para o diretório de origem quando você cria um serviço. Se seu diretório de origem estiver definido com esse valor, isso significa que todo o repositório está dentro do diretório de origem. Portanto, todos os commits enviados para o repositório de origem acionarão uma implantação nesse caso.
- **Qualquer caminho de diretório que não seja a raiz do repositório (não padrão)** — Como somente as alterações enviadas ao diretório de origem acionarão uma implantação automática, quaisquer alterações enviadas ao seu repositório que não estejam no diretório de origem não acionarão uma implantação automática. Portanto, você deve usar uma implantação manual para implantar as alterações que você envia para fora do diretório de origem.

 Note

O App Runner não oferece suporte à implantação automática para imagens públicas do Amazon ECR e para imagens em um repositório do Amazon ECR que pertença a uma AWS conta diferente daquela em que seu serviço está.

Implantação manual

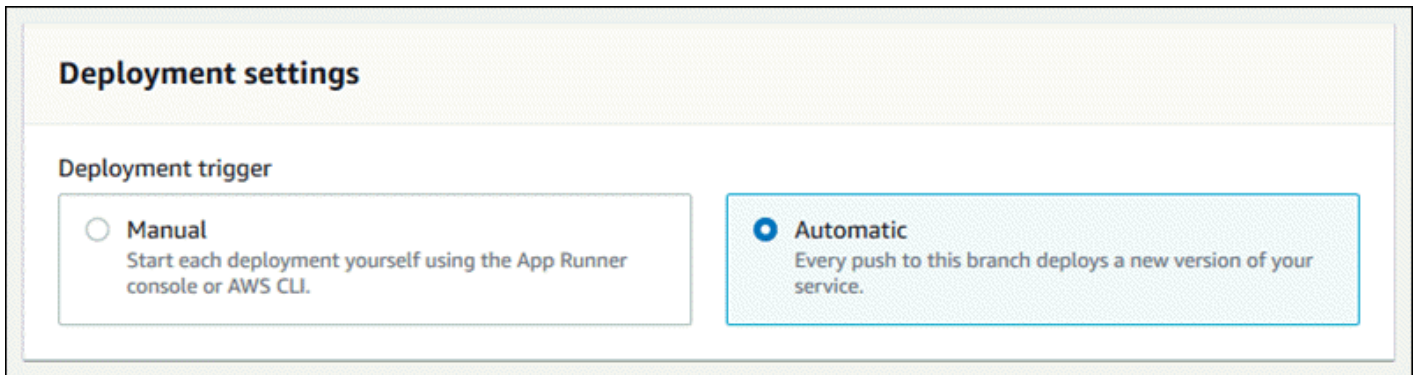
Use a implantação manual quando quiser iniciar explicitamente cada implantação em seu serviço. Você inicia uma implantação se o repositório que você configurou para seu serviço tiver uma nova versão que você deseja implantar. Para obter mais informações, consulte [the section called “Implantação manual”](#).

 Note

Quando você executa uma implantação manual, o App Runner implanta a fonte do repositório completo.

Você pode configurar o método de implantação do seu serviço das seguintes maneiras:

- **Console** — Para um novo serviço que você está criando ou para um serviço existente, na seção Configurações de implantação da página de configuração de origem e implantação, escolha Manual ou Automático.



- **API ou AWS CLI** — Em uma chamada para a [atualização de serviço](#) `CreateService` ou, defina o `AutoDeploymentsEnabled` membro do `SourceConfiguration` parâmetro como `False` para implantação manual ou `True` automática.

Comparando implantações automáticas e manuais

As implantações automáticas e manuais produzem o mesmo resultado: os dois métodos implantam o repositório completo.

A diferença entre os dois métodos é o mecanismo de acionamento:

- As implantações manuais são acionadas por uma implantação do console, uma chamada para a AWS CLI ou uma chamada para a API App Runner. A [Implantação manual](#) seção a seguir fornece os procedimentos para isso.
- As implantações automáticas são acionadas por uma alteração no conteúdo do [diretório de origem](#).

Implantação manual

Com a implantação manual, você precisa iniciar explicitamente cada implantação em seu serviço. Quando você tiver uma nova versão da imagem ou do código do seu aplicativo pronta para ser

implantada, consulte as seções a seguir para saber como realizar uma implantação usando o console e a API.

Note

Quando você executa uma implantação manual, o App Runner implanta a fonte do repositório completo.

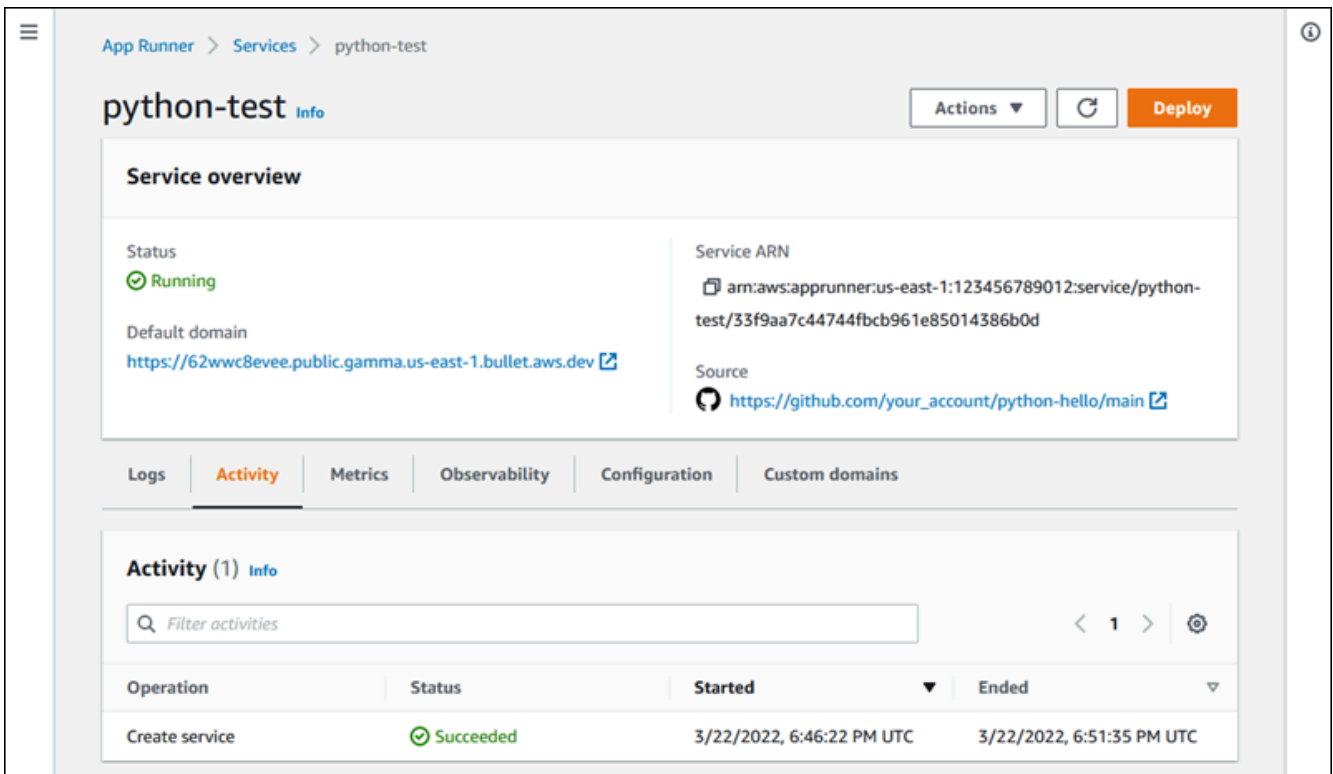
Implante uma versão do seu aplicativo usando um dos seguintes métodos:

App Runner console

Para implantar usando o console do App Runner

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Escolha Implantar.

Resultado: a implantação da nova versão é iniciada. Na página do painel do serviço, o status do serviço muda para Operação em andamento.

4. Aguarde o término da implantação. Na página do painel do serviço, o status do serviço deve voltar para Em execução.
5. Para verificar se a implantação foi bem-sucedida, na página do painel do serviço, escolha o valor do domínio padrão — é a URL do site do seu serviço. Inspecione ou interaja com seu aplicativo web e verifique sua alteração de versão.

 Note

[Para aumentar a segurança de seus aplicativos App Runner, o domínio*.awsapprunner.com é registrado na Lista Pública de Sufixos \(PSL\).](#)

Para maior segurança, recomendamos que você use cookies com um __Host - prefixo se precisar definir cookies confidenciais no nome de domínio padrão para seus aplicativos App Runner. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

App Runner API or AWS CLI

Para implantar usando a API App Runner ou AWS CLI, chame a ação da [StartDeploymentAPI](#). O único parâmetro a ser passado é o ARN do serviço. Você já configurou o local de origem do aplicativo quando criou o serviço, e o App Runner pode encontrar a nova versão. Sua implantação será iniciada se a chamada retornar uma resposta bem-sucedida.

Configurando um serviço App Runner

Ao [criar um AWS App Runner serviço](#), você define vários valores de configuração. Você pode alterar algumas dessas configurações depois de criar o serviço. Outras configurações podem ser aplicadas somente durante a criação do serviço e não podem ser alteradas posteriormente. Este tópico discute a configuração do seu serviço usando a API App Runner, o console do App Runner e um arquivo de configuração do App Runner.

Tópicos

- [Configure seu serviço usando a API App Runner ou AWS CLI](#)

- [Configure seu serviço usando o console do App Runner](#)
- [Configure seu serviço usando um arquivo de configuração do App Runner](#)
- [Configurando a observabilidade para seu serviço](#)
- [Definindo as configurações do serviço usando recursos compartilháveis](#)
- [Configurando verificações de saúde para seu serviço](#)

Configure seu serviço usando a API App Runner ou AWS CLI

A API define quais configurações podem ser alteradas após a criação do serviço. A lista a seguir discute as ações, os tipos e as limitações relevantes.

- [UpdateService](#)ação — Pode ser chamada após a criação para atualizar algumas configurações.
 - Pode ser atualizado — Você pode atualizar as configurações nos `HealthCheckConfiguration` parâmetros `SourceConfigurationInstanceConfiguration`, e. No entanto `SourceConfiguration`, em, você não pode mudar seu tipo de fonte de código para imagem ou vice-versa. Você deve fornecer o mesmo parâmetro de repositório fornecido ao criar o serviço. É um `CodeRepository` ou outro `ImageRepository`.

Você também pode atualizar os seguintes recursos ARNs de configuração separados associados ao serviço:

- `AutoScalingConfigurationArn`
- `VpcConnectorArn`
- Não pode ser atualizado — Você não pode alterar os `EncryptionConfiguration` parâmetros `ServiceName` e que estão disponíveis na [CreateService](#)ação. Eles não podem ser alterados após serem criados. A [UpdateService](#)ação não inclui esses parâmetros.
- API versus arquivo — Você pode definir o `ConfigurationSource` parâmetro do [CodeConfiguration](#) tipo (usado para repositórios de código-fonte como parte de `SourceConfiguration`) como. `Repository` Nesse caso, o App Runner ignora as configurações e as lê em `CodeConfigurationValues` um [arquivo de configuração](#) no seu repositório. Se você `ConfigurationSource` definir como `API`, o App Runner obtém todas as configurações da chamada da API e ignora o arquivo de configuração, mesmo que exista um.
- [TagResource](#)ação — Pode ser chamada após a criação do serviço para adicionar tags ao serviço ou atualizar valores de tags existentes.

- [UntagResource](#)ação — Pode ser chamada após a criação do serviço para remover as tags do serviço.

 Note

Se você criar um conector VPC de tráfego de saída para um serviço, o processo de inicialização do serviço a seguir terá uma latência única. Você pode definir essa configuração para um novo serviço ao criá-lo, ou posteriormente, com uma atualização de serviço. Para obter mais informações, consulte [Latência única](#) o capítulo Networking with App Runner deste guia.

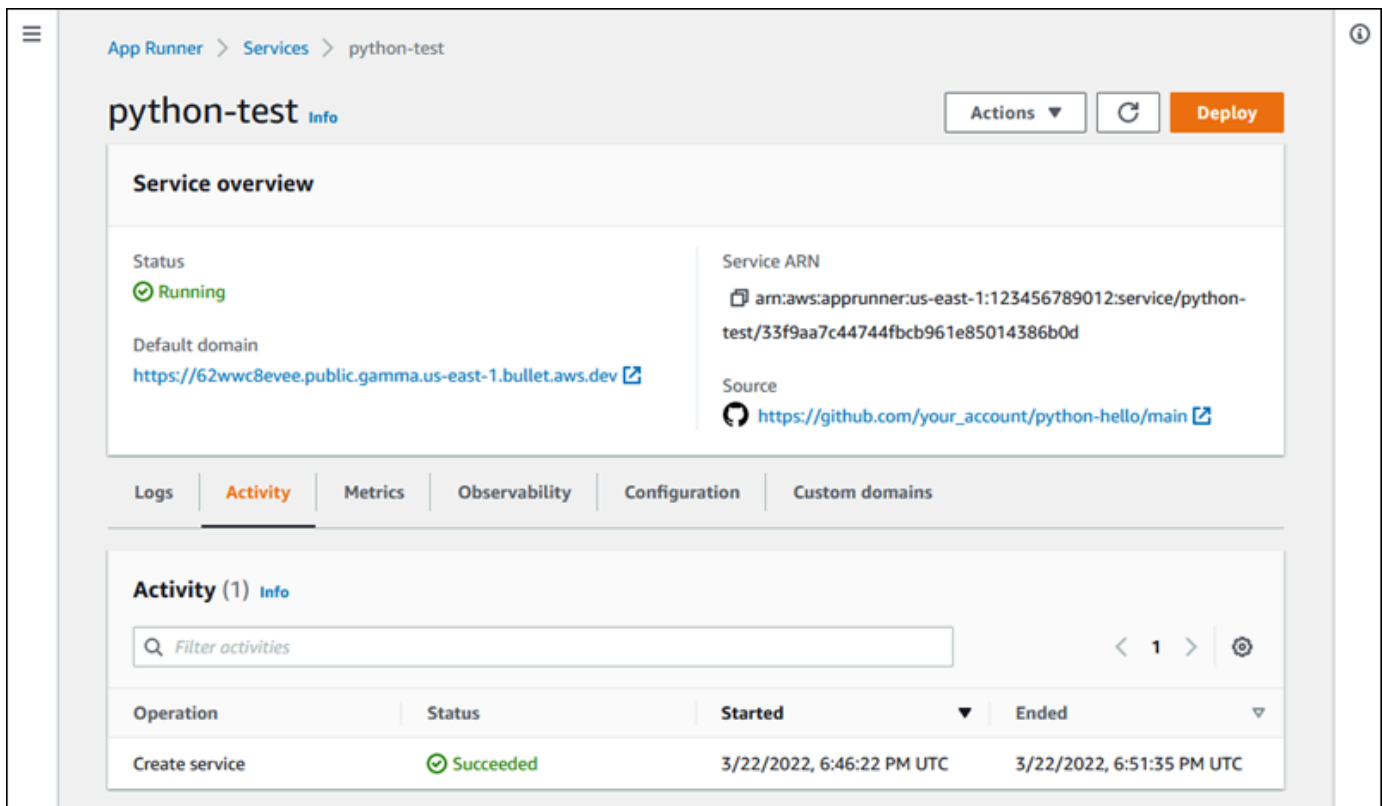
Configure seu serviço usando o console do App Runner

O console usa a API App Runner para aplicar atualizações de configuração. As regras de atualização impostas pela API, conforme definido na seção anterior, determinam o que você pode configurar usando o console. Algumas configurações que estavam disponíveis durante a criação do serviço não estão disponíveis para modificação posterior. Além disso, se você decidir usar um [arquivo de configuração](#), as configurações adicionais ficarão ocultas no console e o App Runner as lerá a partir do arquivo.

Para configurar seu serviço

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Na página do painel do serviço, escolha a guia Configuração.

Resultado: o console exibe as configurações atuais do seu serviço em várias seções: Origem e implantação, Configurar compilação e Configurar serviço.

4. Para atualizar as configurações em qualquer categoria, escolha Editar.
5. Na página de edição de configuração, faça as alterações desejadas e escolha Salvar alterações.

Note

Se você criar um conector VPC de tráfego de saída para um serviço, o processo de inicialização do serviço a seguir terá uma latência única. Você pode definir essa configuração para um novo serviço ao criá-lo, ou posteriormente, com uma atualização de serviço. Para obter mais informações, consulte [Latência única](#) o capítulo Networking with App Runner deste guia.

Configure seu serviço usando um arquivo de configuração do App Runner

Ao criar ou atualizar um serviço do App Runner, você pode instruir o App Runner a ler algumas definições de configuração de um arquivo de configuração que você fornece como parte do seu repositório de origem. Ao fazer isso, você pode gerenciar as configurações relacionadas ao seu código-fonte sob o controle do código-fonte, junto com o próprio código. O arquivo de configuração também fornece algumas configurações avançadas que você não pode definir usando o console ou a API. Para obter mais informações, consulte [Arquivo de configuração do App Runner](#).

Note

Se você criar um conector VPC de tráfego de saída para um serviço, o processo de inicialização do serviço a seguir terá uma latência única. Você pode definir essa configuração para um novo serviço ao criá-lo, ou posteriormente, com uma atualização de serviço. Para obter mais informações, consulte [Latência única](#) o capítulo Networking with App Runner deste guia.

Configurando a observabilidade para seu serviço

AWS App Runner se integra a vários AWS serviços para fornecer um amplo conjunto de ferramentas de observabilidade para seu serviço App Runner. Para obter mais informações, consulte [Observabilidade](#).

O App Runner permite ativar alguns recursos de observabilidade e configurar seu comportamento usando um recurso compartilhável chamado `ObservabilityConfiguration`. Você pode fornecer um recurso de configuração de observabilidade ao criar ou atualizar um serviço. O console do App Runner cria um para você quando você cria um novo serviço do App Runner. Fornecer uma configuração de observabilidade é opcional. Se você não fornecer uma, o App Runner fornecerá uma configuração de observabilidade padrão.

Você pode compartilhar uma única configuração de observabilidade em vários serviços do App Runner para garantir que eles tenham o mesmo comportamento de observabilidade. Para obter mais informações, consulte [the section called “Recursos de configuração”](#).

Você pode configurar os seguintes recursos de observabilidade usando configurações de observabilidade:

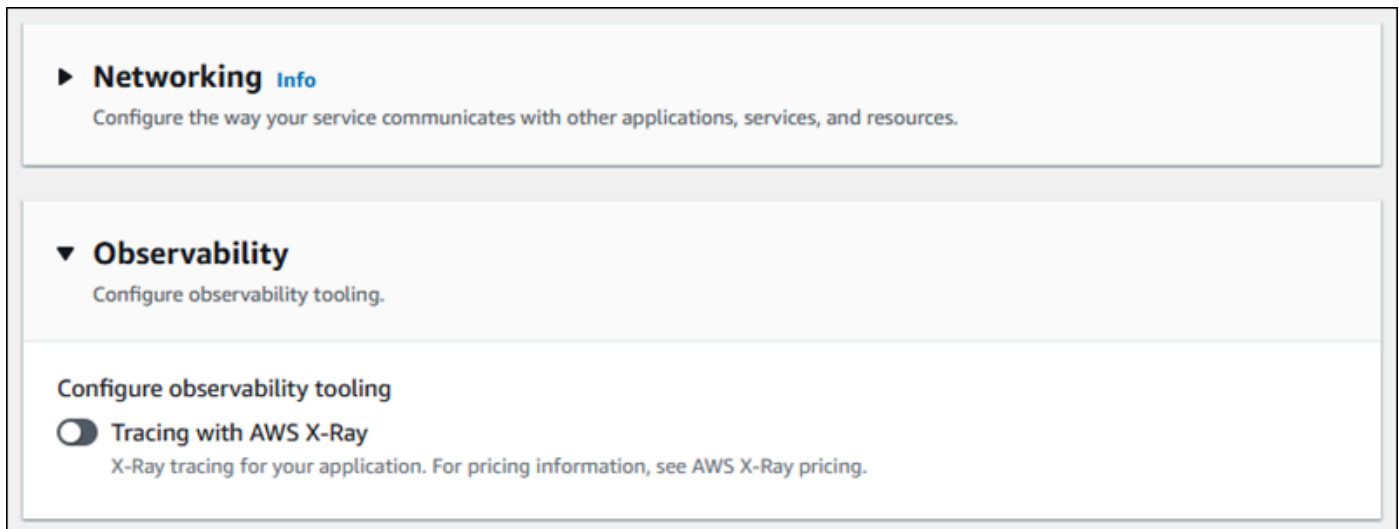
- Configuração de rastreamento — Configurações para rastrear as solicitações que seu aplicativo atende e as chamadas downstream que ele faz. Para obter mais informações sobre rastreamento, consulte [the section called “Rastreamento \(X-Ray\)”](#).

Gerencie a observabilidade

Gerencie a observabilidade dos seus serviços do App Runner usando um dos seguintes métodos:

App Runner console

Ao [criar um serviço](#) usando o console do App Runner ou ao [atualizar sua configuração posteriormente](#), você pode configurar recursos de observabilidade para seu serviço. Procure a seção Configuração de observabilidade na página do console.



App Runner API or AWS CLI

Ao chamar as ações da API [CreateService](#) ou [UpdateService](#) App Runner, você pode usar o objeto de `ObservabilityConfiguration` parâmetro para ativar os recursos de observabilidade e especificar um recurso de configuração de observabilidade para seu serviço.

Use as seguintes ações da API App Runner para gerenciar seus recursos de configuração de observabilidade.

- [CreateObservabilityConfiguration](#) — Cria uma nova configuração de observabilidade ou uma revisão de uma existente.
- [ListObservabilityConfigurations](#) — Retorna uma lista das configurações de observabilidade associadas à sua Conta da AWS, com informações resumidas.

- [DescribeObservabilityConfiguration](#)— Retorna uma descrição completa de uma configuração de observabilidade.
- [DeleteObservabilityConfiguration](#)— Exclui uma configuração de observabilidade. Você pode excluir uma revisão específica ou a última revisão ativa. Talvez seja necessário excluir configurações de observabilidade desnecessárias se atingir a cota de configuração de observabilidade do seu. Conta da AWS

Definindo as configurações do serviço usando recursos compartilháveis

Para alguns recursos, faz sentido compartilhar a configuração entre AWS App Runner os serviços. Por exemplo, talvez você queira que um conjunto de serviços tenha o mesmo comportamento de escalonamento automático. Ou talvez você queira configurações de observabilidade idênticas para todos os seus serviços. O App Runner permite que você compartilhe configurações usando recursos compartilháveis separados. Você cria um recurso que define um conjunto de configurações para um recurso e, em seguida, fornece o Amazon Resource Name (ARN) desse recurso de configuração para um ou mais serviços do App Runner.

O App Runner implementa recursos de configuração compartilháveis para os seguintes recursos:

- [Ajuste de escala automático](#)
- [Observabilidade](#)
- [Acesso por VPC](#)

A página do documento de cada um desses recursos fornece informações sobre as configurações disponíveis e os procedimentos de gerenciamento.

Os recursos que usam recursos de configuração separados compartilham algumas características e considerações de design.

- **Revisões** — Alguns recursos de configuração podem ter revisões. O escalonamento automático e a observabilidade são exemplos de dois recursos de configuração que usam revisões. Nesses casos, cada configuração tem um nome e uma revisão numérica. Várias revisões de uma configuração têm o mesmo nome e números de revisão diferentes. Você pode usar nomes de configuração diferentes para cenários diferentes. Para cada nome, você pode adicionar várias revisões para ajustar as configurações de um cenário específico.

A primeira configuração que você cria com um nome obtém o número de revisão 1. As configurações subsequentes com o mesmo nome recebem números de revisão consecutivos (começando com 2). Você pode associar seu serviço App Runner a uma revisão de configuração específica ou à revisão mais recente da configuração.

- **Compartilhado** — Você pode compartilhar um único recurso de configuração em vários serviços do App Runner. Isso é útil se você quiser manter configurações idênticas nesses serviços. Em particular, se seus recursos oferecem suporte a revisões, você pode configurar vários serviços para usar a revisão mais recente de uma configuração. Você pode fazer isso especificando somente o nome da configuração, mas não uma revisão. Qualquer um dos serviços que você configurou dessa forma recebe atualizações de configuração quando você atualiza o serviço. Para obter mais informações sobre alterações na configuração, consulte [the section called “Configuração”](#).
- **Gerenciamento de recursos** — Você pode usar o App Runner para criar e excluir configurações. Você não pode atualizar diretamente uma configuração. Em vez disso, para recursos que oferecem suporte a revisões, você pode criar uma nova revisão para um nome de configuração existente para atualizar a configuração com eficácia.

 Note

Para escalonamento automático, você pode criar configurações e várias revisões com o console do App Runner e a API do App Runner. Tanto o console do App Runner quanto a API do App Runner também podem excluir configurações e revisões. Consulte mais detalhes em [Gerenciando o escalonamento automático do App Runner](#).

Para outros tipos de configuração, como configurações de observabilidade, você só pode criar uma configuração com uma única revisão com o console do App Runner. Para criar mais revisões e excluir configurações, você deve usar a API App Runner.

- **Cota de recursos** — Há cotas definidas para o número de nomes e revisões de configuração exclusivos que você pode ter para seus recursos de configuração em cada um. Região da AWS Se você atingir essas cotas, deverá excluir um nome de configuração ou pelo menos algumas de suas revisões antes de poder criar mais. Para revisões de configurações de escalonamento automático, você pode usar o console do App Runner ou a API do App Runner para excluí-las. Consulte mais detalhes em [Gerenciando o escalonamento automático do App Runner](#). Você deve usar a API App Runner para excluir outros recursos. Para obter mais informações sobre cotas, consulte [the section called “Cotas de recursos do App Runner”](#).

- Sem custo de recursos — você não incorre em custos adicionais para criar um recurso de configuração. Você pode incorrer em custos pelo recurso em si (por exemplo, você é cobrado pelo AWS X-Ray custo normal ao ativar o rastreamento de X-Ray), mas não pelo recurso de configuração do App Runner que configura o recurso para seu serviço App Runner.

Configurando verificações de saúde para seu serviço

AWS App Runner monitora a integridade do seu serviço realizando verificações de saúde. O protocolo padrão de verificação de integridade é TCP. O App Runner efetua ping no domínio atribuído ao seu serviço. Como alternativa, você pode definir o protocolo de verificação de integridade como HTTP. O App Runner envia solicitações HTTP de verificação de integridade para seu aplicativo web.

Você pode definir algumas configurações relacionadas às verificações de saúde. A tabela a seguir descreve as configurações da verificação de saúde e seus valores padrão.

Configuração	Descrição	Padrão
Protocolo	O protocolo IP que o App Runner usa para realizar verificações de integridade do seu serviço. Se você definir o protocolo comoTCP, o App Runner executará um ping no domínio padrão atribuído ao seu serviço na porta que seu aplicativo está escutando. Se você definir o protocolo comoHTTP, o App Runner enviará solicitações de verificação de integridade para o caminho configurado.	TCP
Path	O URL para o qual o App Runner envia solicitações de verificação de integridade HTTP. Aplicável somente às verificações HTTP.	/
Intervalo	O intervalo de tempo, em segundos, entre as verificações de integridade.	5
Timeout (Tempo limite)	O tempo, em segundos, para aguardar uma resposta de verificação de integridade antes de decidir que ela falhou.	2

Configuração	Descrição	Padrão
Limite saudável	O número de verificações consecutivas que devem ter sucesso antes que o App Runner decida que o serviço está íntegro.	1
Limite insalubre	O número de verificações consecutivas que devem falhar antes que o App Runner decida que o serviço não está íntegro.	5

Configurar verificações de integridade

Configure as verificações de saúde do seu serviço App Runner usando um dos seguintes métodos:

App Runner console

Ao criar seu serviço App Runner usando o console do App Runner ou ao atualizar sua configuração posteriormente, você pode definir as configurações de verificação de integridade. Para obter os procedimentos completos do console, consulte [the section called “Criação”](#) [the section called “Configuração”](#) e. Em ambos os casos, procure a seção de configuração do Health check na página do console.

▼ Health check [Info](#)

Configure load balancer health checks.

Protocol

The IP protocol that App Runner uses to perform health checks for your service.

TCP ▼

Timeout

Amount of time the load balancer waits for a health check response.

5

seconds

Interval

Amount of time between health checks of an individual instance.

10

seconds

Unhealthy threshold

The number of consecutive health check failures that determine an instance is unhealthy.

5

requests

Health threshold

The number of consecutive successful health checks that determine an instance is healthy.

1

requests

App Runner API or AWS CLI

Ao chamar as ações da [UpdateService](#) API [CreateService](#) ou da API, você pode usar o `HealthCheckConfiguration` parâmetro para especificar as configurações da verificação de saúde.

Para obter informações sobre a estrutura do parâmetro, consulte [HealthCheckConfiguration](#) na Referência da AWS App Runner API.

Gerenciando conexões do App Runner

Ao [criar um serviço](#) em AWS App Runner, você configura uma fonte de aplicativo — uma imagem de contêiner ou um repositório de origem que é armazenado com um provedor. O App Runner precisa estabelecer uma conexão autenticada e autorizada com o provedor. Em seguida, o App Runner pode ler seu repositório e implantá-lo em seu serviço. O App Runner não exige o estabelecimento de conexão quando você cria um serviço que acessa o código armazenado em seu. Conta da AWS

O App Runner mantém as informações de conexão em um recurso chamado conexão. O console do App Runner e este guia também se referem às conexões como contas conectadas. O App Runner exige um recurso de conexão quando você cria um serviço que precisa de informações de conexão de terceiros. A seguir estão algumas informações importantes sobre conexões:

- **Provedores** — O App Runner atualmente requer recursos de conexão com o [GitHubBitbucket](#).
- **Compartilhado** — você pode usar um recurso de conexão para criar vários serviços do App Runner que usam a mesma conta do provedor de repositório.
- **Gerenciamento de recursos** — No App Runner, você pode criar e excluir conexões. No entanto, você não pode modificar uma conexão existente.
- **Cota de recursos** — os recursos de conexão têm uma cota definida associada à sua Conta da AWS em cada um. Região da AWS Se você atingir essa cota, talvez seja necessário excluir uma conexão antes de se conectar a uma nova conta de provedor. Você pode excluir uma conexão usando o console ou a API do App Runner, conforme descrito na seção a seguir, [the section called “Gerenciar conexões”](#). Para obter mais informações, consulte [the section called “Cotas de recursos do App Runner”](#).

Gerenciar conexões

Gerencie suas conexões do App Runner usando um dos seguintes métodos:

App Runner console

Ao usar o console do App Runner para [criar um serviço](#), você fornece detalhes da conexão. Você não precisa criar explicitamente um recurso de conexão. No console, você pode escolher se conectar a uma GitHub conta do Bitbucket à qual você já se conectou antes ou se conectar a uma nova conta. Quando necessário, o App Runner cria um recurso de conexão para você. Para uma nova conexão, alguns provedores exigem que você conclua um handshake de autenticação antes de poder usar a conexão. O console conduz você por esse processo.

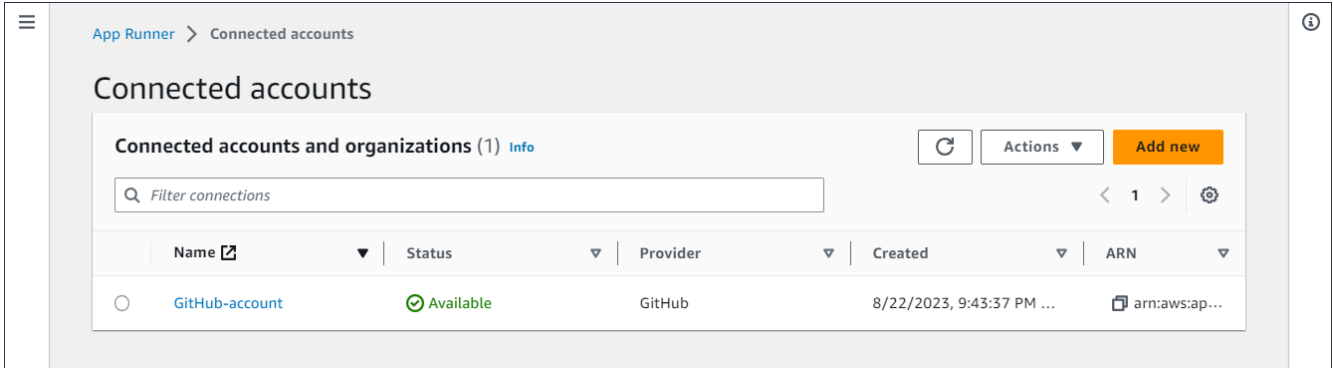
O console também tem uma página para gerenciar suas conexões existentes. Você pode concluir o handshake de autenticação de uma conexão se não tiver feito isso quando criou seu serviço. Você também pode excluir conexões que não está mais usando. O procedimento a seguir mostra como você pode gerenciar as conexões do provedor de repositório.

Para gerenciar conexões em sua conta

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.

2. No painel de navegação, escolha Contas conectadas.

Em seguida, o console exibe uma lista de conexões do provedor de repositório na sua conta.



3. Agora você pode executar uma das seguintes ações com qualquer conexão na lista:

- Abrir GitHub/Bitbucket conta ou organização — Escolha o nome da conexão.
- Handshake de autenticação completo — Selecione a conexão e, no menu Ações, escolha Handshake completo. O console conduz você pelo processo de handshake de autenticação.
- Excluir conexão — Selecione a conexão e, no menu Ações, escolha Excluir. Siga as instruções no aviso de exclusão.

App Runner API or AWS CLI

Você pode usar as seguintes ações da API App Runner para gerenciar suas conexões.

- [CreateConnection](#)— Cria uma conexão com uma conta de provedor de repositório. Depois que a conexão for criada, você deverá concluir manualmente o handshake de autenticação usando o console do App Runner. Esse processo é explicado na seção anterior.
- [ListConnections](#)— Retorna uma lista de conexões do App Runner associadas à sua Conta da AWS.
- [DeleteConnection](#)— Exclui uma conexão. Talvez seja necessário excluir conexões desnecessárias se atingir a cota de conexão do seu Conta da AWS.

Gerenciando o escalonamento automático do App Runner

AWS App Runner aumenta ou diminui automaticamente os recursos computacionais, especificamente instâncias, do seu aplicativo App Runner. O escalonamento automático fornece

tratamento adequado de solicitações quando o tráfego é intenso e reduz seus custos quando o tráfego fica mais lento.

Configuração de escalonamento automático

Você pode configurar alguns parâmetros para ajustar o comportamento de escalonamento automático do seu serviço. O App Runner mantém as configurações de escalonamento automático em um recurso compartilhável chamado `AutoScalingConfiguration`. Você pode criar e manter configurações autônomas de escalonamento automático antes de atribuí-las aos serviços. Depois que eles forem associados a um serviço, você poderá continuar mantendo as configurações. Você também pode optar por criar uma nova configuração de auto scaling enquanto estiver criando um novo serviço ou configurando um existente. Depois que a nova configuração de escalonamento automático for criada, você poderá associá-la ao serviço e continuar com o processo de criação ou configuração do serviço.

Nomenclatura e revisões

Uma configuração de escalonamento automático tem um nome e uma revisão numérica. Várias revisões de uma configuração têm o mesmo nome e números de revisão diferentes. Você pode usar nomes de configuração diferentes para diferentes cenários de escalonamento automático, como alta disponibilidade ou baixo custo. Para cada nome, você pode adicionar várias revisões para ajustar as configurações de um cenário específico. Você pode ter até 10 nomes exclusivos de configuração de auto scaling e até 5 revisões para cada configuração. Se você atingir o limite e precisar criar mais, poderá excluir um e depois criar outro. O App Runner não permitirá que você exclua uma configuração definida como padrão ou em uso por um serviço ativo. Para obter mais informações sobre cotas, consulte [the section called “Cotas de recursos do App Runner”](#).

Definindo uma configuração padrão

Ao criar ou atualizar um serviço do App Runner, você pode fornecer um recurso de configuração de escalonamento automático. Fornecer uma configuração de escalonamento automático é opcional. Se você não fornecer uma, o App Runner fornecerá uma configuração padrão de escalonamento automático com valores recomendados. O recurso de configuração de escalonamento automático oferece a opção de definir sua própria configuração padrão de escalonamento automático em vez de usar o padrão fornecido pelo App Runner. Depois de especificar outra configuração de auto scaling como padrão, essa configuração é automaticamente atribuída como padrão aos novos serviços que você criar no futuro. A nova designação padrão não afeta as associações que foram definidas anteriormente para os serviços existentes.

Configurando serviços com escalonamento automático

Você pode compartilhar uma única configuração de escalonamento automático em vários serviços do App Runner para garantir que os serviços tenham o mesmo comportamento de escalonamento automático. Para obter mais informações sobre como definir configurações de escalonamento automático com o console do App Runner ou a API do App Runner, consulte as seções a seguir neste tópico. Para obter mais informações gerais sobre recursos compartilháveis, consulte [the section called “Recursos de configuração”](#).

Configurações configuráveis

Você pode definir as seguintes configurações de escalonamento automático:

- **Concorrência máxima** — O número máximo de solicitações simultâneas que uma instância processa. Quando o número de solicitações simultâneas excede essa cota, o App Runner aumenta o serviço.
- **Tamanho máximo** — O número máximo de instâncias para as quais seu serviço pode ser escalado. Esse é o maior número de instâncias que podem lidar simultaneamente com o tráfego do seu serviço.
- **Tamanho mínimo** — O número mínimo de instâncias que o App Runner pode provisionar para seu serviço. O serviço sempre tem pelo menos esse número de instâncias provisionadas. Algumas dessas instâncias lidam ativamente com o tráfego. O restante deles faz parte da reserva econômica de capacidade computacional, que está pronta para ser ativada rapidamente. Você paga pelo uso de memória de todas as instâncias provisionadas. Você paga pelo uso da CPU somente do subconjunto ativo.

Note

A contagem de recursos da vCPU determina o número de instâncias que o App Runner pode fornecer ao seu serviço. Esse é um valor de cota ajustável para a contagem de recursos da vCPU Fargate On-Demand que reside no serviço. AWS Fargate Para ver as configurações de cota de vCPU para sua conta ou para solicitar um aumento de cota, use o console Service Quotas no. AWS Management Console Para obter mais informações, consulte as [cotas AWS Fargate de serviço](#) no Amazon Elastic Container Service Developer Guide.

Gerencie o escalonamento automático para um serviço

Gerencie o escalonamento automático dos serviços do App Runner usando um dos seguintes métodos:

App Runner console

Ao [criar um serviço](#) usando o console do App Runner ou [atualizar uma configuração de serviço](#), você pode especificar uma configuração de escalonamento automático.

Note

Quando você altera a configuração ou revisão do auto scaling associada a um serviço, seu serviço é reimplantado.

A página de configuração do Auto Scaling oferece várias opções para configurar o Auto Scaling para seu serviço.

- Para atribuir uma configuração e uma revisão existentes — Escolha um valor no menu suspenso Configurações existentes. A versão de revisão mais recente será padronizada na lista suspensa adjacente. Se existir uma revisão diferente que você prefira selecionar, faça isso na lista suspensa de revisões. Os valores de configuração da versão de revisão são exibidos.
- Para criar e atribuir uma nova configuração de escalonamento automático, selecione Criar novo ASC no menu Criar. Isso abre a página Adicionar configuração personalizada de auto scaling. Insira um nome de configuração e valores para os parâmetros de escalonamento automático. Em seguida, selecione Adicionar. O App Runner cria o novo recurso de configuração de escalonamento automático para você e retorna à seção Escalonamento automático com a nova configuração selecionada e exibida.
- Para criar e atribuir uma nova revisão, primeiro selecione o nome da configuração no menu suspenso Configurações existentes. Em seguida, selecione Criar revisão ASC no menu Criar. Isso abre a página Adicionar configuração personalizada de auto scaling. Insira valores para os parâmetros de escalonamento automático. Em seguida, selecione Adicionar. O App Runner cria uma nova revisão de configuração de escalonamento automático para você e retorna à seção Escalonamento automático com a nova revisão selecionada e exibida.

▼ Auto scaling [Info](#)

Configure automatic scaling behavior.

Auto scaling configurations

Create ▼

Existing configurations

Medium-capacity ▼

v2 ▼

↻

Concurrency

80 requests

Minimum size

8 instance(s)

Maximum size

12 instances

App Runner API or AWS CLI

Ao chamar as ações da API [CreateService](#) ou [UpdateService](#) App Runner, você pode usar o `AutoScalingConfigurationArn` parâmetro para especificar um recurso de configuração de escalonamento automático para seu serviço.

A próxima seção fornece orientação para gerenciar seus recursos de configuração de auto scaling.

Gerencie recursos de configurações de auto scaling

Gerencie as configurações e revisões de auto scaling do App Runner para sua conta usando um dos seguintes métodos:

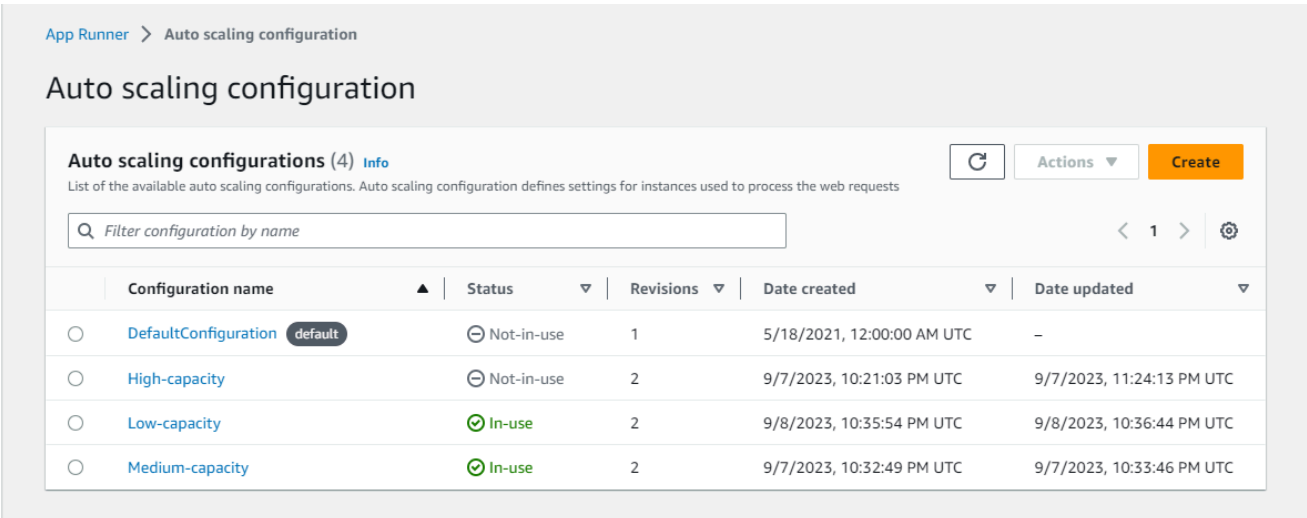
App Runner console

Gerencie configurações de escalonamento automático

A página Configurações do Auto Scaling lista as configurações do Auto Scaling que você configurou na sua conta. Você pode criar e gerenciar suas configurações de auto scaling nesta página e depois atribuí-las a um ou mais serviços do App Runner.

Você pode fazer qualquer uma das ações a seguir nesta página:

- Crie uma nova configuração de escalonamento automático.
- Crie uma nova revisão para uma configuração de auto scaling existente.
- Exclua uma configuração de escalonamento automático.
- Defina uma configuração de escalonamento automático como padrão.



The screenshot displays the 'Auto scaling configuration' page in the AWS App Runner console. At the top, there's a breadcrumb 'App Runner > Auto scaling configuration'. Below it, the title 'Auto scaling configuration' is followed by a sub-header 'Auto scaling configurations (4) Info'. A description states: 'List of the available auto scaling configurations. Auto scaling configuration defines settings for instances used to process the web requests'. There's a search bar 'Filter configuration by name' and a table with columns: Configuration name, Status, Revisions, Date created, and Date updated. The table lists four configurations: 'DefaultConfiguration' (Not-in-use), 'High-capacity' (Not-in-use), 'Low-capacity' (In-use), and 'Medium-capacity' (In-use). The 'Low-capacity' and 'Medium-capacity' configurations are marked as 'In-use' with green checkmarks.

Configuration name	Status	Revisions	Date created	Date updated
DefaultConfiguration default	Not-in-use	1	5/18/2021, 12:00:00 AM UTC	–
High-capacity	Not-in-use	2	9/7/2023, 10:21:03 PM UTC	9/7/2023, 11:24:13 PM UTC
Low-capacity	In-use	2	9/8/2023, 10:35:54 PM UTC	9/8/2023, 10:36:44 PM UTC
Medium-capacity	In-use	2	9/7/2023, 10:32:49 PM UTC	9/7/2023, 10:33:46 PM UTC

Para gerenciar as configurações de escalonamento automático em sua conta

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
2. No painel de navegação, escolha Configurações de escalonamento automático. O console exibe uma lista de configurações de escalonamento automático em sua conta.

Agora você pode fazer qualquer uma das ações a seguir.

- Para criar uma nova configuração de escalonamento automático, siga estas etapas.

- a. Na página Configurações do Auto Scaling, selecione Criar.

A página Criar configuração de escalonamento automático é exibida.

- b. Insira valores para Nome da configuração, Concorrência, Tamanho mínimo e Tamanho máximo.
- c. (Opcional) Se você quiser adicionar tags, selecione Nova tag automática. Em seguida, nos campos que aparecem, insira um Nome e um Valor (opcional).
- d. Escolha Criar.

- Para criar uma nova revisão para uma configuração de auto scaling existente, siga estas etapas.
 - a. Na página Configurações de Auto Scaling, selecione o botão de rádio ao lado da configuração que precisa da nova revisão. Em seguida, selecione Criar revisão no menu Ações.

A página Criar revisão é exibida.

- b. Ativado, insira valores para Concorrência, Tamanho mínimo e Tamanho máximo.
 - c. (Opcional) Se você quiser adicionar tags, selecione Nova tag automática. Em seguida, nos campos que aparecem, insira um Nome e um Valor (opcional).
 - d. Escolha Criar.
- Para excluir uma configuração de escalonamento automático, siga estas etapas.
 - a. Na página Configurações de Auto Scaling, selecione o botão de rádio ao lado da configuração que você precisa excluir.
 - b. Selecione Excluir no menu Ações.
 - c. Para continuar com a exclusão, selecione Excluir na caixa de diálogo de confirmação. Caso contrário, selecione Cancelar.

 Note

O App Runner valida que sua opção de exclusão não está definida como padrão ou está sendo usada atualmente por qualquer serviço ativo.

- Para definir uma configuração de escalonamento automático como padrão, siga estas etapas.
 - a. Na página Configurações de Auto Scaling, selecione o botão de rádio ao lado da configuração que você precisa definir como padrão.
 - b. Selecione Definir como padrão no menu Ações.
 - c. Uma caixa de diálogo é exibida informando que o App Runner usará a revisão mais recente como configuração padrão para todos os novos serviços que você criar. Selecione Confirmar para continuar. Caso contrário, selecione Cancelar.

 Note

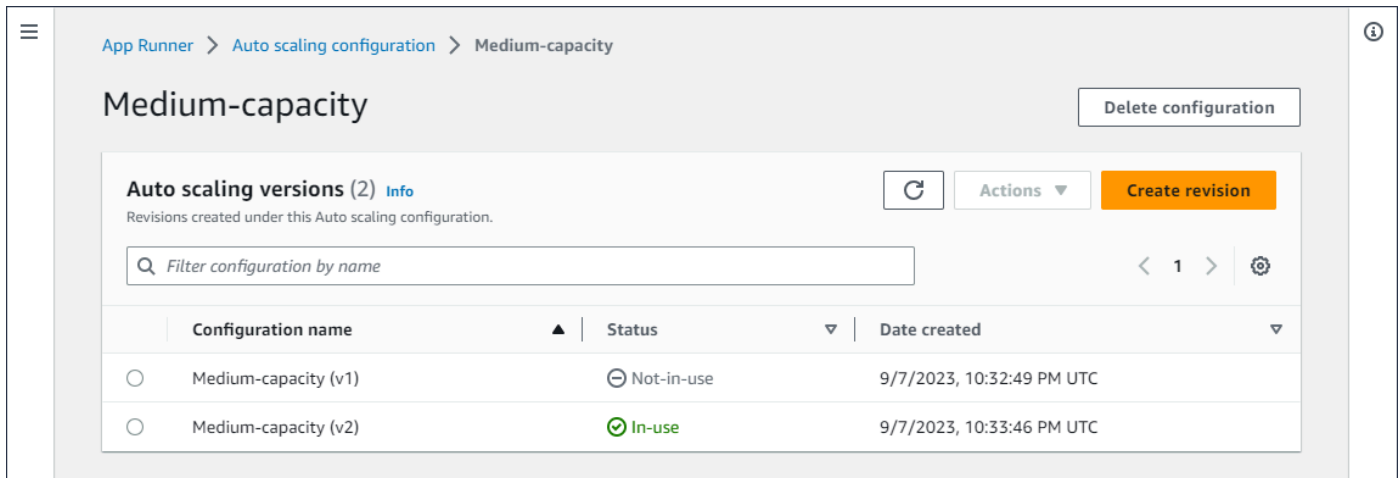
- Quando você define uma configuração de auto scaling como padrão, ela é automaticamente atribuída como configuração padrão aos novos serviços que você criar futuramente.
- A nova designação padrão não afeta as associações que foram definidas anteriormente para os serviços existentes.
- Se a configuração de escalonamento automático padrão designada tiver revisões, o App Runner atribuirá sua revisão mais recente como padrão.

Gerenciar revisões

O console também tem uma página para criar e gerenciar suas revisões de escalonamento automático existentes, chamadas revisões de escalonamento automático. Acesse essa página selecionando o nome de uma configuração na página Configurações de Auto Scaling.

Você pode fazer qualquer uma das ações a seguir na página de revisões do Auto Scaling:

- Crie uma nova revisão de escalonamento automático.
- Defina uma revisão da configuração de escalonamento automático como padrão.
- Exclua uma revisão.
- Exclua toda a configuração do auto scaling, incluindo todas as revisões associadas.
- Veja os detalhes da configuração de uma revisão.
- Veja uma lista dos serviços associados a uma revisão.
- Altere a revisão de um serviço listado.



Para gerenciar revisões de escalonamento automático em sua conta

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Configurações de escalonamento automático. O console exibe uma lista de configurações de escalonamento automático em sua conta. O conjunto anterior de procedimentos na [Gerencie configurações de escalonamento automático](#) seção inclui uma imagem de tela desta página.
3. Agora você pode detalhar uma configuração específica de auto scaling para visualizar e gerenciar todas as suas revisões. No painel Configurações de escalonamento automático, na coluna Nome da configuração, escolha um nome de configurações de escalonamento automático. Selecione o nome real, em vez do botão de rádio. Isso levará você a uma lista de todas as revisões dessa configuração na página de revisões do Auto Scaling.
4. Agora você pode fazer qualquer uma das ações a seguir.
 - Para criar uma nova revisão para uma configuração de auto scaling existente, siga estas etapas.
 - a. Na página Revisões do Auto Scaling, selecione Criar revisão.
A página Criar revisão é exibida.
 - b. Insira valores para Concorrência, Tamanho mínimo e Tamanho máximo.
 - c. (Opcional) Se você quiser adicionar tags, selecione Nova tag automática. Em seguida, nos campos que aparecem, insira um Nome e um Valor (opcional).
 - d. Escolha Criar.
 - Para excluir toda a configuração do auto scaling, incluindo todas as revisões associadas, siga estas etapas.

- a. Selecione Excluir configuração no canto superior direito da página.
- b. Para continuar com a exclusão, selecione Excluir na caixa de diálogo de confirmação. Caso contrário, selecione Cancelar.

 Note

O App Runner valida que sua opção de exclusão não está definida como padrão ou está sendo usada atualmente por qualquer serviço ativo.

- Para definir uma revisão de escalonamento automático como padrão, siga estas etapas.
 - a. Selecione o botão de rádio ao lado da revisão que você precisa definir como padrão.
 - b. Selecione Definir como padrão no menu Ações.

 Note

- Quando você define uma configuração de auto scaling como padrão, ela é automaticamente atribuída como configuração padrão aos novos serviços que você criar futuramente.
- A nova designação padrão não afeta as associações que foram definidas anteriormente para os serviços existentes.

- Para ver os detalhes da configuração de uma revisão, siga estas etapas.
 - Selecione o botão de rádio ao lado da revisão.

Os detalhes da configuração da revisão, incluindo o ARN, são exibidos no painel dividido inferior. Consulte a imagem da tela no final deste procedimento.

- Para ver uma lista dos serviços associados a uma revisão, siga estas etapas.
 - Selecione o botão de rádio ao lado da revisão.

O painel Serviços é exibido no painel dividido inferior, abaixo dos detalhes da configuração da revisão. O painel lista todos os serviços que usam essa revisão de configuração de escalonamento automático. Consulte a imagem da tela no final deste procedimento.

- Para alterar a revisão de um serviço listado, siga estas etapas.
 - a. Selecione o botão de rádio ao lado da revisão, caso ainda não tenha feito isso.

O painel Serviços é exibido no painel dividido inferior, abaixo dos detalhes da configuração da revisão. O painel lista todos os serviços que usam essa revisão de configuração de escalonamento automático. Consulte a imagem da tela no final deste procedimento.
 - b. No painel Serviços, selecione o botão de rádio ao lado do serviço que você deseja modificar. Em seguida, selecione Alterar revisões.
 - c. O painel Alterar revisão do ASC é exibido. Escolha entre as revisões disponíveis no menu suspenso. Somente as revisões da configuração de auto scaling que você escolheu anteriormente estão disponíveis. Se você precisar mudar para uma configuração diferente de escalonamento automático, siga os procedimentos na seção [the section called “Gerencie o escalonamento automático para um serviço”](#) anterior.

Selecione Atualizar para continuar com a alteração. Caso contrário, selecione Cancelar.



Note

Quando você altera uma revisão associada a um serviço, seu serviço é reimplantado.

Você deve selecionar atualizar neste painel para ver as associações atualizadas.

Para ver a atividade contínua e o status da reimplantação do serviço, use as trilhas do painel para navegar até App Runner > Serviços, selecionar o serviço e, em seguida, visualizar a guia Registros no painel Visão geral do serviço.

The screenshot shows the AWS App Runner console interface for an auto scaling configuration named 'Medium-capacity'. The breadcrumb navigation at the top indicates the path: App Runner > Auto scaling configuration > Medium-capacity. A 'Delete configuration' button is located in the top right corner.

The main section is titled 'Medium-capacity' and contains a sub-section for 'Auto scaling versions (2)'. It lists two versions: 'Medium-capacity (v1)' with a status of 'Not-in-use' and 'Medium-capacity (v2)' with a status of 'In-use'. The 'v2' version is selected and highlighted. Below the list, the details for 'Medium-capacity (v2)' are displayed, including its concurrency (80 requests), minimum size (8 instances), maximum size (12 instances), and its ARN.

At the bottom of the console, there is a section for 'Services (2)' that lists two services: 'myAppDev' and 'pythonTest'. Both services are associated with the 'Medium-capacity (v2)' auto scaling configuration.

App Runner API or AWS CLI

Use as seguintes ações da API App Runner para gerenciar seus recursos de configuração de escalonamento automático.

- [CreateAutoScalingConfiguration](#)— Cria uma nova configuração de escalonamento automático ou uma revisão de uma existente.
- [UpdateDefaultAutoScalingConfiguration](#)— Define uma configuração de escalonamento automático como padrão. A configuração de escalonamento automático padrão existente será definida como não padrão automaticamente.
- [ListAutoScalingConfigurations](#)— Retorna uma lista das configurações de escalonamento automático associadas à sua Conta da AWS, com informações resumidas.

- [ListServicesForAutoScalingConfiguration](#)— Retorna uma lista dos serviços associados do App Runner usando uma configuração de escalonamento automático.
- [DescribeAutoScalingConfiguration](#)— Retorna uma descrição completa de uma configuração de escalonamento automático.
- [DeleteAutoScalingConfiguration](#)— Exclui uma configuração de escalonamento automático. Você pode excluir uma configuração de escalonamento automático de nível superior, uma revisão específica de uma ou todas as revisões associadas à configuração de nível superior. Use o `DeleteAllRevisions` parâmetro opcional para excluir todas as revisões. Se você atingir a [cota de recursos](#) de configuração de escalonamento automático para sua Conta da AWS, talvez seja necessário excluir configurações desnecessárias de escalonamento automático.

Gerenciando nomes de domínio personalizados para um serviço App Runner

Quando você cria um AWS App Runner serviço, o App Runner aloca um nome de domínio para ele. Esse é um subdomínio no `awsapprunner.com` domínio de propriedade do App Runner. Você pode usar o nome de domínio para acessar o aplicativo web que está sendo executado no seu serviço.

Note

[Para aumentar a segurança de seus aplicativos App Runner, o domínio*.awsapprunner.com é registrado na Lista Pública de Sufixos \(PSL\).](#) Para maior segurança, recomendamos que você use cookies com um `__Host-` prefixo se precisar definir cookies confidenciais no nome de domínio padrão para seus aplicativos App Runner. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

Se você possui um nome de domínio, pode associá-lo ao seu serviço App Runner. Depois que o App Runner validar seu novo domínio, você poderá usá-lo para acessar seu aplicativo, além do domínio do App Runner. Você pode associar até cinco domínios personalizados.

Note

Opcionalmente, você pode incluir o `www` subdomínio do seu domínio. No entanto, atualmente, isso só é suportado pela API. O console do App Runner não suporta a inclusão do `www` subdomínio do seu domínio.

Note

AWS App Runner não suporta o uso de zonas hospedadas privadas do Route 53. As zonas hospedadas privadas personalizam a resolução de nomes de domínio para o tráfego da Amazon VPC. Para obter mais informações sobre zonas hospedadas privadas, consulte [Trabalho com zonas hospedadas privadas](#) na documentação do Route 53.

Associe (vincule) um domínio personalizado ao seu serviço

Ao associar um domínio personalizado ao seu serviço, você deve adicionar os registros CNAME e os registros de destino DNS ao seu servidor DNS. As seções a seguir fornecem informações sobre registros CNAME e registros de destino DNS e como usá-los.

Note

Se você estiver usando o Amazon Route 53 como seu provedor de DNS, o App Runner configura automaticamente seu domínio personalizado com a validação de certificado e os registros DNS necessários para vincular ao seu aplicativo web App Runner. Isso acontece quando você usa o console do App Runner para vincular seu domínio personalizado ao seu serviço. O [Gerenciar domínios personalizados](#) tópico a seguir fornece mais informações.

Registros CNAME

Quando você associa um domínio personalizado ao seu serviço, o App Runner fornece um conjunto de registros de validação de certificados para validação de certificados. Você deve adicionar esses registros de validação de certificado ao seu servidor de Sistema de Nomes de Domínio (DNS). Adicione os registros de validação do certificado, fornecidos pelo App Runner, ao seu servidor DNS. Dessa forma, o App Runner pode validar que você possui ou controla o domínio.

Note

Para renovar automaticamente seus certificados de domínio personalizados, certifique-se de não excluir os registros de validação de certificados do seu servidor DNS. Para obter informações sobre como resolver problemas relacionados à renovação do certificado, consulte [the section called “Renovação de certificado de domínio personalizado”](#).

O App Runner usa o ACM para verificar o domínio. Se você estiver usando registros CAA em seus registros DNS, certifique-se de que pelo menos um registro CAA faça referência. `amazon.com` Caso contrário, o ACM não poderá verificar o domínio e criar seu domínio com êxito.

Se você receber erros relacionados ao CAA, consulte os links a seguir para saber como resolvê-los:

- [Problemas de Autorização da Autoridade de Certificação \(CAA\)](#)
- [Como faço para resolver erros de CAA ao emitir ou renovar um certificado ACM?](#)
- [Nomes de domínios personalizados](#)

Note

Se você estiver usando o Amazon Route 53 como seu provedor de DNS, o App Runner configura automaticamente seu domínio personalizado com a validação de certificado e os registros DNS necessários para vincular ao seu aplicativo web App Runner. Isso acontece quando você usa o console do App Runner para vincular seu domínio personalizado ao seu serviço. O [Gerenciar domínios personalizados](#) tópico a seguir fornece mais informações.

Registros de destino do DNS

Adicione os registros de destino DNS ao seu servidor DNS para direcionar o domínio do App Runner. Adicione um registro para o domínio personalizado e outro para o `www` subdomínio, se você escolher essa opção. Em seguida, aguarde até que o status do domínio personalizado se torne Ativo no console do App Runner. Isso normalmente leva vários minutos, mas pode levar de 24 a 48 horas (1 a 2 dias). Quando seu domínio personalizado é validado, o App Runner começa a rotear o tráfego desse domínio para seu aplicativo web.

 Note

Para melhor compatibilidade com os serviços do App Runner, recomendamos que você use o Amazon Route 53 como seu provedor de DNS. Se você não usa o Amazon Route 53 para gerenciar seus registros DNS públicos, entre em contato com seu provedor de DNS para descobrir como adicionar registros.

Se você estiver usando o Amazon Route 53 como seu provedor de DNS, você pode adicionar CNAME ou registro de alias para o subdomínio. Para o domínio raiz, certifique-se de usar o registro de alias.

Você pode comprar um nome de domínio do Amazon Route 53 ou de outro provedor. Para comprar um nome de domínio com o Amazon Route 53, consulte [Registro de um novo domínio](#) no Guia do desenvolvedor do Amazon Route 53.

Para obter instruções sobre como configurar um destino de DNS no Route 53, consulte [Roteamento de tráfego para seus recursos](#), no Amazon Route 53 Developer Guide.

Para obter instruções sobre como configurar um destino DNS em outros registradores, como Shopify GoDaddy, Hover e assim por diante, consulte a documentação específica sobre a adição de registros de destino DNS.

Especifique um domínio para associar ao seu serviço App Runner

Você pode especificar um domínio para associar ao seu serviço App Runner das seguintes formas:

- Um domínio raiz — O DNS tem algumas limitações inerentes que podem impedir você de criar registros CNAME para o nome de domínio raiz. Por exemplo, se seu nome de domínio `forexample.com`, você pode criar um registro CNAME que direciona o tráfego `acme.example.com` para o seu serviço App Runner. No entanto, você não pode criar um registro CNAME que direcione o tráfego `example.com` para o seu serviço App Runner. Para criar um domínio raiz, certifique-se de adicionar um registro de alias.

Um registro de alias é específico do Route 53 e tem as seguintes vantagens em relação aos registros CNAME:

- O Route 53 oferece mais flexibilidade, pois registros de alias podem ser criados para domínio raiz ou subdomínio. Por exemplo, se seu nome de domínio `forexample.com`, você pode criar um registro que encaminha solicitações para `example.com` ou `acme.example.com` para seu serviço App Runner.

- É mais econômico. Isso ocorre porque o Route 53 não cobra por solicitações que usam um registro de alias para rotear o tráfego.
- Um subdomínio — por exemplo, `login.example.com` ou `admin.login.example.com`. Opcionalmente, você também pode associar o `www` subdomínio como parte da mesma operação. Você pode adicionar um registro CNAME ou alias para o subdomínio.
- Um curinga — Por exemplo, `*.example.com`. Você não pode usar a `www` opção nesse caso. Você pode especificar um curinga somente como o subdomínio imediato de um domínio raiz e somente por si só. Essas não são especificações válidas: `login*.example.com`, `*.login.example.com`. Essa especificação curinga associa todos os subdomínios imediatos e não associa o domínio raiz em si. O domínio raiz deve ser associado em uma operação separada.

Uma associação de domínio mais específica substitui uma menos específica. Por exemplo, `login.example.com` substitui `*.example.com`. O certificado e o CNAME da associação mais específica são usados.

O exemplo a seguir mostra como você pode usar várias associações de domínio personalizadas:

1. `example.com` Associe-se à página inicial do seu serviço. Habilite `www` a associação `www.example.com`.
2. `login.example.com` Associe-se à página de login do seu serviço.
3. `*.example.com` Associe a uma página personalizada “não encontrada”.

Desassociar (desvincular) um domínio personalizado

Você pode desassociar (desvincular) um domínio personalizado do seu serviço App Runner. Quando você desvincula um domínio, o App Runner interrompe o roteamento do tráfego desse domínio para seu aplicativo web.

Note

Você deve excluir os registros do domínio que você desassociou do seu servidor DNS.

O App Runner cria internamente certificados que rastreiam a validade do domínio. Esses certificados são armazenados no AWS Certificate Manager (ACM). O App Runner não exclui esses certificados por 7 dias depois que um domínio é desassociado do seu serviço ou depois que o serviço é excluído.

Gerenciar domínios personalizados

Gerencie domínios personalizados para seu serviço App Runner usando um dos seguintes métodos:

Note

Para melhor compatibilidade com os serviços do App Runner, recomendamos que você use o Amazon Route 53 como seu provedor de DNS. Se você não usa o Amazon Route 53 para gerenciar seus registros DNS públicos, entre em contato com seu provedor de DNS para descobrir como adicionar registros.

Se você estiver usando o Amazon Route 53 como seu provedor de DNS, você pode adicionar CNAME ou registro de alias para o subdomínio. Para o domínio raiz, certifique-se de usar o registro de alias.

App Runner console

Para associar (vincular) um domínio personalizado usando o console do App Runner

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.

The screenshot shows the AWS App Runner console for a service named 'python-test'. The breadcrumb navigation is 'App Runner > Services > python-test'. The service name 'python-test' is displayed with an 'Info' link. To the right are 'Actions', a refresh icon, and a 'Deploy' button. The 'Service overview' section contains the following information:

- Status:** Running
- Default domain:** <https://62wwc8evee.public.gamma.us-east-1.bullet.aws.dev>
- Service ARN:** `arn:aws:apprunner:us-east-1:123456789012:service/python-test/33f9aa7c44744fbc961e85014386b0d`
- Source:** https://github.com/your_account/python-hello/main

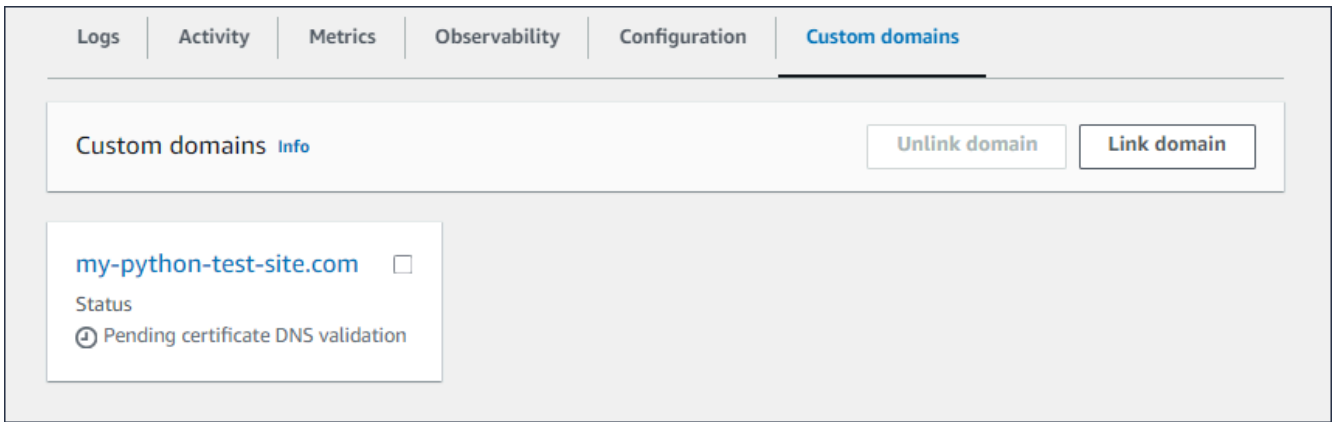
Below the overview is a tabbed interface with 'Logs', 'Activity', 'Metrics', 'Observability', 'Configuration', and 'Custom domains'. The 'Activity' tab is selected, showing 'Activity (1) Info'. It includes a search bar 'Filter activities' and a table of activities:

Operation	Status	Started	Ended
Create service	Succeeded	3/22/2022, 6:46:22 PM UTC	3/22/2022, 6:51:35 PM UTC

3. Na página do painel do serviço, escolha a guia Domínios personalizados.

O console mostra os domínios personalizados associados ao seu serviço ou Nenhum domínio personalizado.

The screenshot shows the 'Custom domains' tab in the AWS App Runner console. The breadcrumb navigation is 'App Runner > Services > python-test > Custom domains'. The tab 'Custom domains' is selected. The section title is 'Custom domains Info'. To the right are 'Unlink domain' and 'Link domain' buttons. Below this, it says 'No custom domains' and 'Your service can always be found at: <https://gnvmp7tpys.us-east-1.awsapprunner.com>'. At the bottom is a 'Link domain' button.



4. Na guia Domínios personalizados, escolha Vincular domínio.
5. A página Link de domínio personalizado é exibida.
 - Se seu domínio personalizado estiver registrado no Amazon Route 53, selecione Amazon Route 53 como registrador de domínio.
 - a. Selecione o nome do domínio na lista suspensa. Essa lista exibe o nome dos seus nomes de domínio do Route 53 e o ID da zona hospedada.
 - b. Selecione o tipo de registro DNS.
 - c. Escolha Vincular domínio.

**Note**

Primeiro, você deve criar um domínio do Route 53 usando o serviço Amazon Route 53 a partir da mesma AWS conta que você usa para gerenciar seus outros recursos do App Runner.

Link custom domain Info

Custom domains can be provided from Amazon or a third-party provider and must have a certificate to ensure a secure connection.

Link custom domain Info

Link a custom domain that you own. App Runner uses https in hyperlinks to your domain.

Domain registrar

☒ Amazon Route 53

☐ Non-Amazon

Domain registrar

JU

DNS record type

☒ ALIAS

☐ CNAME

CancelLink domain

Note

Se o App Runner exibir uma mensagem de erro informando que a tentativa de configuração automática falhou, você poderá continuar configurando os registros DNS manualmente. Esse problema pode surgir se o mesmo nome de domínio tiver sido previamente desvinculado de um serviço, sem que os registros do provedor de DNS que apontam para o serviço sejam excluídos posteriormente. Nesse caso, o App Runner está impedido de sobrescrever automaticamente esses registros. Para concluir a configuração do DNS, pule o restante das etapas desse procedimento e siga as instruções em [Configurar um registro de alias do Amazon Route 53](#)

- Se seu domínio personalizado estiver registrado em outro registrador de domínio, selecione Non—Amazon para registrador de domínios.
 - a. Insira o nome do domínio.
 - b. Escolha Vincular domínio.

Link custom domain [Info](#)

Custom domains can be provided from Amazon or a third-party provider and must have a certificate to ensure a secure connection.

Link custom domain [Info](#)

Link a custom domain that you own. App Runner uses https in hyperlinks to your domain.

Domain registrar

☐ Amazon Route 53

☒ Non-Amazon

Domain name

apprunnertestservice.com

[Cancel](#) [Link domain](#)

6. A página Configurar DNS é exibida.

- Se Amazon Route 53 for seu provedor de DNS, essa etapa é opcional.

Nesse ponto, o App Runner configurou automaticamente seu domínio do Route 53 com a validação de certificado e os registros DNS necessários.

Note

Se esse mesmo nome de domínio tivesse sido anteriormente desvinculado de um serviço, sem que os registros do provedor de DNS que apontam para a exclusão posterior do serviço, a configuração automática que o App Runner tentou fazer poderia ter falhado. Para contornar esse problema e concluir a associação de DNS, continue com as etapas (1) e (2) na página Configurar DNS para copiar os registros atuais de destino e certificado para o provedor de DNS.

- Copie os registros de validação do certificado e os registros de destino do DNS e adicione-os ao seu servidor DNS. O App Runner pode então validar que você possui ou controla o domínio.

 Note

Para renovar automaticamente seus certificados de domínio personalizados, certifique-se de não excluir os registros de validação de certificados do seu servidor DNS.

- Para obter mais informações sobre Configurar a validação de certificados, consulte [Validação de DNS](#) no [Guia do AWS Certificate Manager Usuário](#).
- Para obter informações sobre como configurar o destino DNS com o registro de alias do Amazon Route 53, consulte. [the section called “Configurar um registro de alias do Amazon Route 53”](#)
- Se você estiver usando um provedor de DNS diferente do Amazon Route 53, siga estas etapas.
- Copie os registros de validação do certificado e os registros de destino do DNS e adicione-os ao seu servidor DNS. O App Runner pode então validar que você possui ou controla o domínio.

 Note

Para renovar automaticamente seus certificados de domínio personalizados, certifique-se de não excluir os registros de validação de certificados do seu servidor DNS.

- Para obter mais informações sobre Configurar a validação de certificados, consulte [Validação de DNS](#) no [Guia do AWS Certificate Manager Usuário](#).
- Para obter instruções sobre como configurar um destino DNS em outros registradores, como Shopify GoDaddy, Hover e assim por diante, consulte a documentação específica sobre a adição do DNS Target.

[App Runner](#) > [Services](#) > [python-test](#) > [Configure DNS](#)

my-python-test-site.com [Info](#)

[Unlink domain](#)[Close](#)

Configure DNS

1. Configure certificate validation

Supply CNAME records to your DNS provider within 72 hours.

Record name		Value	
<code>_761caaec9295b45520d472a35119b21e.my-python-test-site.com.</code>	Copy	<code>_a0536edab0ac0a672b661d02bbb6ad49.yxmgqtjrrf.acm-validations.aws.</code>	Copy
<code>_d302cb75f0113815aa3aa0cc7bfdb72.2a57j781ztda5joakq20j1ljwritpe.my-python-test-site.com.</code>	Copy	<code>_b8dd42350638056fc170d5381bea9475.yxmgqtjrrf.acm-validations.aws.</code>	Copy

2. Configure DNS target

Supply this to your DNS provider for the destination of CNAME or ALIAS records.

Record name		Value	
<code>my-python-test-site.com</code>	Copy	<code>gnvmp7tpys.us-east-1.awsapprunner.com</code>	Copy

3. Wait for status to become 'Active'

It can take 24-48 hours after adding the records for the status to change.

Status

[🕒](#) Pending certificate DNS validation

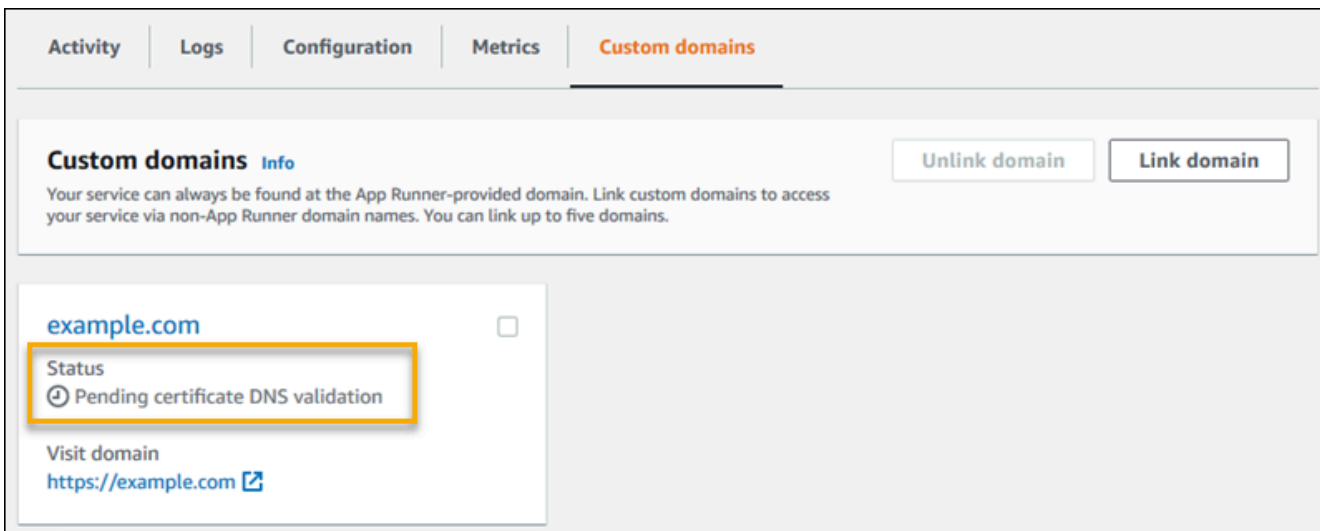
4. Verify

Verify that your service is available at the custom domain.

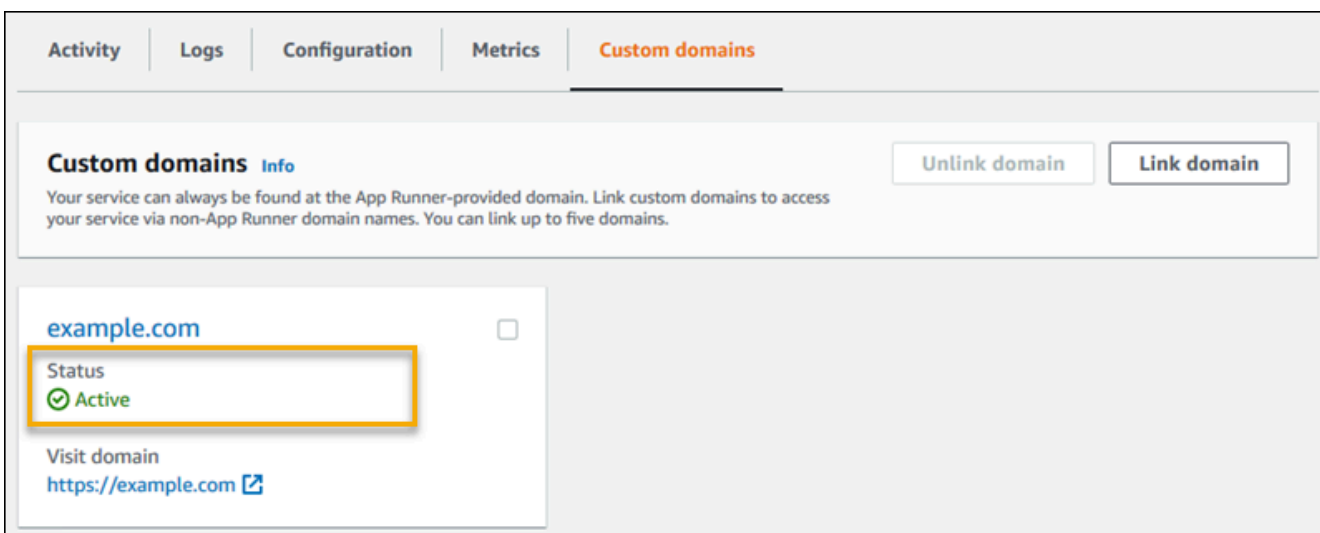
<https://my-python-test-site.com> [🔗](#)

7. Escolha Fechar

O console mostra o painel novamente. A guia Domínios personalizados tem um novo quadro mostrando o domínio que você acabou de vincular no status de validação de DNS do certificado pendente.



- Quando o status do domínio mudar para Ativo, verifique se o domínio funciona para rotear o tráfego navegando até ele.



Note

Para obter instruções sobre como solucionar erros relacionados ao domínio personalizado, consulte [the section called “Nomes de domínios personalizados”](#).

Para desassociar (desvincular) um domínio personalizado usando o console do App Runner

- Na guia Domínios personalizados, selecione o quadro do domínio que você deseja desassociar e escolha Desvincular domínio.

2. Na caixa de diálogo Desvincular domínio, verifique a ação escolhendo Desvincular domínio.

 Note

Você deve excluir os registros do domínio que você desassociou do seu servidor DNS.

App Runner API or AWS CLI

Para associar um domínio personalizado ao seu serviço usando a API App Runner ou AWS CLI chame a ação da [AssociateCustomDomain](#) API. Quando a chamada é bem-sucedida, é retornado um [CustomDomain](#) objeto que descreve o domínio personalizado que está sendo associado ao seu serviço. O objeto mostra um CREATING status e contém uma lista de [CertificateValidationRecord](#) objetos. A chamada também retorna o alias de destino que você pode usar para configurar o destino DNS. Esses são registros que você pode adicionar ao seu DNS.

Para desassociar um domínio personalizado do seu serviço usando a API App Runner ou AWS CLI chame a ação da [DisassociateCustomDomain](#) API. Quando a chamada é bem-sucedida, é retornado um [CustomDomain](#) objeto que descreve o domínio personalizado que está sendo desassociado do seu serviço. O objeto mostra um DELETING status.

Tópicos

- [Configure o registro de alias do Amazon Route 53 para seu DNS de destino](#)

Configure o registro de alias do Amazon Route 53 para seu DNS de destino

 Note

Você não precisa seguir esse procedimento se o Amazon Route 53 for seu provedor de DNS. Nesse caso, o App Runner configura automaticamente seu domínio do Route 53 com a validação de certificado e os registros DNS necessários para vincular ao seu aplicativo web App Runner.

Se a tentativa de configuração automática do App Runner falhar, siga este procedimento para concluir a configuração do DNS. Se o mesmo nome de domínio tiver sido anteriormente desvinculado de um serviço, sem que os registros do provedor de DNS que apontam para o serviço sejam excluídos posteriormente, o App Runner será impedido de sobrescrever

automaticamente esses registros. Este procedimento explica como copiá-los manualmente para o DNS do Route 53.

Você pode usar o Amazon Route 53 como seu provedor de DNS para rotear o tráfego para o seu serviço App Runner. É um serviço web de Sistema de Nomes de Domínio (DNS) altamente disponível e escalável. O registro do Amazon Route 53 contém as configurações que controlam como o tráfego é roteado para o seu serviço App Runner. Você cria um registro CNAME ou um registro ALIAS. Para uma comparação entre registros CNAME e alias, consulte [Escolha entre registros alias e sem alias](#), no Amazon Route 53 Developer Guide.

 Note

Atualmente, o Amazon Route 53 oferece suporte ao registro de alias para serviços criados após 1º de agosto de 2022.

Amazon Route 53 console

Para configurar o registro de alias do Amazon Route 53

1. Faça login AWS Management Console e abra o [console do Route 53](#).
2. No painel de navegação, escolha Zonas hospedadas.
3. Escolha o nome da zona hospedada que você deseja usar para rotear o tráfego para seu serviço App Runner.
4. Escolha Create record (Criar registro).
5. Especifique os seguintes valores:
 - Política de roteamento: escolha a política de roteamento aplicável. Para obter mais informações, consulte [Escolha de uma política de roteamento](#).
 - Nome do registro: insira o nome de domínio que você deseja usar para rotear o tráfego para seu serviço App Runner. O valor padrão é o nome da hosted zone. Por exemplo, se o nome da zona hospedada for `example.com` e você quiser usar para `acme.example.com` rotear o tráfego para seu ambiente, insira `acme`.
 - Valorizar/rotear o tráfego para: escolha Alias para o aplicativo App Runner e, em seguida, escolha a região de origem do endpoint. Escolha o nome de domínio do aplicativo para o qual você deseja rotear o tráfego.

- Tipo de registro: aceite o IPv4 endereço padrão, A —.
- Avalie a integridade do alvo: aceite o valor padrão, Sim.

6. Escolha Criar registros.

O registro de alias do Route 53 que você criou é propagado em todos os servidores do Route 53 em 60 segundos. Quando os servidores do Route 53 são propagados com seu registro de alias, você pode rotear o tráfego para seu serviço App Runner usando o nome do registro de alias que você criou.

Para obter informações sobre como solucionar problemas se as alterações de DNS estão demorando muito para se propagar, consulte [Por que minhas alterações de DNS estão demorando tanto para se propagarem no Route 53 e nos](#) resolvedores públicos? .

Amazon Route 53 API or AWS CLI

Para configurar o registro de alias do Amazon Route 53 usando a API do Amazon Route 53 ou AWS CLI chamar a ação da [ChangeResourceRecordSets](#) API. Para saber mais sobre o ID da zona hospedada de destino do Route 53, consulte [Endpoints de serviço](#).

Pausar e retomar um serviço do App Runner

Se precisar desativar temporariamente seu aplicativo web e interromper a execução do código, você pode pausar seu AWS App Runner serviço. O App Runner reduzirá a capacidade computacional do serviço a zero.

Quando estiver pronto para executar seu aplicativo novamente, você poderá retomar o serviço App Runner. O App Runner provisiona nova capacidade computacional, implanta nela a aplicação e executa a aplicação. A fonte do seu aplicativo não foi reimplantada e nenhuma compilação é necessária. Em vez disso, o App Runner continua com sua versão atualmente implantada. Seu aplicativo mantém seu domínio App Runner.

Important

- Quando você pausa seu serviço, seu aplicativo perde seu estado. Por exemplo, qualquer armazenamento efêmero que seu código usou é perdido. Para seu código, pausar e retomar seu serviço é o equivalente a implantar em um novo serviço.

- Se você pausar um serviço devido a uma falha no código (por exemplo, um bug descoberto ou um problema de segurança), não poderá implantar uma nova versão antes de retomar o serviço.

Portanto, recomendamos que você mantenha o serviço em execução e, em vez disso, reverta para a última versão estável do aplicativo.

- Quando você retoma seu serviço, o App Runner implanta a última versão do aplicativo que foi usada antes de você pausar o serviço. Se você adicionou novas versões de origem desde a pausa do serviço, o App Runner não as implanta automaticamente, mesmo que a implantação automática seja selecionada. Por exemplo, suponha que você tenha novas versões de imagem no repositório de imagens ou novos commits no repositório de código. Essas versões não são implantadas automaticamente.

Para implantar uma versão mais recente, execute uma implantação manual ou adicione outra versão ao seu repositório de origem depois de retomar o serviço App Runner.

Comparação entre pausar e excluir

Pause seu serviço App Runner para desativá-lo temporariamente. Somente os recursos computacionais são encerrados e seus dados armazenados (por exemplo, a imagem do contêiner com a versão do seu aplicativo) permanecem intactos. A retomada do serviço é rápida — seu aplicativo está pronto para ser implantado em novos recursos computacionais. Seu domínio do App Runner permanece o mesmo.

Exclua seu serviço App Runner para removê-lo permanentemente. Seus dados armazenados são excluídos. Se você precisar recriar o serviço, o App Runner precisará buscar sua fonte novamente e também criá-la se for um repositório de código. A aplicação Web obtém um novo domínio do App Runner.

Quando seu serviço está pausado

Quando você pausa seu serviço e ele está no status Pausado, ele responde de forma diferente às solicitações de ação, incluindo chamadas de API ou operações de console. Quando um serviço é pausado, você ainda pode realizar ações do App Runner que não modifiquem a definição ou a configuração do serviço de uma forma que afete seu tempo de execução. Em outras palavras, se uma ação alterar o comportamento, a escala ou outras características de um serviço em execução, você não poderá executar essa ação em um serviço pausado.

As listas a seguir fornecem informações sobre ações de API que você pode e não pode realizar em um serviço pausado. As operações equivalentes do console também são permitidas ou negadas.

Ações que você pode executar em um serviço pausado

- *List**e *Describe** ações — Ações que só leem informações.
- *DeleteService*— Você sempre pode excluir um serviço.
- *TagResource*, *UntagResource* — As tags estão associadas a um serviço, mas não fazem parte de sua definição e não afetam seu comportamento em tempo de execução.

Ações que você não pode executar em um serviço pausado

- *StartDeployment*ações (ou uma [implantação manual](#) usando o console)
- *UpdateService*(ou uma alteração na configuração usando o console, exceto para alterações de marcação)
- *CreateCustomDomainAssociations*, *DeleteCustomDomainAssociations*
- *CreateConnection*, *DeleteConnection*

Pausar e retomar seu serviço

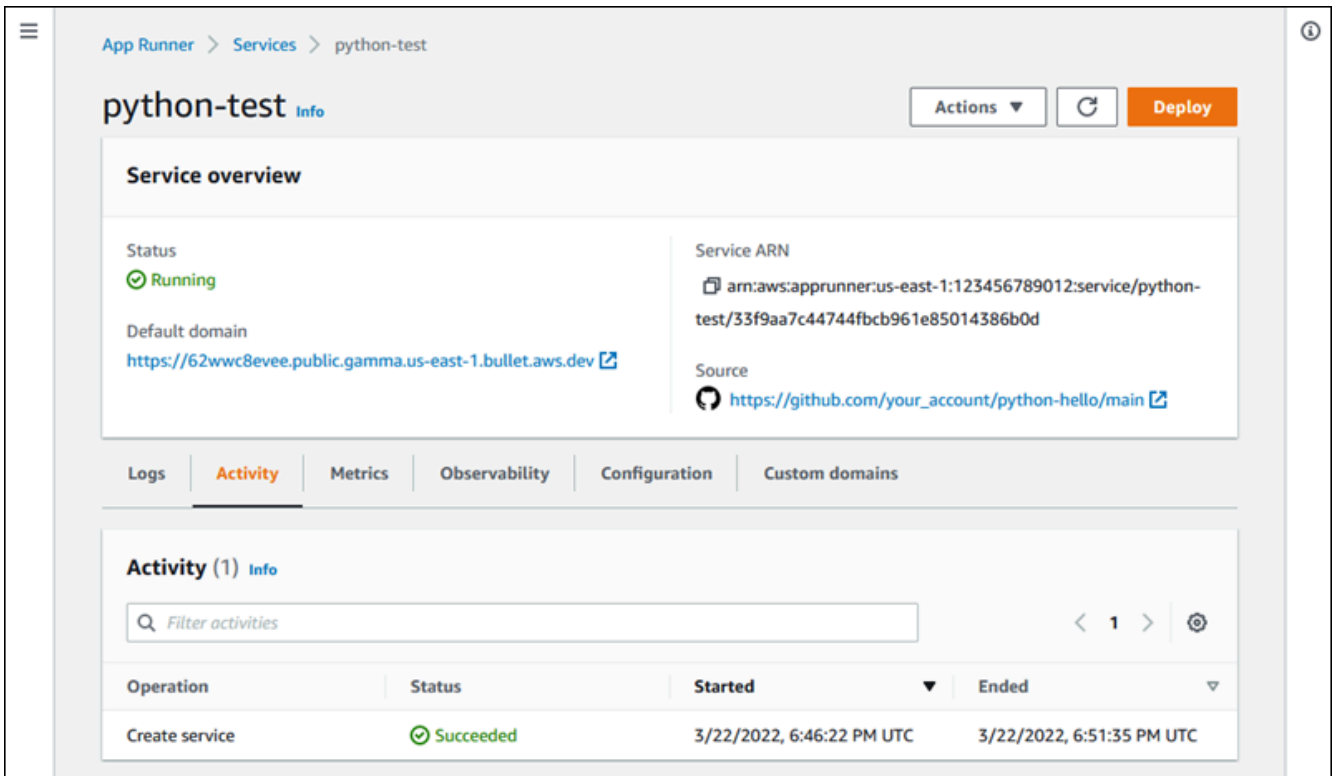
Pause e retome seu serviço App Runner usando um dos seguintes métodos:

App Runner console

Para pausar seu serviço usando o console do App Runner

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Escolha Ações e, em seguida, escolha Pausar.

Na página do painel do serviço, o status do serviço muda para Operação em andamento e, em seguida, muda para Pausado. Seu serviço agora está pausado.

Para retomar seu serviço usando o console do App Runner

1. Escolha Ações e, em seguida, escolha Continuar.

Na página do painel do serviço, o status do serviço muda para Operação em andamento.

2. Aguarde até que o serviço seja retomado. Na página do painel do serviço, o status do serviço volta para Em execução.
3. Para verificar se a retomada do serviço foi bem-sucedida, na página do painel do serviço, escolha o valor do domínio App Runner. É a URL do site do seu serviço. Verifique se seu aplicativo web está sendo executado corretamente.

App Runner API or AWS CLI

Para pausar seu serviço usando a API App Runner ou chame AWS CLI a ação da [PauseServiceAPI](#). Se a chamada retornar uma resposta bem-sucedida com a exibição de um

objeto [Service](#) "Status": "OPERATION_IN_PROGRESS", o App Runner começará a pausar seu serviço.

Para retomar seu serviço usando a API App Runner ou AWS CLI chame a ação da [ResumeService](#) API. Se a chamada retornar uma resposta bem-sucedida com a exibição de um objeto [Service](#) "Status": "OPERATION_IN_PROGRESS", o App Runner começará a retomar seu serviço.

Excluindo um serviço do App Runner

Quando quiser encerrar o aplicativo web que está sendo executado no seu AWS App Runner serviço, você pode excluir o serviço. A exclusão de um serviço interrompe a execução do serviço Web, remove os recursos subjacentes e exclui os dados associados.

Talvez você queira excluir um serviço do App Runner por um ou mais dos seguintes motivos:

- Você não precisa mais do aplicativo web — por exemplo, ele foi retirado ou é uma versão de desenvolvimento que você acabou de usar.
- Você atingiu a cota de serviço do App Runner — Você deseja criar um novo serviço no mesmo Região da AWS e atingiu a cota associada à sua conta. Para obter mais informações, consulte [the section called “Cotas de recursos do App Runner”](#).
- Considerações de segurança ou privacidade — Você deseja que o App Runner exclua os dados que ele armazena para o seu serviço.

Comparação entre pausar e excluir

Pause seu serviço App Runner para desativá-lo temporariamente. Somente os recursos computacionais são encerrados e seus dados armazenados (por exemplo, a imagem do contêiner com a versão do seu aplicativo) permanecem intactos. A retomada do serviço é rápida — seu aplicativo está pronto para ser implantado em novos recursos computacionais. Seu domínio do App Runner permanece o mesmo.

Exclua seu serviço App Runner para removê-lo permanentemente. Seus dados armazenados são excluídos. Se você precisar recriar o serviço, o App Runner precisará buscar sua fonte novamente e também criá-la se for um repositório de código. A aplicação Web obtém um novo domínio do App Runner.

O que o App Runner exclui?

Quando você exclui seu serviço, o App Runner exclui alguns itens associados e não exclui outros. As listas a seguir fornecem os detalhes.

Itens que o App Runner exclui:

- Imagem de contêiner — uma cópia da imagem que você implantou ou da imagem que o App Runner criou a partir do seu código-fonte. Ele é armazenado no Amazon Elastic Container Registry (Amazon ECR) usando componentes Contas da AWS internos que são de propriedade do App Runner.
- Configuração do serviço — As configurações associadas ao seu serviço App Runner. Eles são armazenados no Amazon DynamoDB usando sistemas Contas da AWS internos que são de propriedade do App Runner.

Itens que o App Runner não exclui:

- Conexão — Você pode ter uma conexão associada ao seu serviço. Uma conexão do App Runner é um recurso separado que pode ser compartilhado entre vários serviços do App Runner. Se você não precisar mais da conexão, poderá excluí-la explicitamente. Para obter mais informações, consulte [the section called “Conexões”](#).
- Certificados de domínio personalizados — Se você vincular domínios personalizados a um serviço do App Runner, o App Runner cria internamente certificados que rastreiam a validade do domínio. Eles são armazenados em AWS Certificate Manager (ACM). O App Runner não exclui o certificado por sete dias depois que um domínio é desvinculado do seu serviço ou depois que o serviço é excluído. Para obter mais informações, consulte [the section called “Nomes de domínios personalizados”](#).

Exclua seu serviço

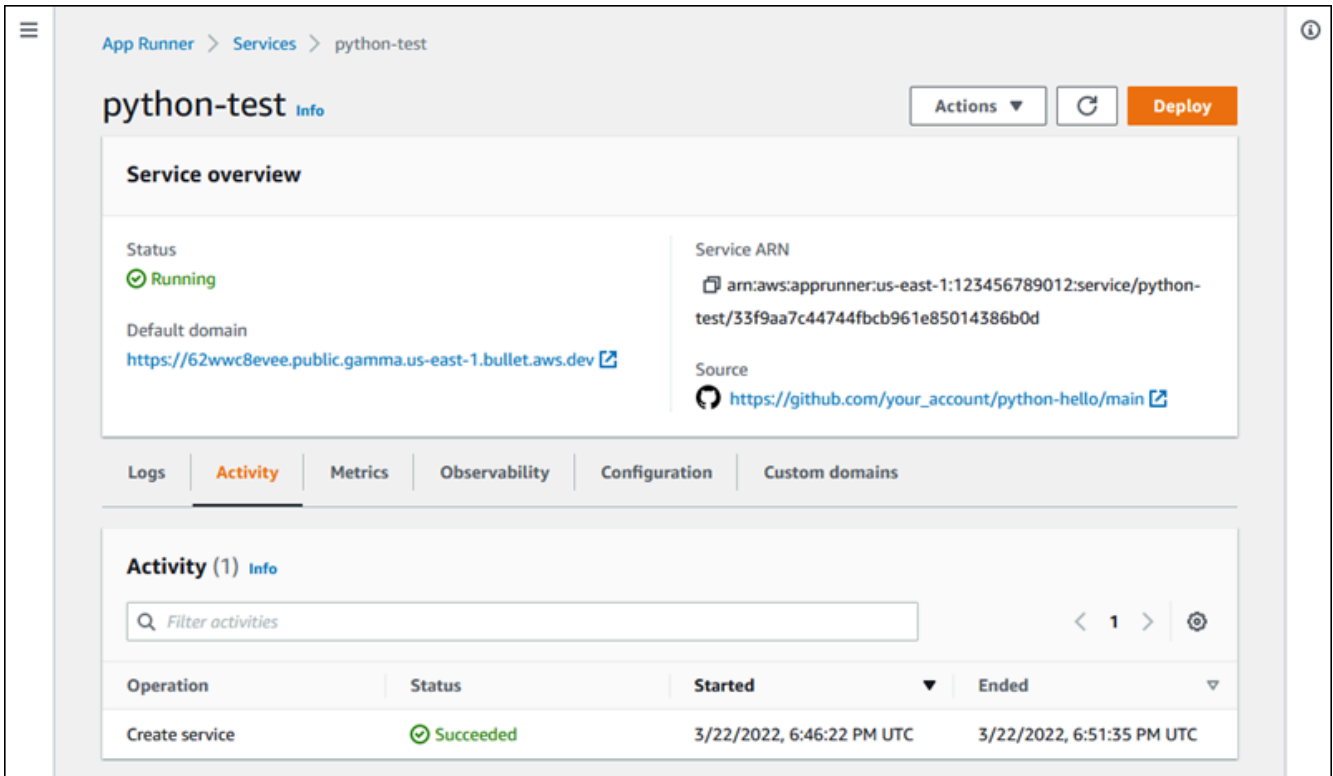
Exclua seu serviço App Runner usando um dos seguintes métodos:

App Runner console

Para excluir seu serviço usando o console do App Runner

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Escolha Ações e, em seguida, escolha Excluir.

O console leva você para a página Serviços. O serviço excluído exibe o status Operação em andamento, e depois desaparece da lista. Seu serviço agora está excluído.

App Runner API or AWS CLI

Para excluir seu serviço usando a API App Runner ou AWS CLI chame a ação da [DeleteService](#) API. Se a chamada retornar uma resposta bem-sucedida com a exibição de um objeto [Service](#) "Status": "OPERATION_IN_PROGRESS", o App Runner começará a excluir seu serviço.

Fazendo referência a variáveis de ambiente

Com o App Runner, você pode referenciar segredos e configurações como variáveis de ambiente em seu serviço ao [criar um serviço](#) ou [atualizar um](#) serviço.

Você pode referenciar dados de configuração não confidenciais, como tempos limite e contagens de novas tentativas em texto sem formatação, como pares de valores-chave. Os dados de configuração que você faz referência em texto sem formatação não são criptografados e são visíveis para outras pessoas na configuração do serviço App Runner e nos registros do aplicativo.

Note

Por motivos de segurança, não faça referência a nenhum dado confidencial em texto sem formatação no seu serviço App Runner.

Referenciando dados confidenciais como variáveis de ambiente

O App Runner oferece suporte para referenciar com segurança dados confidenciais como variáveis de ambiente em seu serviço. Considere armazenar os dados confidenciais que você deseja referenciar no AWS Secrets Manager ou AWS Systems Manager Parameter Store. Em seguida, você pode referenciá-las com segurança em seu serviço como variáveis de ambiente no console do App Runner ou chamando a API. Isso separa efetivamente o gerenciamento de segredos e parâmetros do código do aplicativo e da configuração do serviço, melhorando a segurança geral dos aplicativos em execução no App Runner.

Note

O App Runner não cobra por referenciar o Secrets Manager e o SSM Parameter Store como variáveis de ambiente. No entanto, você paga o preço padrão pelo uso do Secrets Manager e do SSM Parameter Store.

Para obter mais informações sobre definição de preço, consulte o seguinte:

- [AWS Preços do Secrets Manager](#)
- [AWS Preços do SSM Parameter Store](#)

A seguir está o processo para referenciar dados confidenciais como variáveis de ambiente:

1. Armazene dados confidenciais, como chaves de API, credenciais de banco de dados, parâmetros de conexão de banco de dados ou versões de aplicativos como segredos ou parâmetros em um AWS Secrets Manager ou no AWS Systems Manager Parameter Store.
2. Atualize a política do IAM da sua função de instância para que o App Runner possa acessar os segredos e os parâmetros armazenados no Secrets Manager e no SSM Parameter Store. Para obter mais informações, consulte [Permissões do](#).
3. Faça referência segura aos segredos e parâmetros como variáveis de ambiente atribuindo um nome e fornecendo seu Amazon Resource Name (ARN). Você pode adicionar variáveis de ambiente ao [criar um serviço](#) ou [atualizar a configuração de um serviço](#). Você pode usar uma das seguintes opções para adicionar variáveis de ambiente:
 - Console App Runner
 - API do App Runner
 - Arquivo de configuração da `apprunner.yaml`

 Note

Você não pode atribuir `PORT` como nome uma variável de ambiente ao criar ou atualizar seu serviço App Runner. É uma variável de ambiente reservada para o serviço App Runner.

Para obter mais informações sobre como referenciar segredos e parâmetros, consulte [Gerenciamento de variáveis de ambiente](#).

 Note

Como o App Runner armazena apenas a referência ao segredo e ao parâmetro ARNs, os dados confidenciais não são visíveis para outras pessoas na configuração do serviço App Runner e nos registros do aplicativo.

Considerações

- Certifique-se de atualizar sua função de instância com as permissões apropriadas para acessar os segredos e os parâmetros no AWS Secrets Manager ou no AWS Systems Manager Parameter Store. Para obter mais informações, consulte [Permissões do](#).
- Certifique-se de que o AWS Systems Manager Parameter Store esteja no Conta da AWS mesmo serviço que você deseja iniciar ou atualizar. Atualmente, você não pode referenciar os parâmetros do SSM Parameter Store em todas as contas.
- Quando os segredos e os valores dos parâmetros são alternados ou alterados, eles não são atualizados automaticamente no seu serviço App Runner. Reimplante seu serviço App Runner, pois o App Runner só extrai segredos e parâmetros durante a implantação.
- Você também tem a opção de chamar AWS Secrets Manager diretamente o AWS Systems Manager Parameter Store por meio do SDK em seu serviço App Runner.
- Para evitar erros, certifique-se do seguinte ao referenciá-las como variáveis de ambiente:
 - Você especifica o ARN correto do segredo.
 - Você especifica o nome correto ou o ARN do parâmetro.

Permissões

Para habilitar a referência a segredos e parâmetros armazenados no AWS Secrets Manager ou SSM Parameter Store, adicione as permissões apropriadas à política do IAM do seu papel de instância para acessar o Secrets Manager e o SSM Parameter Store.

Note

O App Runner não pode acessar recursos em sua conta sem sua permissão. Você fornece a permissão atualizando sua política do IAM.

Você pode usar os seguintes modelos de política para atualizar sua função de instância no console do IAM. Você pode modificar esses modelos de política para atender às suas necessidades específicas. Para obter mais informações sobre como atualizar uma função de instância, consulte [Como modificar uma função](#) no Guia do usuário do IAM.

 Note

Você também pode copiar os seguintes modelos do console do App Runner ao [criar as variáveis de ambiente](#).

Copie o modelo a seguir para sua função de instância para adicionar permissão para referenciar segredos de AWS Secrets Manager.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt*"
      ],
      "Resource": [
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:<secret_name>",
        "arn:aws:kms:<region>:<aws_account_id>:key/<key_id>"
      ]
    }
  ]
}
```

Copie o modelo a seguir para sua função de instância para adicionar permissão aos parâmetros de referência do AWS Systems ManagerParameter Store.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter_name>"
      ]
    }
  ]
}
```

```
}
```

Gerenciando suas variáveis de ambiente

Gerencie as variáveis de ambiente do seu serviço App Runner usando um dos seguintes métodos:

- [the section called “Console do App Runner”](#)
- [the section called “API App Runner ou AWS CLI”](#)

Console do App Runner

Ao [criar um serviço](#) ou [atualizar um serviço](#) no console do App Runner, você pode adicionar variáveis de ambiente.

Adicionando variável de ambiente

Para adicionar uma variável de ambiente

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Com base no fato de você estar criando ou atualizando um serviço, execute uma das seguintes etapas:
 - Se você estiver criando um novo serviço, escolha Criar um serviço App Runner e acesse Configurar serviço.
 - Se você estiver atualizando um serviço existente, selecione o serviço que deseja atualizar e acesse a guia Configuração do serviço.
3. Vá para Variáveis de ambiente - opcional em Configurações de serviço.
4. Escolha qualquer uma das opções a seguir com base em sua necessidade:
 - Escolha Texto simples na fonte da variável de ambiente e insira seus pares de valores-chave em Nome da variável de ambiente e Valor da variável de ambiente, respectivamente.

Note

Escolha Texto sem formatação se quiser referenciar dados não confidenciais. Esses dados não são criptografados e são visíveis para outras pessoas na configuração do serviço App Runner e nos registros do aplicativo.

- Escolha Secrets Manager na fonte da variável de ambiente para referenciar o segredo armazenado AWS Secrets Manager como variável de ambiente em seu serviço. Forneça o nome da variável de ambiente e o Nome de recurso da Amazon (ARN) do segredo que você está referenciando em Nome da variável de ambiente e Valor da variável de ambiente, respectivamente.
- Escolha SSM Parameter Store na fonte da variável de ambiente para referenciar o parâmetro armazenado no SSM Parameter Store como variável de ambiente em seu serviço. Forneça o nome da variável de ambiente e o ARN do parâmetro que você está referenciando em Nome da variável de ambiente e Valor da variável de ambiente, respectivamente.

 Note

- Você não pode atribuir PORT como nome uma variável de ambiente ao criar ou atualizar seu serviço App Runner. É uma variável de ambiente reservada para o serviço App Runner.
- Se o parâmetro SSM Parameter Store estiver no Região da AWS mesmo serviço que você deseja iniciar, você pode especificar o Amazon Resource Name (ARN) completo ou o nome do parâmetro. Se o parâmetro estiver em uma região diferente, você precisará especificar o ARN completo.
- Verifique se o parâmetro ao qual você está se referindo está na mesma conta do serviço que você está lançando ou atualizando. Atualmente, você não pode referenciar o parâmetro SSM Parameter Store em todas as contas.

5. Escolha Adicionar variável de ambiente para fazer referência a outra variável de ambiente.
6. Expanda os modelos de política do IAM para visualizar e copiar os modelos de política do IAM fornecidos para o repositório de parâmetros do SSM AWS Secrets Manager e do SSM. Você só precisa fazer isso se ainda não tiver atualizado a política do IAM da sua função de instância com as permissões necessárias. Para obter mais informações, consulte [Permissões do](#).

Removendo a variável de ambiente

Antes de excluir uma variável de ambiente, certifique-se de que o código do aplicativo esteja atualizado para refletir a mesma. Se o código do aplicativo não for atualizado, seu serviço App Runner poderá falhar.

Para remover variáveis de ambiente

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Vá para a guia Configuração do serviço que você deseja atualizar.
3. Vá para Variáveis de ambiente - opcional em Configurações de serviço.
4. Escolha Remover ao lado da variável de ambiente que você deseja remover. Você recebe uma mensagem para confirmar a exclusão.
5. Escolha Excluir.

API App Runner ou AWS CLI

Você pode referenciar dados confidenciais armazenados no Secrets Manager e no SSM Parameter Store adicionando-os como variáveis de ambiente em seu serviço.

Note

Atualize a política do IAM da sua função de instância para que o App Runner possa acessar segredos e parâmetros armazenados no Secrets Manager e no SSM Parameter Store. Para obter mais informações, consulte [Permissões do](#) .

Para referenciar segredos e configurações como variáveis de ambiente

1. Crie um segredo ou configuração no Secrets Manager ou no SSM Parameter Store.

Os exemplos a seguir mostram como criar um segredo e um parâmetro usando o SSM Parameter Store.

Example Criando um segredo - Solicitação

O exemplo a seguir mostra como criar um segredo que represente a credencial do banco de dados.

```
aws secretsmanager create-secret \  
-name DevRdsCredentials \  
-description "Rds credentials for development account." \  
-secret-string "{\"user\":\"diegor\",\"password\":\"EXAMPLE-PASSWORD\"}"
```

Example Criando um segredo - Resposta

```
arn:aws:secretsmanager:<region>:<aws_account_id>:secret:DevRdsCredentials
```

Example Criando uma configuração - Solicitação

O exemplo a seguir mostra como criar um parâmetro que representa a cadeia de conexão do 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 Criando uma configuração - Resposta

```
arn:aws:ssm:<region>:<aws_account_id>:parameter/DevRdsConnectionString
```

2. Faça referência aos segredos e configurações que estão armazenados no Secrets Manager e no SSM Parameter Store adicionando-os como variáveis de ambiente. Você pode adicionar variáveis de ambiente ao criar ou atualizar seu serviço App Runner.

Os exemplos a seguir mostram como referenciar segredos e configurações como variáveis de ambiente em um serviço App Runner baseado em código e em imagem.

Example Arquivo.json de entrada para o serviço App Runner baseado em imagem

```
{
  "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 Serviço App Runner baseado em imagem — Solicitação

```

aws apprunner create-service \
--cli-input-json file://input.json

```

Example Serviço App Runner baseado em imagens — Resposta

```

{
  ...
  "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>"
  }
  ...
}

```

Exemplo Arquivo.json de entrada para o serviço App Runner baseado em código

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

Exemplo Serviço App Runner baseado em código — Solicitação

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

Exemplo Serviço App Runner baseado em código — Resposta

```
{
  ...
  "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:XXXXXXXXX",
            "Credential2" : "arn:aws:ssm:<region>:<aws_account_id>:parameter/
<parameter-name>"
          }
        }
      },
      "InstanceConfiguration": {
        "CPU": "1 vCPU",
        "Memory": "3 GB",
        "InstanceRoleArn": "<instance-role-arn>"
      }
    },
    ...
  }
```

3. O `apprunner.yaml` modelo é atualizado para refletir os segredos adicionados.

Veja a seguir um exemplo do `apprunner.yaml` modelo atualizado.

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"
```

Rede com o App Runner

Este capítulo descreve as configurações de rede para seus AWS App Runner serviços.

Neste capítulo, você aprenderá o seguinte:

- Como configurar seu tráfego de entrada para endpoints públicos e privados. Para obter mais informações, consulte [Configurando configurações de rede para tráfego de entrada](#).
- Como configurar seu tráfego de saída para acessar outros aplicativos em execução em uma Amazon VPC. Para obter mais informações, consulte [Habilitar o acesso à VPC para tráfego de saída](#).

Note

Atualmente, o App Runner oferece suporte ao tipo de endereço de pilha dupla (IPv4 e IPv6) somente para tráfego público de entrada. Somente IPv4 há suporte para tráfego de saída e tráfego de entrada privado.

Tópicos

- [Terminologia](#)
- [Definindo configurações de rede para tráfego de entrada](#)
- [Habilitando o acesso à VPC para tráfego de saída](#)

Terminologia

Para saber como personalizar seu tráfego de rede para atender às suas necessidades, vamos entender os seguintes termos usados neste capítulo.

Termos gerais

Para saber o que é necessário para se associar a uma Amazon Virtual Private Cloud (VPC), vamos entender os seguintes termos:

- VPC: Uma Amazon VPC é uma rede virtual logicamente isolada que oferece controle total sobre seu ambiente de rede virtual, incluindo posicionamento de recursos, conectividade e segurança. É

uma rede virtual que se assemelha muito a uma rede tradicional que você operaria em seu próprio data center.

- **Endpoint da interface VPC:** o endpoint da interface VPC, um recurso AWS PrivateLink , conecta uma VPC a um serviço de endpoint. Crie um endpoint de interface VPC para enviar tráfego para serviços de endpoint que usam um Network Load Balancer para distribuir tráfego. O tráfego destinado ao serviço de endpoint é resolvido usando DNS.
- **Regiões:** cada região é uma área geográfica separada onde você pode hospedar um serviço App Runner.
- **Zonas de disponibilidade:** uma zona de disponibilidade é um local isolado dentro de uma AWS região. É um ou mais data centers discretos com energia, rede e conectividade redundantes. As zonas de disponibilidade permitem tornar as aplicação em produção altamente disponíveis, tolerantes a falhas e escaláveis.
- **Sub-redes:** uma sub-rede é um intervalo de endereços IP em sua VPC. Cada sub-rede fica alocada em uma única zona de disponibilidade. Você pode iniciar um AWS recurso em uma sub-rede especificada. Use uma sub-rede pública para recursos que devem estar conectados à Internet e uma sub-rede privada para recursos que não estarão conectados à Internet.
- **Grupos de segurança:** um grupo de segurança controla o tráfego que pode alcançar e sair dos recursos aos quais está associado. Os grupos de segurança fornecem uma camada adicional de segurança para proteger os AWS recursos em cada sub-rede, oferecendo mais controle sobre o tráfego da rede. Quando você cria uma VPC, ela vem com um grupo de segurança padrão. Você pode criar grupos de segurança adicionais para cada VPC. Você pode associar um grupo de segurança somente aos recursos dentro da VPC para a qual ele foi criado.
- **Pilha dupla:** uma pilha dupla é um tipo de endereço que oferece suporte ao tráfego de rede tanto de endpoints IPv4 quanto IPv6 de terminais.

Termo específico para configurar o tráfego de saída

Conector VPC

Um conector VPC é um recurso do App Runner que permite que o serviço App Runner acesse aplicativos executados em uma Amazon VPC privada.

Termos específicos para configurar o tráfego de entrada

Para saber como você pode tornar seus serviços acessíveis de forma privada somente de dentro de uma Amazon VPC, vamos entender os seguintes termos:

- **Conexão de entrada de VPC:** a conexão de entrada de VPC é um recurso do App Runner que fornece um endpoint do App Runner para o tráfego de entrada. O App Runner atribui o recurso VPC Ingress Connection nos bastidores quando você escolhe um endpoint privado no console do App Runner para seu tráfego de entrada. O recurso VPC Ingress Connection conecta seu serviço App Runner ao endpoint da interface VPC da Amazon VPC.

 Note

Se você estiver usando a API App Runner, o recurso VPC Ingress Connection não será criado automaticamente.

- **Endpoint privado:** o endpoint privado é uma opção de console do App Runner que você seleciona para configurar o tráfego de rede de entrada para ser acessível somente de dentro de uma Amazon VPC.

Definindo configurações de rede para tráfego de entrada

Você pode configurar seu serviço para receber tráfego de entrada de um endpoint privado ou público.

Um endpoint público é a configuração padrão. Ele abre seu serviço para qualquer tráfego de entrada da Internet pública. Ele também oferece a flexibilidade de escolher entre o protocolo de Internet versão 4 (IPv4) ou o tipo de endereço de pilha dupla (IPv4 e IPv6) para seu serviço.

Um endpoint privado só permite que o tráfego de uma Amazon VPC acesse seu serviço App Runner. Isso é feito configurando um endpoint de interface VPC, um AWS PrivateLink recurso, para seu serviço App Runner. Dessa forma, criando uma conexão privada entre a Amazon VPC e seu serviço App Runner.

 Note

Atualmente, o App Runner oferece suporte ao tipo de endereço de pilha dupla (IPv4 e IPv6) somente para endpoint público. Somente para endpoint privado, IPv4 é suportado.

A seguir estão os tópicos abordados como parte da configuração de sua rede para tráfego de entrada:

- Como configurar seu tráfego de entrada para tornar seu serviço disponível de forma privada somente de dentro de uma Amazon VPC. Para obter mais informações, consulte [Habilitando endpoint privado para tráfego de entrada](#).
- Como configurar seu serviço para receber tráfego da Internet do tipo de endereço de pilha dupla. Para obter mais informações, consulte [Habilitando o dual stack para tráfego público de entrada](#).

Cabeçalhos

Com o App Runner, você pode acessar a fonte original IPv4 e IPv6 os endereços do tráfego que entra no seu aplicativo. Os endereços IP de origem originais são preservados atribuindo-lhes o cabeçalho da X-Forwarded-For solicitação. Isso permite que seus aplicativos busquem os endereços IP de origem originais quando necessário.

Note

Se seu serviço estiver configurado para usar um endpoint privado, o cabeçalho da X-Forwarded-For solicitação não poderá ser usado para acessar os endereços IP de origem originais. Se usado, ele recupera valores falsos.

Habilitando endpoint privado para tráfego de entrada

Por padrão, quando você cria um AWS App Runner serviço, o serviço pode ser acessado pela Internet. No entanto, você também pode tornar seu serviço App Runner privado e acessível somente de dentro de uma Amazon Virtual Private Cloud (Amazon VPC).

Com o serviço App Runner privado, você tem controle total sobre o tráfego de entrada, adicionando uma camada adicional de segurança. Isso é útil em vários casos de uso, incluindo a execução de aplicativos web corporativos internos APIs ou aplicativos que ainda estão em desenvolvimento e que exigem um nível maior de privacidade e segurança ou precisam atender a requisitos específicos de conformidade.

Note

[Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de grupo de segurança para endpoints privados em vez de WAF web. ACLs](#) Isso ocorre porque atualmente não oferecemos suporte ao

encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Como resultado, as regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP.

Para saber mais sobre segurança de infraestrutura e grupos de segurança, incluindo as melhores práticas, consulte os seguintes tópicos no Guia do usuário da Amazon VPC:

[Controle o tráfego de rede e Controle o tráfego para seus recursos da AWS usando grupos de segurança](#).

Quando seu serviço App Runner é privado, você pode acessá-lo de dentro de uma Amazon VPC. Não é necessário um gateway de internet, dispositivo NAT ou conexão VPN.

Note

Atualmente, o App Runner oferece suporte ao tipo de endereço de pilha dupla (IPv4 e IPv6) somente para tráfego público de entrada. Somente IPv4 há suporte para tráfego de saída e tráfego de entrada privado.

Considerações

- Antes de configurar um endpoint de interface VPC para o App Runner, leia as [considerações](#) no guia.AWS PrivateLink
- As políticas de VPC endpoint não são compatíveis com o App Runner. Por padrão, o acesso total ao App Runner é permitido por meio do endpoint da interface VPC. Como alternativa, você pode associar um grupo de segurança às interfaces de rede do endpoint para controlar o tráfego para o App Runner por meio do endpoint da interface VPC.
- [Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de grupo de segurança para endpoints privados em vez de WAF web. ACLs](#) Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Como resultado, as regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP.
- Depois de habilitar um endpoint privado, seu serviço só poderá ser acessado pela sua VPC e não poderá ser acessado pela Internet.

- Para maior disponibilidade, é recomendável selecionar pelo menos duas sub-redes na zona de disponibilidade diferentes para o endpoint da interface VPC. Não recomendamos usar apenas uma sub-rede.
- Você pode usar o mesmo endpoint da interface VPC para acessar vários serviços do App Runner em uma VPC.

Para obter informações sobre os termos usados nesta seção, consulte [Terminologia](#).

Permissões

A seguir está a lista de permissões necessárias para habilitar o endpoint privado:

- ec2: CreateTags
- ec2: CreateVpcEndpoint
- ec2: ModifyVpcEndpoint
- ec2: DeleteVpcEndpoints
- ec2: DescribeSubnets
- ec2: DescribeVpcEndpoints
- ec2: DescribeVpcs

Endpoint da interface VPC

Um endpoint de interface VPC é um AWS PrivateLink recurso que conecta uma Amazon VPC a um serviço de endpoint. Você pode especificar em qual Amazon VPC você gostaria que seu serviço App Runner estivesse acessível passando um endpoint de interface VPC. Para criar um endpoint de interface VPC, especifique o seguinte:

- A Amazon VPC para habilitar a conectividade.
- Adicione grupos de segurança. Por padrão, um grupo de segurança é atribuído ao endpoint da interface VPC. Você pode optar por associar um grupo de segurança personalizado para aumentar o controle do tráfego de entrada da rede.
- Adicione sub-redes. Para garantir maior disponibilidade, é recomendável selecionar pelo menos duas sub-redes para cada zona de disponibilidade a partir da qual você acessará o serviço App Runner. Um endpoint de interface de rede é criado em cada sub-rede que você habilita para o endpoint da interface VPC. Essas são interfaces de rede gerenciadas pelo solicitante que

servem como ponto de entrada para o tráfego destinado ao App Runner. Uma interface de rede gerenciada pelo solicitante é uma interface de rede que um serviço da AWS cria na sua VPC em seu nome.

- Se você estiver usando a API, adicione o endpoint da interface VPC do App Runner.
Servicename Por exemplo,

```
com.amazonaws.region.apprunner.requests
```

Você pode criar um endpoint de interface VPC usando um dos seguintes serviços: AWS

- Console App Runner. Para obter mais informações, consulte [Gerenciar endpoint privado](#).
- Console ou API do Amazon VPC e AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte [Acesso Serviços da AWS por meio AWS PrivateLink](#) do AWS PrivateLink Guia.

Note

Você é cobrado por cada endpoint de interface VPC usado com base nos preços.AWS PrivateLink Portanto, para melhor eficiência de custos, você pode usar o mesmo endpoint de interface VPC para acessar vários serviços do App Runner em uma VPC. No entanto, para um melhor isolamento, considere associar um endpoint de interface VPC diferente para cada um dos seus serviços do App Runner.

Conexão de ingresso da VPC

Uma conexão de entrada de VPC é um recurso do App Runner que especifica um endpoint do App Runner para o tráfego de entrada. O App Runner atribui o recurso VPC Ingress Connection nos bastidores quando você escolhe um endpoint privado no console do App Runner para seu tráfego de entrada. Escolha essa opção para permitir que somente o tráfego de uma Amazon VPC acesse seu serviço App Runner. O recurso VPC Ingress Connection conecta seu serviço App Runner ao endpoint da interface VPC da Amazon VPC. Você pode criar um recurso de conexão de entrada de VPC somente se estiver usando as operações de API para definir as configurações de rede para o tráfego de entrada. Para obter mais informações sobre como criar o recurso VPC Ingress Connection, consulte a Referência da [CreateVpcIngressConnection](#) AWS App Runner API.

Note

Um recurso de conexão de entrada de VPC do App Runner pode se conectar a um endpoint de interface de VPC da Amazon VPC. Além disso, você só pode criar um recurso de conexão de entrada de VPC para cada serviço do App Runner.

Endpoint privado

O endpoint privado é uma opção de console do App Runner que você pode escolher se quiser receber apenas tráfego de entrada de uma Amazon VPC. Escolher a opção de endpoint privado no console do App Runner oferece a opção de conectar seu serviço a uma VPC configurando seu endpoint de interface VPC. Nos bastidores, o App Runner atribui um recurso de conexão de entrada da VPC ao endpoint da interface da VPC que você configura.

Note

Somente o tráfego de IPv4 rede é compatível com o endpoint privado.

Resumo

Torne seu serviço privado permitindo que apenas o tráfego de uma Amazon VPC acesse seu serviço App Runner. Para conseguir isso, você cria um endpoint de interface VPC para a Amazon VPC selecionada usando o App Runner ou o Amazon VPC. No console do App Runner, você cria um endpoint de interface VPC ao ativar o endpoint privado para o tráfego de entrada. Em seguida, o App Runner cria automaticamente um recurso VPC Ingress Connection e se conecta ao endpoint da interface VPC e ao seu serviço App Runner. Isso cria uma conexão de serviço privada que garante que somente o tráfego da VPC selecionada possa acessar seu serviço App Runner.

Gerenciando endpoint privado

Gerencie o endpoint privado para o tráfego de entrada usando um dos seguintes métodos:

- [the section called “Console App Runner”](#)
- [the section called “API App Runner ou AWS CLI”](#)

 Note

Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de grupo de segurança para endpoints privados em vez de WAF web. ACLs Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Como resultado, as regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP.

Para saber mais sobre segurança de infraestrutura e grupos de segurança, incluindo as melhores práticas, consulte os seguintes tópicos no Guia do usuário da Amazon VPC: Controle o tráfego de rede e Controle o tráfego para seus recursos da AWS usando grupos de segurança.

Console App Runner

Ao criar um serviço usando o console do App Runner ou ao atualizar sua configuração posteriormente, você pode optar por configurar o tráfego de entrada.

Para configurar seu tráfego de entrada, escolha uma das opções a seguir.

- Endpoint público: para tornar seu serviço acessível a todos os serviços pela Internet. Por padrão, o endpoint público é selecionado.
- Endpoint privado: para tornar seu serviço App Runner acessível somente de dentro de uma Amazon VPC.

 Note

Atualmente, o App Runner oferece suporte IPv6 somente para endpoints públicos. IPv6 endpoints não são compatíveis com os serviços do App Runner hospedados em uma Amazon Virtual Private Cloud (Amazon VPC). Se você atualizar um serviço que usa um endpoint público de pilha dupla para um endpoint privado, seu serviço App Runner usará como padrão o tráfego somente de IPv4 endpoints e não receberá tráfego de endpoints. IPv6

Habilitar endpoint privado

Habilite um endpoint privado associando-o ao endpoint da interface VPC da Amazon VPC que você deseja acessar. Você pode criar um novo endpoint de interface VPC ou escolher um existente.

Para criar um endpoint de interface VPC

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Vá para a seção Rede em Configurar serviço.
3. Escolha Endpoint privado, para tráfego de rede de entrada. As opções para se conectar a uma VPC usando o endpoint da interface VPC são abertas.
4. Escolha Criar novo endpoint. A caixa de diálogo Create new VPC interface endpoint é aberta.
5. Insira um nome para o endpoint da interface VPC.
6. Escolha o endpoint de interface VPC necessário na lista suspensa disponível.
7. Escolha o grupo de segurança na lista suspensa. A adição de grupos de segurança fornece uma camada adicional de segurança ao endpoint da interface VPC. É recomendável escolher dois ou mais grupos de segurança. Se você não escolher um grupo de segurança, o App Runner atribuirá um grupo de segurança padrão ao endpoint da interface VPC. Certifique-se de que as regras do grupo de segurança não bloqueiem os recursos que desejam se comunicar com seu serviço App Runner. As regras do grupo de segurança devem permitir recursos que interajam com seu serviço App Runner.

Note

[Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de grupo de segurança para endpoints privados em vez de WAF web. ACLs](#) Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Como resultado, as regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP.

Para saber mais sobre segurança de infraestrutura e grupos de segurança, incluindo as melhores práticas, consulte os seguintes tópicos no Guia do usuário da Amazon VPC: [Controle o tráfego de rede e Controle o tráfego para seus recursos da AWS usando grupos de segurança](#).

8. Escolha as sub-redes necessárias na lista suspensa. É recomendável selecionar pelo menos duas sub-redes para cada zona de disponibilidade a partir da qual você acessará o serviço App Runner.
9. (Opcional) Escolha Adicionar nova tag e insira a chave da tag e o valor da tag.
10. Escolha Criar. A página Configurar serviço é aberta mostrando a mensagem de criação bem-sucedida do endpoint da interface VPC na barra superior.

Para escolher um endpoint de interface VPC existente

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Vá para a seção Rede em Configurar serviço.
3. Escolha Endpoint privado, para tráfego de rede de entrada. As opções para se conectar a uma VPC usando o endpoint da interface VPC são abertas. Uma lista dos endpoints de interface VPC disponíveis é mostrada.
4. Escolha o endpoint de interface VPC necessário listado em VPC interface endpoints.
5. Escolha Avançar para criar seu serviço. O App Runner ativa o endpoint privado.

 Note

Depois que seu serviço for criado, você poderá optar por editar os grupos de segurança e as sub-redes associados ao endpoint da interface VPC, se necessário.

Para verificar os detalhes do endpoint privado, acesse seu serviço e expanda a seção Rede na guia Configuração. Ele mostra detalhes da VPC e do endpoint da interface VPC associado ao endpoint privado.

Atualizar o endpoint da interface VPC

Depois que seu serviço App Runner for criado, você poderá editar o endpoint da interface VPC associado ao endpoint privado.

 Note

Você não pode atualizar o nome do endpoint e os campos da VPC.

Para atualizar o endpoint da interface VPC

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Acesse seu serviço e escolha Configurações de rede no painel esquerdo.
3. Escolha Tráfego de entrada para visualizar os endpoints da interface VPC associados aos respectivos serviços.
4. Escolha o endpoint da interface VPC que você deseja editar.
5. Selecione Editar. A caixa de diálogo para editar o endpoint da interface VPC é aberta.
6. Escolha os grupos de segurança e sub-redes necessários e clique em Atualizar. A página que mostra os detalhes do endpoint da interface VPC é aberta com a mensagem de atualização bem-sucedida do endpoint da interface VPC na barra superior.

Note

Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de grupo de segurança para endpoints privados em vez de WAF web. ACLs Isso ocorre porque atualmente não oferecemos suporte

ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Como resultado, as regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP.

Para saber mais sobre segurança de infraestrutura e grupos de segurança, incluindo as melhores práticas, consulte os seguintes tópicos no Guia do usuário da Amazon VPC:

[Controle o tráfego de rede e Controle o tráfego para seus recursos da AWS usando grupos de segurança.](#)

Excluir endpoint da interface VPC

Se você não quiser que seu serviço App Runner seja acessível de forma privada, você pode definir seu tráfego de entrada como Público. Mudar para Público remove o endpoint privado, mas não exclui o endpoint da interface VPC

Para excluir o endpoint da interface VPC

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Acesse seu serviço e escolha Configurações de rede no painel esquerdo.

3. Escolha Tráfego de entrada para visualizar os endpoints da interface VPC associados aos respectivos serviços.

 Note

Antes de excluir um endpoint da interface VPC, remova-o de todos os serviços aos quais ele está conectado atualizando seu serviço.

4. Escolha Excluir.

Se houver serviços conectados ao endpoint da interface VPC, você receberá a mensagem Não é possível excluir o endpoint da interface VPC. Se não houver serviços conectados ao endpoint da interface VPC, você receberá uma mensagem para confirmar a exclusão.

5. Escolha Excluir. A página de configurações de rede é aberta para o tráfego de entrada com a mensagem de exclusão bem-sucedida do endpoint da interface VPC na barra superior.

API App Runner ou AWS CLI

Você pode implantar um aplicativo no App Runner que só pode ser acessado de dentro de uma Amazon VPC.

Para obter informações sobre as permissões necessárias para tornar seu serviço privado, consulte [the section called “Permissões”](#).

 Note

Atualmente, o App Runner oferece suporte IPv6 somente para endpoints públicos. IPv6 endpoints não são compatíveis com os serviços do App Runner hospedados em uma Amazon Virtual Private Cloud (Amazon VPC). Se você atualizar um serviço que usa um endpoint público de pilha dupla para um endpoint privado, seu serviço App Runner usará como padrão o tráfego somente de IPv4 endpoints e não receberá tráfego de endpoints. IPv6

Para criar uma conexão de serviço privada com a Amazon VPC

1. Crie um endpoint de interface VPC, um AWS PrivateLink recurso, para se conectar ao App Runner. Para fazer isso, especifique sub-redes e grupos de segurança a serem associados ao aplicativo. Veja a seguir um exemplo de criação de um endpoint de interface VPC.

Note

Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de grupo de segurança para endpoints privados em vez de WAF web. ACLs. Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Como resultado, as regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP.

Para saber mais sobre segurança de infraestrutura e grupos de segurança, incluindo as melhores práticas, consulte os seguintes tópicos no Guia do usuário da Amazon VPC: [Controle o tráfego de rede e Controle o tráfego para seus recursos da AWS usando grupos de segurança](#).

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. Faça referência ao endpoint da interface VPC usando as ações da API [CreateService](#) ou do [UpdateService](#) App Runner por meio da CLI. Configure seu serviço para não ser acessível publicamente. `IsPubliclyAccessible` Defina como `False` no `IngressConfiguration` membro do `NetworkConfiguration` parâmetro. Veja a seguir um exemplo de referência ao endpoint da interface VPC.

Example

```
aws apprunner create-service
--network-configuration: ingress-configuration=<ingress_configuration>
--service-name: com.amazonaws.us-east-1.apprunner.requests
--source-configuration: <source_configuration>
# Ingress Configuration
{
  "IsPubliclyAccessible": False
```

```
}
```

3. Chame a ação da `create-vpc-ingress-connection` API para criar o recurso VPC Ingress Connection para o App Runner e associá-lo ao endpoint da interface VPC que você criou na etapa anterior. Ele retorna um nome de domínio usado para acessar seu serviço na VPC especificada. Veja a seguir um exemplo de criação de um recurso de conexão de entrada de VPC.

Example Solicitação

```
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 Resposta

```
{
  "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>
}
```

Atualizar a conexão de entrada VPC

Você pode atualizar o recurso VPC Ingress Connection. A conexão de entrada da VPC precisa estar em um dos seguintes estados para ser atualizada:

- DISPONÍVEL
- CRIAÇÃO_FALHADA
- ATUALIZAÇÃO COM FALHA

Veja a seguir um exemplo de atualização de um recurso do VPC Ingress Connection.

Example Solicitação

```
aws apprunner update-vpc-ingress-connection
  --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Resposta

```
{
  "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>
}
```

Excluir conexão de entrada de VPC

Você pode excluir o recurso VPC Ingress Connection se não precisar mais da conexão privada com a Amazon VPC.

A conexão de entrada da VPC deve estar em um dos seguintes estados para ser excluída:

- DISPONÍVEL
- FALHA NA CRIAÇÃO
- FALHA NA ATUALIZAÇÃO
- FALHA NA EXCLUSÃO

Veja a seguir um exemplo de exclusão de uma conexão de entrada de VPC

Example Solicitação

```
aws apprunner delete-vpc-ingress-connection
  --vpc-ingress-connection-arn: <vpc_ingress_connection_arn>
```

Example Resposta

```
{
```

```
"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>  
}
```

Use as seguintes ações da API App Runner para gerenciar o tráfego de entrada privado do seu serviço.

- [CreateVpcIngressConnection](#)— Crie um novo recurso de conexão de entrada de VPC. O App Runner exige esse recurso quando você deseja associar um serviço do App Runner a uma endpoint da Amazon VPC.
- [ListVpcIngressConnections](#)— Retorne uma lista dos endpoints do AWS App Runner VPC Ingress Connection associados à sua conta. AWS
- [DescribeVpcIngressConnection](#)— Retorne uma descrição completa do recurso AWS App Runner VPC Ingress Connection.
- [UpdateVpcIngressConnection](#)— Atualize o recurso AWS App Runner VPC Ingress Connection.
- [DeleteVpcIngressConnection](#)— Exclua um recurso do App Runner VPC Ingress Connection associado ao serviço App Runner.

Para obter mais informações sobre como usar a API App Runner, consulte o guia de [referência da API App Runner](#).

IPv6 Habilitando o tráfego público de entrada

Se você quiser que seu serviço receba tráfego de rede de entrada de IPv6 endereços ou de ambos os IPv4 IPv6 endereços, escolha o tipo de endereço Dual-Stack para o endpoint público. Ao criar um novo aplicativo, você pode encontrar essa configuração na seção Configurar serviço > Rede. Para obter mais informações sobre como ativar o IPv6 uso do console do App Runner ou da API do App Runner, consulte [the section called “Gerencie pilha dupla para endpoint público”](#)

Para obter mais informações sobre a adoção de IPv6 um AWS, consulte [IPv6 em. AWS](#)

O App Runner oferece suporte a pilha dupla somente para endpoints públicos do serviço App Runner. Somente IPv4 é compatível com todos os serviços privados do App Runner.

 Note

Quando você tem o tipo de endereço IP definido como Dual-Stack e altera sua configuração de rede de endpoint público para privado, o App Runner alterará automaticamente seu tipo de endereço para. IPv4 Isso ocorre porque o App Runner oferece suporte IPv6 somente para endpoints públicos.

Aprenda informações básicas sobre IPv4 vs IPv6

A camada IPv4 de rede, comumente usada para rotear o tráfego de rede pela Internet, usa um esquema de endereços de 32 bits. Esse espaço de endereço é limitado e pode ser esgotado com um grande número de dispositivos de rede. Por esse motivo, a Network Address Translation (NAT) é normalmente usada para rotear vários IPv4 endereços por meio de um único endereço de rede pública.

IPv6, uma versão mais recente do Internet Protocol, se baseia IPv4 e expande o espaço de endereço com um esquema de endereçamento de 128 bits. Com IPv6, você pode criar uma rede com um número quase ilimitado de dispositivos conectados. Devido à grande quantidade de endereços de rede, o NAT não é necessário para. IPv6

IPv4 e os IPv6 endpoints não são compatíveis entre si porque os IPv4 endpoints não podem receber IPv6 tráfego de entrada e vice-versa. A pilha dupla fornece uma solução conveniente, na qual tanto o tráfego IPv4 quanto o tráfego de IPv6 rede podem ser suportados simultaneamente.

Gerenciando pilha dupla para tráfego público de entrada

Gerencie o tipo de endereço de pilha dupla para tráfego público de entrada usando um dos seguintes métodos:

- [the section called “Console App Runner”](#)
- [the section called “API App Runner ou AWS CLI”](#)

Console App Runner

Você pode escolher o tipo de endereço de pilha dupla para o tráfego de entrada da Internet ao criar um serviço usando o console do App Runner ou ao atualizar sua configuração posteriormente.

Para habilitar o tipo de endereço de pilha dupla

1. Ao [criar](#) ou [atualizar](#) um serviço, expanda a seção Rede em Configurar serviço.
2. Escolha Endpoint público, para tráfego de rede de entrada. A opção de tipo de endereço IP do endpoint público é aberta.
3. Expanda Tipo de endereço IP de endpoint público para ver os seguintes tipos de endereço IP.
 - IPv4
 - Pilha dupla (e) IPv4 IPv6

Note

Se você não expandir o tipo de endereço IP do endpoint público para fazer uma seleção, o App Runner atribuirá IPv4 como configuração padrão.

4. Escolha Dual-stack (IPv4 e). IPv6
5. Escolha Avançar e, em seguida, Criar e implantar se estiver criando um serviço. Caso contrário, escolha Salvar alterações se você estiver atualizando um serviço.

Quando o serviço é implantado, seu aplicativo começa a receber tráfego de rede tanto de IPv6 endpoints IPv4 quanto de endpoints.

Note

Atualmente, o App Runner oferece suporte IPv6 somente para endpoints públicos. IPv6 endpoints não são compatíveis com os serviços do App Runner hospedados em uma Amazon Virtual Private Cloud (Amazon VPC). Se você atualizar um serviço que usa um endpoint público de pilha dupla para um endpoint privado, seu serviço App Runner usará como padrão o tráfego somente de IPv4 endpoints e não receberá tráfego de endpoints. IPv6

Para alterar o tipo de endereço

1. Siga as etapas para [atualizar](#) um serviço e navegar até Rede.
2. Navegue até Tipo de endereço IP do endpoint público em Tráfego de rede de entrada e selecione o tipo de endereço necessário.
3. Escolha Salvar alterações. Seu serviço é atualizado com sua seleção.

API App Runner ou AWS CLI

Ao chamar as ações da API [CreateService](#) ou [UpdateService](#) App Runner, use o `IpAddressType` membro do `NetworkConfiguration` parâmetro para especificar o tipo de endereço. Os valores suportados que você pode especificar são `IPv4` `DUAL_STACK` e. Especifique `DUAL_STACK` se você deseja que seu serviço receba tráfego da Internet de IPv4 e IPv6 endpoints. Se você não especificar nenhum valor para `IpAddressType`, por padrão, será `IPv4` aplicado.

Veja a seguir um exemplo para criar um serviço com a pilha dupla como endereço IP. Este exemplo chama um `input.json` arquivo.

Example Solicitação para criar um serviço com suporte de pilha dupla

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

Example Conteúdo 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 Resposta

```
{
  "Service": {
    "ServiceName": "example-service",
    "ServiceId": "<service-id>",
    "ServiceArn": "arn:aws:apprunner:us-east-2:123456789012:service/example-
service/<service-id>",
    "ServiceUrl": "1234567890.us-east-2.awsapprunner.com",
    "CreatedAt": "2023-10-16T12:30:51.724000-04:00",
    "UpdatedAt": "2023-10-16T12:30:51.724000-04:00",
    "Status": "OPERATION_IN_PROGRESS",
    "SourceConfiguration": {
      "ImageRepository": {
        "ImageIdentifier": "public.ecr.aws/aws-containers/hello-app-runner:latest",
        "ImageConfiguration": {
          "Port": "8000"
        },
        "ImageRepositoryType": "ECR_PUBLIC"
      },
      "AutoDeploymentsEnabled": false
    },
    "InstanceConfiguration": {
      "Cpu": "1024",
      "Memory": "2048"
    },
    "HealthCheckConfiguration": {
      "Protocol": "TCP",
      "Path": "/",
      "Interval": 5,
      "Timeout": 2,
      "HealthyThreshold": 1,
      "UnhealthyThreshold": 5
    },
    "AutoScalingConfigurationSummary": {
      "AutoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
DefaultConfiguration/1/00000000000000000000000000000001",
      "AutoScalingConfigurationName": "DefaultConfiguration",
      "AutoScalingConfigurationRevision": 1
    },
    "NetworkConfiguration": {
      "IpAddressType": "DUAL_STACK",
      "EgressConfiguration": {
```

```
    "EgressType": "DEFAULT"
  },
  "IngressConfiguration": {
    "IsPubliclyAccessible": true
  }
},
"OperationId": "24bd100b1e111ae1a1f0e1115c4f11de"
}
```

Note

Atualmente, o App Runner oferece suporte IPv6 somente para endpoints públicos. IPv6 endpoints não são compatíveis com os serviços do App Runner hospedados em uma Amazon Virtual Private Cloud (Amazon VPC). Se você atualizar um serviço que usa um endpoint público de pilha dupla para um endpoint privado, seu serviço App Runner usará como padrão o tráfego somente de IPv4 endpoints e não receberá tráfego de endpoints. IPv6

Para obter mais informações sobre o parâmetro da API, consulte [NetworkConfiguration](#).

Habilitando o acesso à VPC para tráfego de saída

Por padrão, seu AWS App Runner aplicativo pode enviar mensagens para endpoints públicos. Isso inclui suas próprias soluções e qualquer outro site público ou serviço da web. Serviços da AWS Seu aplicativo pode até mesmo enviar mensagens para endpoints públicos de aplicativos que são executados em uma VPC a [partir da Amazon Virtual Private Cloud](#) (Amazon VPC). Se você não configurar uma VPC ao iniciar seu ambiente, o App Runner usará a VPC padrão, que é pública.

Você pode optar por iniciar seu ambiente em uma VPC personalizada para personalizar as configurações de rede e segurança para o tráfego de saída. Você pode habilitar seu AWS App Runner serviço para acessar aplicativos que são executados em uma VPC privada a partir da Amazon Virtual Private Cloud (Amazon VPC). Depois de fazer isso, seu aplicativo pode se conectar e enviar mensagens para outros aplicativos hospedados em uma [Amazon Virtual Private Cloud](#) (Amazon VPC). Exemplos são um banco de dados Amazon RDS ElastiCache, Amazon e outros serviços privados hospedados em uma VPC privada.

Conector VPC

Você pode associar seu serviço a uma VPC criando um VPC endpoint a partir do console do App Runner, chamado VPC Connector. Para criar um conector VPC, especifique a VPC, uma ou mais sub-redes e, opcionalmente, um ou mais grupos de segurança. Depois de configurar um conector VPC, você pode usá-lo com um ou mais serviços do App Runner.

Latência única

Se você configurar seu serviço App Runner com um conector VPC personalizado para tráfego de saída, ele poderá ter uma latência de inicialização única de dois a cinco minutos. O processo de inicialização espera até que o conector VPC esteja pronto para se conectar a outros recursos antes de definir o status do serviço como Executando. Você pode configurar um serviço com um conector VPC personalizado ao criá-lo pela primeira vez ou pode fazer isso depois, fazendo uma atualização do serviço.

Observe que, se você reutilizar a mesma configuração do conector VPC para outro serviço, não haverá latência. A configuração do conector VPC é baseada na combinação de grupo de segurança e sub-rede. Para uma determinada configuração de conector VPC, a latência só acontece uma vez, durante a criação inicial do hiperplano ENIs do conector VPC (interfaces de rede elástica).

Mais sobre conectores VPC personalizados e Hyperplane AWS

[Os conectores VPC no App Runner são baseados no AWS Hyperplane, o sistema de rede interno da Amazon que está por trás de vários AWS recursos, como Network Load Balancer, NAT Gateway e AWS. PrivateLink](#) A tecnologia AWS Hyperplane fornece recursos de alto rendimento e baixa latência, além de um maior grau de compartilhamento. Uma ENI de hiperplano é criada em suas sub-redes quando você cria um conector VPC e o associa ao seu serviço. A configuração do conector VPC é baseada em uma combinação de grupo de segurança e sub-rede, e você pode referenciar o mesmo conector VPC em vários serviços do App Runner. Como resultado, o Hyperplane subjacente ENIs é compartilhado entre os serviços do App Runner. Esse compartilhamento é viável, mesmo quando você aumenta o número de tarefas necessárias para lidar com a carga da solicitação e resulta em uma utilização mais eficiente do espaço IP em sua VPC. Para obter mais informações, consulte Análise [aprofundada sobre a rede VPC do AWS App Runner](#) no blog do AWS Container.

Sub-rede

Cada sub-rede está em uma zona de disponibilidade específica. Para alta disponibilidade, recomendamos que você selecione sub-redes em pelo menos três zonas de disponibilidade. Se a

região tiver menos de três zonas de disponibilidade, recomendamos que você selecione suas sub-redes em todas as zonas de disponibilidade suportadas.

Ao selecionar uma sub-rede para sua VPC, certifique-se de escolher uma sub-rede privada, não uma sub-rede pública. Isso ocorre porque, quando você cria um conector VPC, o serviço App Runner cria um Hyperplane ENI em cada uma das sub-redes. Cada Hyperplane ENI recebe apenas um endereço IP privado e é marcado com uma tag da `AWSAppRunnerManagedchave`. Se você escolher uma sub-rede pública, ocorrerão erros ao executar o serviço App Runner. No entanto, se seu serviço precisar acessar alguns serviços que estão na Internet ou em outros serviços públicos Serviços da AWS, consulte [the section called “Considerações ao selecionar uma sub-rede”](#).

Considerações ao selecionar uma sub-rede

- Quando você conecta seu serviço a uma VPC, o tráfego de saída não tem acesso à Internet pública. Todo o tráfego de saída do seu aplicativo é direcionado pela VPC à qual seu serviço está conectado. Todas as regras de rede da VPC se aplicam ao tráfego de saída do seu aplicativo. Isso significa que seus serviços não podem acessar a Internet pública AWS APIs e. Para obter acesso, faça o seguinte:
 - Conecte as sub-redes à Internet por meio de um gateway [NAT](#).
 - Configure [endpoints VPC para os](#) Serviços da AWS que você deseja acessar. Seu serviço permanece dentro da Amazon VPC usando AWS PrivateLink
- Algumas zonas de disponibilidade em algumas Regiões da AWS não oferecem suporte às sub-redes que podem ser usadas com os serviços do App Runner. Se você escolher sub-redes nessas zonas de disponibilidade, seu serviço não será criado ou atualizado. Para essas situações, o App Runner fornece uma mensagem de erro detalhada apontando para as sub-redes e zonas de disponibilidade não suportadas. Quando isso ocorrer, solucione o problema removendo as sub-redes não suportadas da sua solicitação e tente novamente.

Grupo de segurança

Opcionalmente, você pode especificar os grupos de segurança que o App Runner usa para acessar AWS nas sub-redes especificadas. Se você não especificar grupos de segurança, o App Runner usa o grupo de segurança padrão da VPC. O grupo de segurança padrão permite todo o tráfego de saída.

Adicionar um grupo de segurança fornece uma camada adicional de segurança aos conectores VCP, oferecendo mais controle sobre o tráfego da rede. O conector VPC é usado somente para

comunicação externa do seu aplicativo. Você usa regras de saída para permitir a comunicação com os endpoints de destino desejados. Você também deve garantir que todos os grupos de segurança associados ao recurso de destino tenham as regras de entrada apropriadas. Caso contrário, esses recursos não aceitarão tráfego proveniente dos grupos de segurança do VPC Connector.

Note

Quando você associa seu serviço a uma VPC, o tráfego a seguir não é afetado:

- Tráfego de entrada — As mensagens recebidas pelo seu aplicativo não são afetadas por uma VPC associada. As mensagens são roteadas por meio do nome de domínio público associado ao seu serviço e não interagem com a VPC.
- Tráfego do App Runner — O App Runner gerencia várias ações em seu nome, como extrair código-fonte e imagens, enviar registros e recuperar segredos. O tráfego que essas ações geram não é roteado pela sua VPC.

Para saber mais sobre como AWS App Runner se integra à Amazon VPC, [AWS consulte App Runner VPC Networking](#).

Note

Para tráfego de saída, o App Runner atualmente suporta apenas IPv4

Gerenciar o acesso à VPC

Note

Se você criar um conector VPC de tráfego de saída para um serviço, o processo de inicialização do serviço a seguir terá uma latência única. Você pode definir essa configuração para um novo serviço ao criá-lo, ou posteriormente, com uma atualização de serviço. Para obter mais informações, consulte [Latência única](#) o capítulo Networking with App Runner deste guia.

Gerencie o acesso à VPC para seus serviços do App Runner usando um dos seguintes métodos:

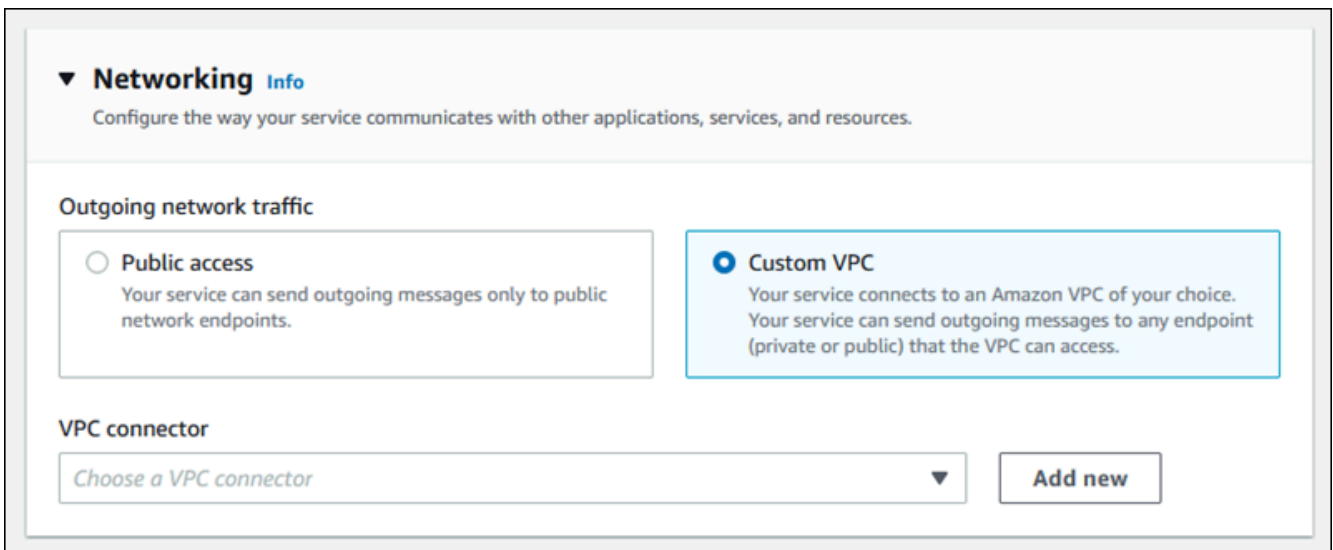
App Runner console

Ao [criar um serviço](#) usando o console do App Runner ou ao [atualizar sua configuração posteriormente](#), você pode optar por configurar seu tráfego de saída. Procure a seção Configuração de rede na página do console. Para tráfego de rede de saída, escolha uma das seguintes opções:

- Acesso público: para associar seu serviço a endpoints públicos de outros Serviços da AWS.
- VPC personalizada: para associar seu serviço a uma VPC da Amazon VPC. Seu aplicativo pode se conectar e enviar mensagens para outros aplicativos hospedados em uma Amazon VPC.

Para habilitar a VPC personalizada

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Vá para a seção Rede em Configurar serviço.



3. Escolha VPC personalizada para tráfego de rede de saída.
4. No painel de navegação, escolha Conector VPC.

Se você criou os conectores VPC, o console exibirá uma lista de conectores VPC na sua conta. Você pode escolher um conector VPC existente e escolher Avançar para revisar sua configuração. Em seguida, vá para a última etapa. Como alternativa, você pode adicionar um novo conector VPC usando as etapas a seguir.

5. Escolha Adicionar novo para criar um novo conector VPC para seu serviço.

Em seguida, a caixa de diálogo Adicionar novo conector VPC é aberta.

Add new VPC connector ✕

You can share a VPC connector with other App Runner services in your account.

VPC connector name

my-vpc-two-subnets

VPC

To create a new VPC visit [Amazon VPC](#)

vpc-5732152e (.0/16) ▼ ↻

Subnets

Choose one or more subnets ▼ ↻

subnet-3d337367 (.0/20) us-east-1b ✕

subnet-04575a4c (.0/20) us-east-1a ✕

Security groups

Choose one or more security groups ▼ ↻

sg-11c24e61 (default) ✕

Tags — *optional*

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

No tags associated with the resource.

Add new tag

You can add 50 more tags.

Cancel Add

- Insira um nome para seu conector VPC e selecione o VPC necessário na lista disponível.

7. Para Sub-redes, selecione uma sub-rede para cada zona de disponibilidade a partir da qual você planeja acessar o serviço App Runner. Para melhor disponibilidade, escolha três sub-redes. Ou, se houver menos de três sub-redes, escolha todas as sub-redes disponíveis.

 Note

Certifique-se de atribuir sub-redes privadas ao conector VPC. Se você atribuir sub-redes públicas ao conector VPC, seu serviço não será criado ou revertido automaticamente durante uma atualização.

8. (Opcional) Em Grupo de segurança, selecione os grupos de segurança a serem associados às interfaces de rede do endpoint.
9. (Opcional) Para adicionar uma tag, escolha Adicionar nova tag e insira a chave e o valor da tag.
10. Escolha Adicionar.

Os detalhes do conector VPC que você criou aparecem em Conector VPC.

11. Escolha Avançar para revisar sua configuração e, em seguida, escolha Criar e implantar.

O App Runner cria um recurso de conector VPC para você e o associa ao seu serviço. Se o serviço for criado com sucesso, o console mostrará o painel do serviço, com uma visão geral do serviço novo.

App Runner API or AWS CLI

Ao chamar as ações da API [CreateService](#) ou [UpdateService](#) App Runner, use o `EgressConfiguration` membro do `NetworkConfiguration` parâmetro para especificar um recurso de conector VPC para seu serviço.

Use as seguintes ações da API App Runner para gerenciar seus recursos do VPC Connector.

- [CreateVpcConnector](#)— Cria um novo conector VPC.
- [ListVpcConnectors](#)— Retorna uma lista dos conectores VPC associados ao seu. Conta da AWS A lista inclui descrições completas.
- [DescribeVpcConnector](#)— Retorna uma descrição completa de um conector VPC.
- [DeleteVpcConnector](#)— Exclui um conector VPC. Se você atingir a cota de conectores VPC para o seu Conta da AWS, talvez seja necessário excluir conectores VPC desnecessários.

Para implantar um aplicativo no App Runner que tenha acesso de saída a uma VPC, primeiro você deve criar um conector VPC. Você pode fazer isso especificando uma ou mais sub-redes e grupos de segurança para associar ao aplicativo. Em seguida, você pode referenciar o conector VPC no Create ou UpdateService por meio da CLI, conforme ilustrado no exemplo a seguir:

```
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
```

Observabilidade para seu serviço App Runner

AWS App Runner se integra a vários AWS serviços para fornecer um amplo conjunto de ferramentas de observabilidade para seu serviço App Runner. Os tópicos deste capítulo descrevem esses recursos.

Tópicos

- [Rastreado a atividade do serviço App Runner](#)
- [Visualizando registros do App Runner transmitidos para o Logs CloudWatch](#)
- [Visualizando as métricas de serviço do App Runner reportadas para CloudWatch](#)
- [Gerenciando eventos do App Runner em EventBridge](#)
- [Registrando chamadas da API App Runner com AWS CloudTrail](#)
- [Rastreamento para seu aplicativo App Runner com X-Ray](#)

Rastreado a atividade do serviço App Runner

AWS App Runner usa uma lista de operações para acompanhar as atividades em seu serviço App Runner. Uma operação representa uma chamada assíncrona para uma ação de API, como criar um serviço, atualizar uma configuração e implantar um serviço. As seções a seguir mostram como monitorar a atividade no console do App Runner e usar a API.

Rastreie a atividade do serviço App Runner

Monitore a atividade do serviço App Runner usando um dos seguintes métodos:

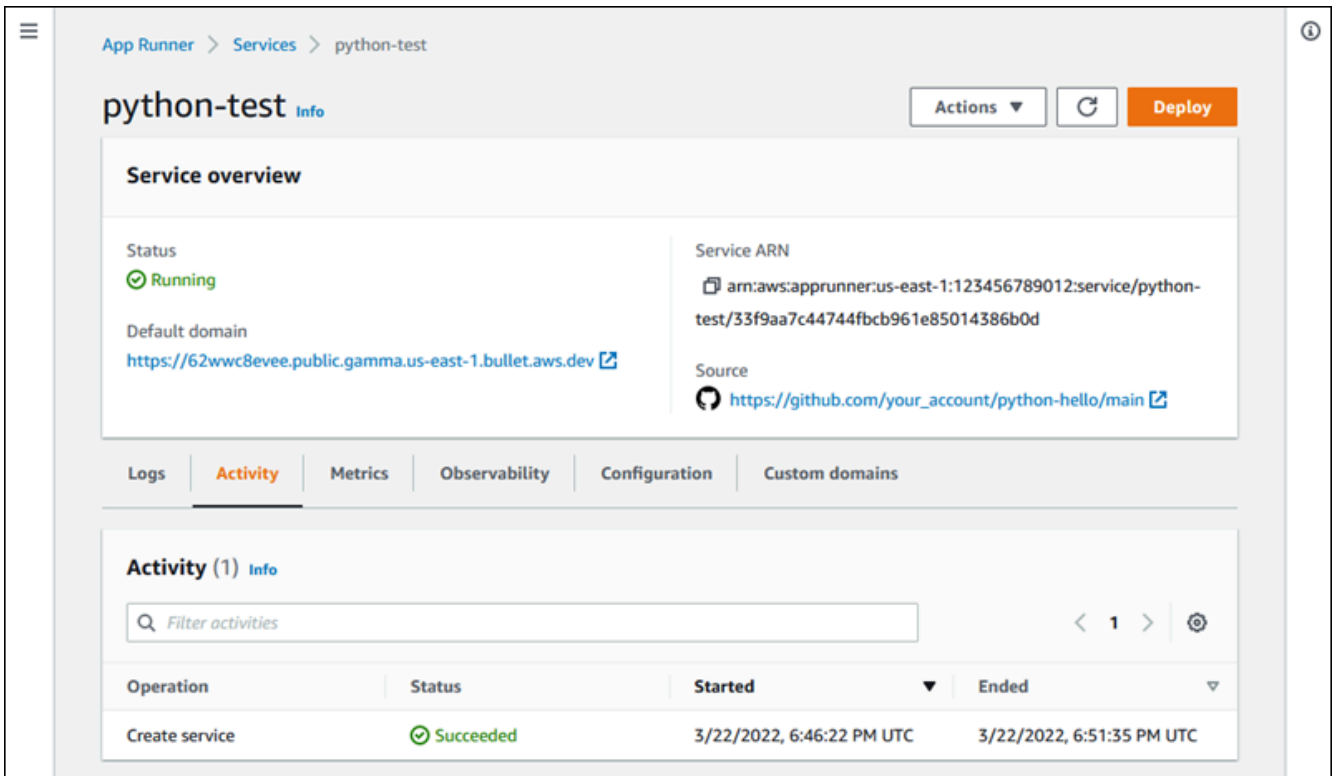
App Runner console

O console do App Runner exibe sua atividade de serviço do App Runner e fornece mais maneiras de explorar as operações.

Para visualizar a atividade do seu serviço

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Na página do painel do serviço, escolha a guia Atividade, se ela ainda não tiver sido escolhida.

O console exibe uma lista de operações.

4. Para encontrar operações específicas, examine a lista inserindo um termo de pesquisa. Você pode pesquisar qualquer valor que apareça na tabela.
5. Escolha qualquer operação listada para ver ou baixar o registro relacionado.

App Runner API or AWS CLI

A [ListOperations](#) ação, dada o Amazon Resource Name (ARN) de um serviço App Runner, retorna uma lista das operações que ocorreram nesse serviço. Cada item da lista contém um ID de operação e alguns detalhes de rastreamento.

Visualizando registros do App Runner transmitidos para o Logs CloudWatch

Você pode usar o Amazon CloudWatch Logs para monitorar, armazenar e acessar arquivos de log gerados por seus recursos em vários AWS serviços. Para obter mais informações, consulte o [Guia do usuário do Amazon CloudWatch Logs](#).

AWS App Runner coleta a saída das implantações do seu aplicativo e do seu serviço ativo e a transmite para o Logs. CloudWatch As seções a seguir listam os fluxos de log do App Runner e mostram como visualizá-los no console do App Runner.

Grupos e streams de registros do App Runner

CloudWatch Os registros mantêm os dados de registro em fluxos de registro que são organizados posteriormente em grupos de registros. Um fluxo de log é uma sequência de eventos de log de uma fonte específica. Um grupo de logs é um grupo de fluxos de log que compartilham as mesmas configurações de retenção, monitoramento e controle de acesso.

O App Runner define dois grupos de CloudWatch registros de registros, cada um com vários fluxos de registros, para cada serviço do App Runner em seu. Conta da AWS

Registros de serviço

O grupo de registros de serviço contém a saída de registro gerada pelo App Runner à medida que ele gerencia seu serviço App Runner e age sobre ele.

Nome do grupo de logs	Exemplo
/aws/apprunner/ <i>service-name</i> / <i>service-id</i> /service	/aws/apprunner/python-test/ ac7ec8b51ff34746bcb6654e0bc b23da/service

Dentro do grupo de registros de serviços, o App Runner cria um fluxo de registros de eventos para capturar atividades no ciclo de vida do seu serviço App Runner. Por exemplo, isso pode ser iniciar seu aplicativo ou pausá-lo.

Além disso, o App Runner cria um fluxo de log para cada operação assíncrona de longa duração relacionada ao seu serviço. O nome do fluxo de log reflete o tipo de operação e o ID específico da operação.

Uma implantação é um tipo de operação. Os registros de implantação contêm a saída de registro das etapas de criação e implantação que o App Runner executa quando você cria um serviço ou implanta uma nova versão do seu aplicativo. Os nomes do fluxo de log de implantação começam com `deployment/` e terminam com o ID da operação que executa a implantação. Essa operação é uma [CreateService](#) chamada para a implantação inicial do aplicativo ou uma [StartDeployment](#) chamada para cada implantação posterior.

Em um log de implantação, cada mensagem de log começa com um prefixo:

- [AppRunner]— Saída que o App Runner gera durante a implantação.
- [Build]— Saída de seus próprios scripts de construção.

Nome do fluxo de logs	Exemplo
events	N/A (nome fixo)
<i>operation-type /operation-id</i>	deployment/c2c8eeedea164f459cf78f12a8953390

Logs de aplicações

O grupo de registros do aplicativo contém a saída do código do aplicativo em execução.

Nome do grupo de logs	Exemplo
/aws/apprunner/ <i>service-name</i> / <i>service-id</i> /application	/aws/apprunner/python-test/ac7ec8b51ff34746bcb6654e0bcb23da/application

Dentro do grupo de registros do aplicativo, o App Runner cria um fluxo de registros para cada instância (unidade de escalabilidade) que está executando seu aplicativo.

Nome do fluxo de logs	Exemplo
instance/ <i>instance-id</i>	instance/1a80bc9134a84699b7 b3432ebee5b591

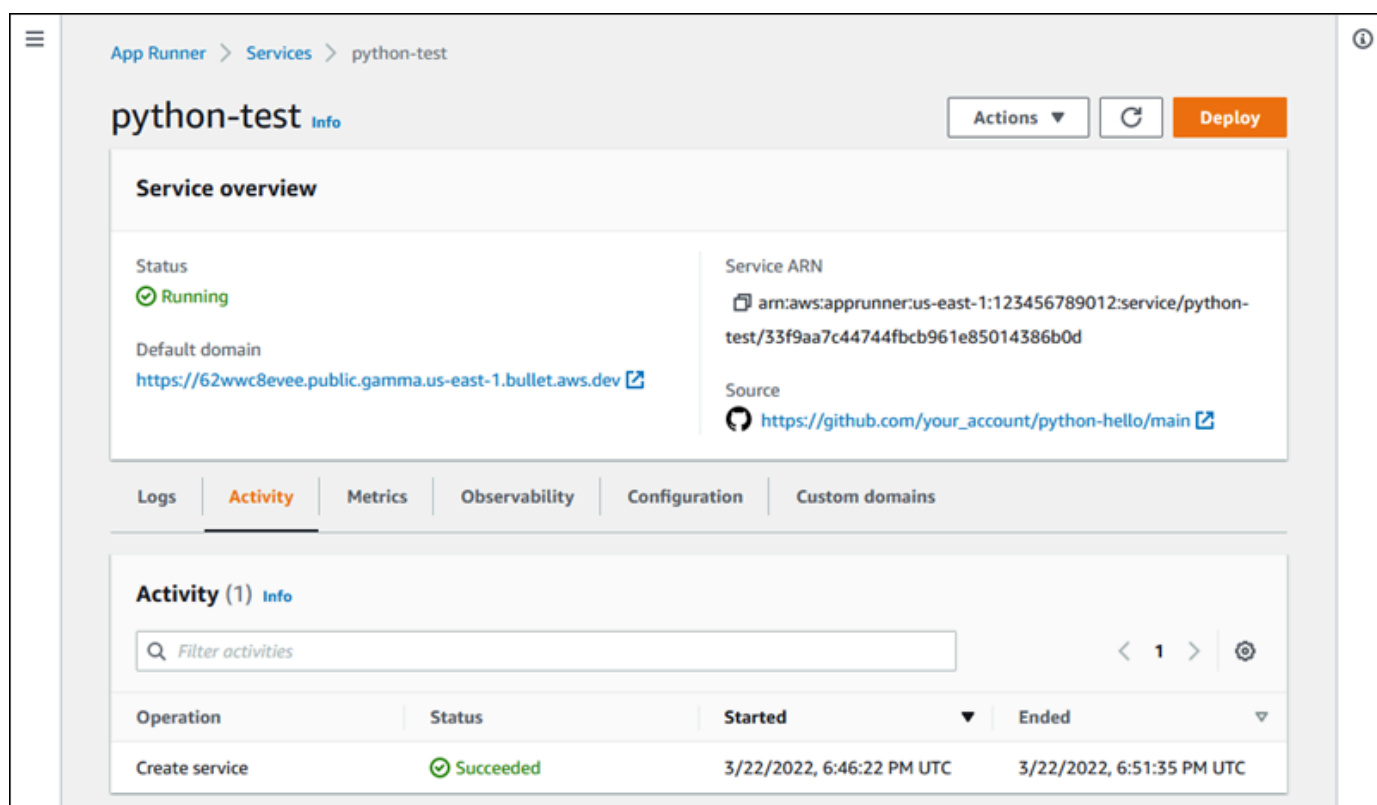
Visualizando os registros do App Runner no console

O console do App Runner exibe um resumo de todos os registros do seu serviço e permite que você os visualize, explore e baixe.

Para ver os registros do seu serviço

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Na página do painel do serviço, escolha a guia Registros.

O console exibe alguns tipos de registros em várias seções:

- Registro de eventos — Atividade no ciclo de vida do seu serviço App Runner. O console exibe os eventos mais recentes.
- Registros de implantação — forneça implantações de repositórios para seu serviço App Runner. O console exibe um fluxo de log separado para cada implantação.
- Registros do aplicativo — A saída do aplicativo web que é implantado no seu serviço App Runner. O console combina a saída de todas as instâncias em execução em um único fluxo de registros.

The screenshot displays the AWS App Runner console interface. It features three main sections: 'Event log', 'Deployment logs', and 'Application logs'. The 'Event log' section shows a list of events with timestamps and descriptions. The 'Deployment logs' section includes a search bar and a table with columns for Operation, Status, Started, and Ended. The 'Application logs' section shows a list of logs with columns for Name and Last written.

Event log Info

1 2020-09-24T14:21:40.879-07:00 Build service started
 2 2020-09-24T14:21:40.879-07:00 Build service completed
 3 2020-09-24T14:21:40.879-07:00 my-web-service1 server running
 4 2020-09-24T14:21:40.879-07:00 Deploying service
 5
 6
 7
 8
 9
 10

Deployment logs (1) Info

Find deployment

Operation	Status	Started	Ended
Automatic deployment	In progress	12/21/2020, 2:30:31 PM UTC	—

Application logs Info

Name	Last written
Application logs	12/21/2020, 2:30:31 PM UTC

4. Para encontrar implantações específicas, examine a lista de registros de implantação inserindo um termo de pesquisa. Você pode pesquisar qualquer valor que apareça na tabela.
5. Para visualizar o conteúdo de um registro, escolha Exibir registro completo (registro de eventos) ou o nome do fluxo de registros (registros de implantação e aplicativos).
6. Escolha Baixar para baixar um registro. Para um fluxo de registros de implantação, selecione primeiro um fluxo de registros.
7. Escolha Visualizar em CloudWatch para abrir o CloudWatch console e usar seus recursos completos para explorar seus registros de serviço do App Runner. Para um fluxo de registros de implantação, selecione primeiro um fluxo de registros.

 Note

O CloudWatch console é particularmente útil se você quiser visualizar os registros do aplicativo de instâncias específicas em vez do registro combinado do aplicativo.

Visualizando as métricas de serviço do App Runner reportadas para CloudWatch

A Amazon CloudWatch monitora seus recursos da Amazon Web Services (AWS) e os aplicativos em que você executa AWS em tempo real. Você pode usar CloudWatch para coletar e monitorar métricas, que são variáveis que você pode medir para seus recursos e aplicativos. Você também pode usá-lo para criar alarmes que monitorem as métricas. Quando um determinado limite é atingido, CloudWatch envia notificações ou faz alterações automaticamente nos recursos monitorados. Para obter mais informações, consulte o [Guia CloudWatch do usuário da Amazon](#).

AWS App Runner coleta uma variedade de métricas que fornecem maior visibilidade sobre o uso, o desempenho e a disponibilidade dos seus serviços do App Runner. Algumas métricas rastreiam instâncias individuais que executam seu serviço web, enquanto outras estão no nível geral do serviço. As seções a seguir listam as métricas do App Runner e mostram como visualizá-las no console do App Runner.

Métricas do App Runner

O App Runner coleta as seguintes métricas relacionadas ao seu serviço e as publica CloudWatch no namespace. `AWS/AppRunner`

 Note

Antes de 23 de agosto de 2023, as métricas de utilização da CPU e da memória eram baseadas em unidades de vCPU e megabytes de memória utilizados, em vez da porcentagem de utilização, conforme calculado hoje. Se seu aplicativo foi executado no App Runner antes dessa data e você optar por voltar a ver as métricas dessa data no App Runner ou no CloudWatch console, você verá uma exibição das métricas em ambas as unidades e também verá algumas irregularidades como resultado.

⚠ Important

Você precisará atualizar todos CloudWatch os alarmes baseados nos valores da métrica de utilização da CPU e da memória antes de 23 de agosto de 2023. Atualize os alarmes para serem acionados com base na porcentagem de utilização, em vez de vCPU ou megabytes. Para obter mais informações, consulte o [Guia CloudWatch do usuário da Amazon](#).

As métricas em nível de instância são coletadas para cada instância (unidade de escalabilidade) individualmente.

O que é medido?	Métrica	Descrição
CPU utilization	CPUUtilization	A porcentagem do uso médio da CPU durante períodos de um minuto do uso total da CPU reservado pela configuração do serviço.
Memory utilization	MemoryUtilization	A porcentagem do uso médio de memória durante períodos de um minuto do total de memória reservada pela configuração do serviço.

As métricas de nível de serviço são coletadas para todo o serviço.

O que é medido?	Métrica	Descrição
CPU utilization	CPUUtilization	A porcentagem do uso agregado da CPU em todas as instâncias durante períodos de um minuto do uso total da CPU reservado pela configuração do serviço.
Memory utilization	MemoryUtilization	A porcentagem de uso de memória agregada em todas as instâncias durante períodos de um minuto do total de memória reservada pela configuração do serviço.
Concurrency	Concurrency	O número aproximado de solicitações simultâneas que estão sendo tratadas pelo serviço.

O que é medido?	Métrica	Descrição
HTTP request count	Requests	O número de solicitações HTTP que o serviço recebeu.
HTTP status counts	2xxStatus Responses 4xxStatus Responses 5xxStatus Responses	O número de solicitações HTTP que retornaram cada status de resposta, agrupadas por categoria (2XX, 4XX, 5XX).
HTTP request latency	RequestLatency	O tempo, em milissegundos, que seu serviço web levou para processar solicitações HTTP.
Instance counts	ActiveInstances	O número de instâncias que estão processando solicitações HTTP para seu serviço.

 **Note**
Se a `ActiveInstances` métrica exibir zero, isso significa que não há solicitações para o serviço. Isso não indica que o número de instâncias do seu serviço seja zero.

Visualizando métricas do App Runner no console

O console do App Runner exibe graficamente as métricas que o App Runner coleta para seu serviço e fornece mais maneiras de explorá-las.

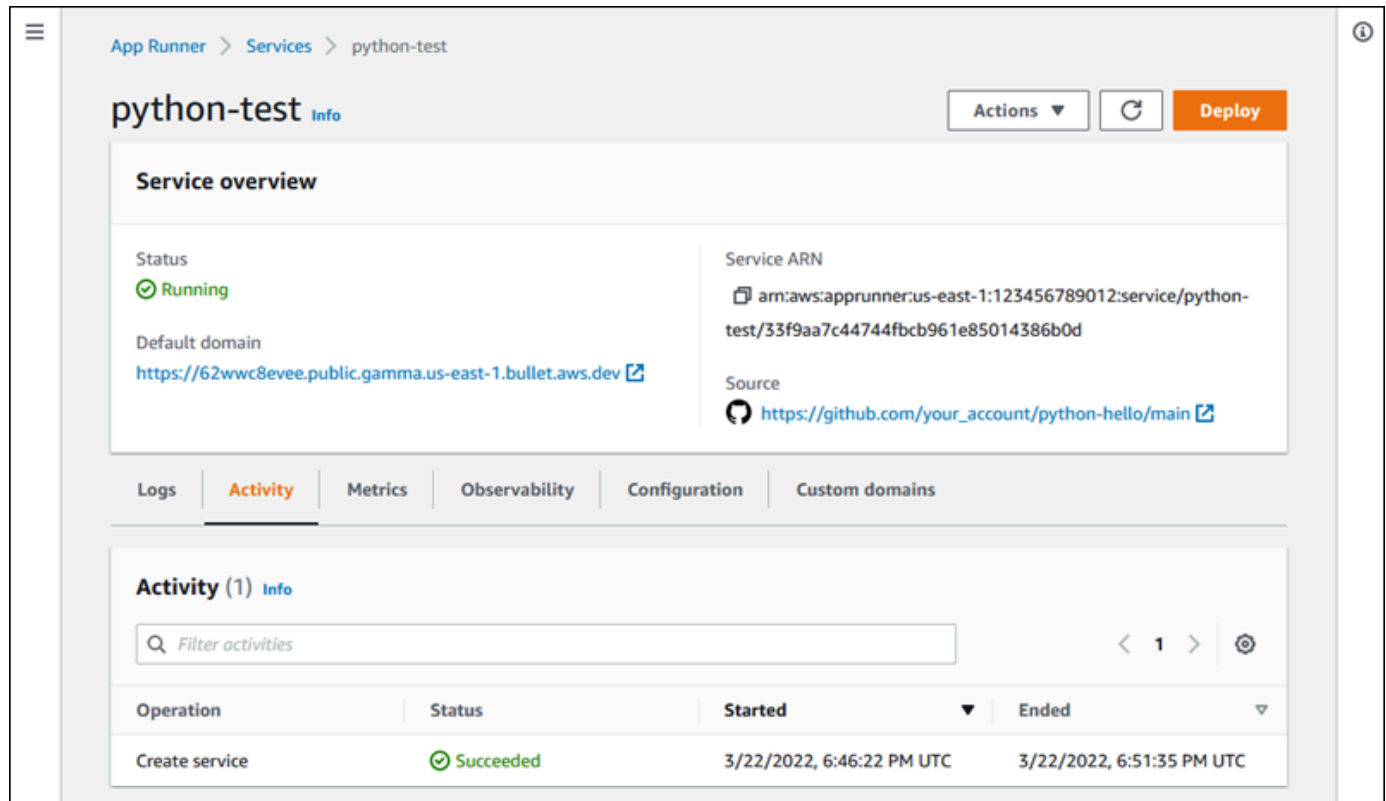
Note

No momento, o console exibe somente métricas de serviço. Para ver as métricas da instância, use o CloudWatch console.

Para ver os registros do seu serviço

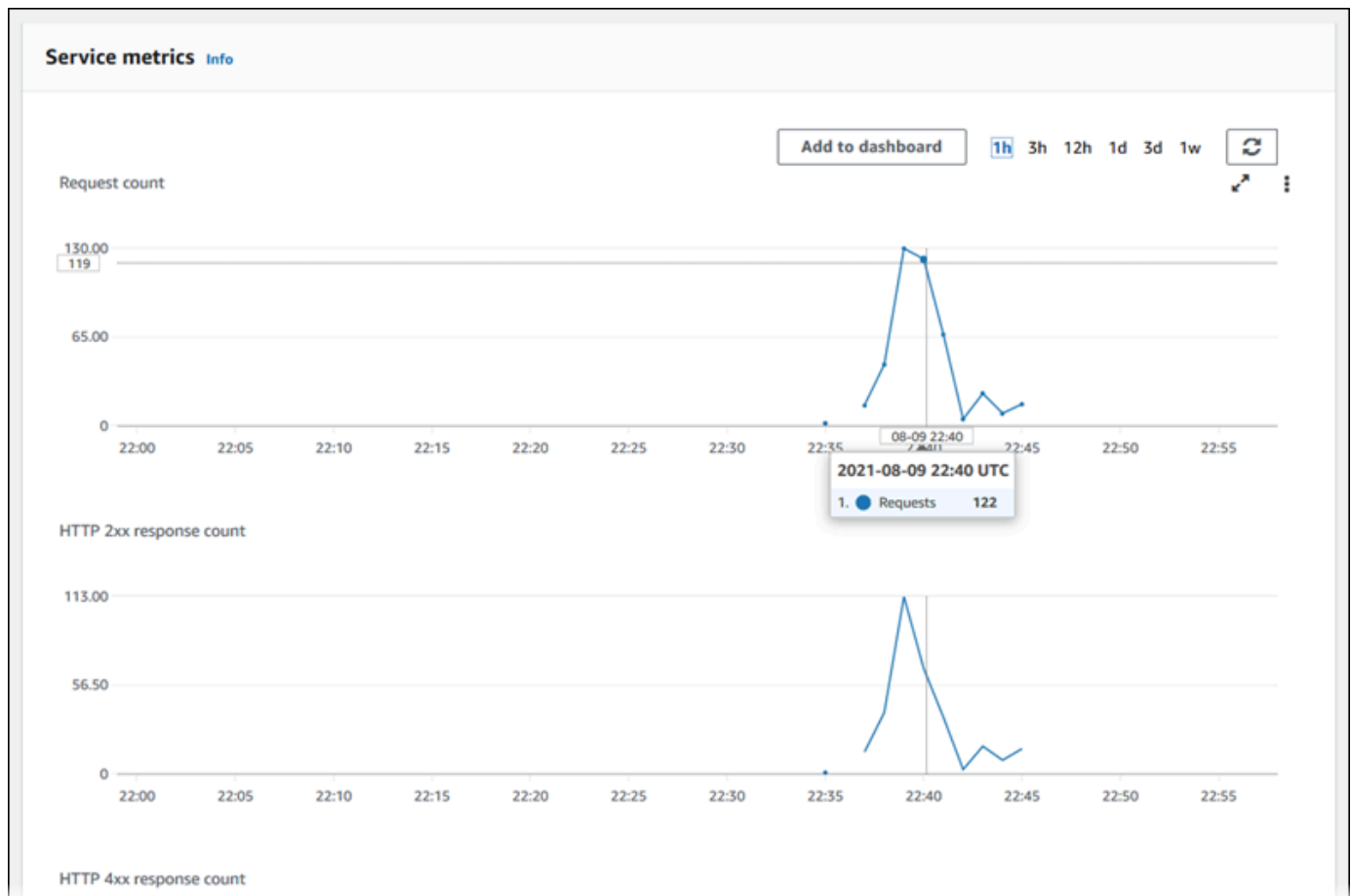
1. Abra o [console do App Runner](#) e, na lista Regiões, selecione seu Região da AWS.
2. No painel de navegação, escolha Serviços e, em seguida, escolha seu serviço App Runner.

O console exibe o painel do serviço com uma visão geral do serviço.



3. Na página do painel do serviço, escolha a guia Métricas.

O console exibe um conjunto de gráficos de métricas.



4. Escolha uma duração (por exemplo, 12h) para definir gráficos de métricas para o período recente dessa duração.
5. Escolha Adicionar ao painel na parte superior de uma das seções do gráfico ou use o menu em qualquer gráfico para adicionar as métricas relevantes a um painel no CloudWatch console para uma investigação mais aprofundada.

Gerenciando eventos do App Runner em EventBridge

Usando a Amazon EventBridge, você pode configurar regras orientadas por eventos que monitoram um fluxo de dados em tempo real do seu AWS App Runner serviço para determinados padrões. Quando um padrão para uma regra é correspondido, EventBridge inicia uma ação em um destino, como AWS Lambda, Amazon ECS e Amazon SNS. Por exemplo, você pode definir uma regra para enviar notificações por e-mail sinalizando um tópico do Amazon SNS sempre que uma implantação do seu serviço falhar. Ou você pode definir uma função Lambda para notificar um canal do Slack sempre que uma atualização de serviço falhar. Para obter mais informações sobre EventBridge, consulte o [Guia EventBridge do usuário da Amazon](#).

O App Runner envia os seguintes tipos de eventos para EventBridge

- Alteração do status do serviço — Uma alteração no status de um serviço do App Runner. Por exemplo, o status de um serviço foi alterado para `DELETE_FAILED`.
- Alteração do status da operação do serviço — Uma alteração no status de uma operação longa e assíncrona em um serviço do App Runner. Por exemplo, um serviço começou a ser criado, uma atualização de serviço foi concluída com êxito ou uma implantação de serviço foi concluída com erros.

Criação de uma EventBridge regra para agir em eventos do App Runner

Um EventBridge evento é um objeto que define alguns EventBridge campos padrão, como o AWS serviço de origem e o tipo de detalhe (evento), e um conjunto de campos específico do evento com os detalhes do evento. Para criar uma EventBridge regra, você usa o EventBridge console para definir um padrão de evento (quais eventos devem ser rastreados) e especificar uma ação alvo (o que deve ser feito em uma partida). Um padrão de evento é semelhante aos eventos aos quais ele corresponde. Você especifica um subconjunto de campos para corresponder e, para cada campo, especifica uma lista de valores possíveis. Este tópico fornece exemplos de eventos e padrões de eventos do App Runner.

Para obter mais informações sobre a criação de EventBridge regras, consulte [Criação de uma regra para um AWS serviço](#) no Guia EventBridge do usuário da Amazon.

Note

Alguns serviços oferecem suporte a padrões predefinidos em EventBridge. Isso simplifica a forma como um padrão de evento é criado. Você seleciona valores de campo em um formulário e EventBridge gera o padrão para você. No momento, o App Runner não oferece suporte a padrões predefinidos. Você precisa inserir o padrão como um objeto JSON. Você pode usar os exemplos deste tópico como ponto de partida.

Exemplos de eventos do App Runner

Estes são alguns exemplos de eventos para os quais o App Runner envia. EventBridge

- Um evento de alteração do status do serviço. Especificamente, um serviço que mudou do `OPERATION_IN_PROGRESS` para o `RUNNING` status.

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "AppRunner Service Status Change",
  "source": "aws.apprunner",
  "account": "111122223333",
  "time": "2021-04-29T11:54:23Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:apprunner:us-east-2:123456789012:service/my-app/8fe1e10304f84fd2b0df550fe98a71fa"
  ],
  "detail": {
    "previousServiceStatus": "OPERATION_IN_PROGRESS",
    "currentServiceStatus": "RUNNING",
    "serviceName": "my-app",
    "serviceId": "8fe1e10304f84fd2b0df550fe98a71fa",
    "message": "Service status is set to RUNNING.",
    "severity": "INFO"
  }
}
```

- Um evento de alteração do status da operação. Especificamente, uma UpdateService operação concluída com êxito.

```
{
  "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"
}
}
```

Exemplos de padrões de eventos do App Runner

Os exemplos a seguir demonstram padrões de eventos que você pode usar em EventBridge regras para corresponder a um ou mais eventos do App Runner. Um padrão de evento é semelhante a um evento. Inclua somente os campos que você deseja combinar e forneça uma lista em vez de um escalar para cada um.

- Combine todos os eventos de alteração de status do serviço para serviços de uma conta específica, em que o serviço não está mais no RUNNING status.

```
{
  "detail-type": [ "AppRunner Service Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "previousServiceStatus": [ "RUNNING" ]
  }
}
```

- Combine todos os eventos de alteração do status da operação para serviços de uma conta específica em que a operação falhou.

```
{
  "detail-type": [ "AppRunner Service Operation Status Change" ],
  "source": [ "aws.apprunner" ],
  "account": [ "111122223333" ],
  "detail": {
    "operationStatus": [
      "CreateServiceFailed",
      "DeleteServiceFailed",
      "UpdateServiceFailed",
      "DeploymentFailed",
      "PauseServiceFailed",
      "ResumeServiceFailed"
    ]
  }
}
```



```
}  
}
```

Referência do evento App Runner

Alteração do status do serviço

Um evento de alteração do status do serviço foi `detail-type` definido como `AppRunner Service Status Change`. Ele tem os seguintes campos e valores de detalhes:

```
"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"
```

Alteração do status da operação

Um evento de alteração do status da operação foi `detail-type` definido como `AppRunner Service Operation Status Change`. Ele tem os seguintes campos e valores de detalhes:

```
"operationStatus": "see following table",  
"serviceName": "your service name",  
"serviceId": "your service ID",  
"message": "see following table",  
"severity": "varies"
```

A tabela a seguir lista todos os códigos de status possíveis e mensagens relacionadas.

Status	Mensagem
CreateServiceStarted	A criação do serviço foi iniciada.
CreateServiceCompletedSuccessfully	A criação do serviço foi concluída com sucesso.
CreateServiceFailed	Falha na criação do serviço. Para obter detalhes, consulte os registros de serviço.

Status	Mensagem
DeleteServiceStarted	A exclusão do serviço foi iniciada.
DeleteServiceCompletedSuccessfully	A exclusão do serviço foi concluída com sucesso.
DeleteServiceFailed	Falha na exclusão do serviço.
UpdateServiceStarted	
UpdateServiceCompletedSuccessfully	A atualização do serviço foi concluída com êxito. Novo aplicativo e configuração são implantados.
	A atualização do serviço foi concluída com êxito. Uma nova configuração é implantada.
UpdateServiceFailed	Falha na atualização do serviço. Para obter detalhes, consulte os registros de serviço.
DeploymentStarted	A implantação foi iniciada.
DeploymentCompletedSuccessfully	Implantação concluída com sucesso.
DeploymentFailed	Falha na implantação. Para obter detalhes, consulte os registros de serviço.
PauseServiceStarted	A pausa no serviço foi iniciada.
PauseServiceCompletedSuccessfully	A pausa do serviço foi concluída com sucesso.
PauseServiceFailed	Falha na pausa do serviço.
ResumeServiceStarted	O currículo do serviço foi iniciado.
ResumeServiceCompletedSuccessfully	Curriculo do serviço concluído com sucesso.
ResumeServiceFailed	Falha na retomada do serviço.

Registrando chamadas da API App Runner com AWS CloudTrail

O App Runner é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no App Runner. CloudTrail captura todas as chamadas de API para o App Runner como eventos. As chamadas capturadas incluem chamadas do console do App Runner e chamadas de código para as operações da API do App Runner. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para o App Runner. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao App Runner, o endereço IP de onde a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o [Guia AWS CloudTrail do usuário](#).

Informações sobre o App Runner em CloudTrail

CloudTrail é ativado no seu Conta da AWS quando você cria a conta. Quando a atividade ocorre no App Runner, essa atividade é registrada em um CloudTrail evento junto com outros eventos de AWS serviço no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes no seu Conta da AWS. Para obter mais informações, consulte [Visualização de eventos com histórico de CloudTrail eventos](#).

Para um registro contínuo dos eventos em seu Conta da AWS, incluindo eventos do App Runner, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as Regiões da AWS. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros AWS serviços para analisar e agir com base nos dados de eventos coletados nos CloudTrail registros. Para obter mais informações, consulte:

- [Visão Geral para Criar uma Trilha](#)
- [CloudTrail Serviços e integrações compatíveis](#)
- [Configurando notificações do Amazon SNS para CloudTrail](#)
- [Recebendo arquivos de CloudTrail log de várias regiões](#) e [recebendo arquivos de CloudTrail log de várias contas](#)

Todas as ações do App Runner são registradas CloudTrail e documentadas na Referência da AWS App Runner API. Por exemplo, chamadas para as `StartDeployment` ações `CreateServiceDeleteConnection`, e geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário-raiz ou usuário do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro AWS serviço.

Para obter mais informações, consulte [Elemento `userIdentity` do CloudTrail](#).

Entendendo as entradas do arquivo de log do App Runner

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação e os parâmetros da solicitação. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a `CreateService` ação.

Note

Por motivos de segurança, alguns valores de propriedades são editados nos registros e substituídos pelo texto `HIDDEN_DUE_TO_SECURITY_REASONS`. Isso evita a exposição não intencional de informações secretas. No entanto, você ainda pode ver que essas propriedades foram passadas na solicitação ou retornadas na resposta.

Exemplo de entrada de CloudTrail registro para a ação **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,
    },
  },
}

```

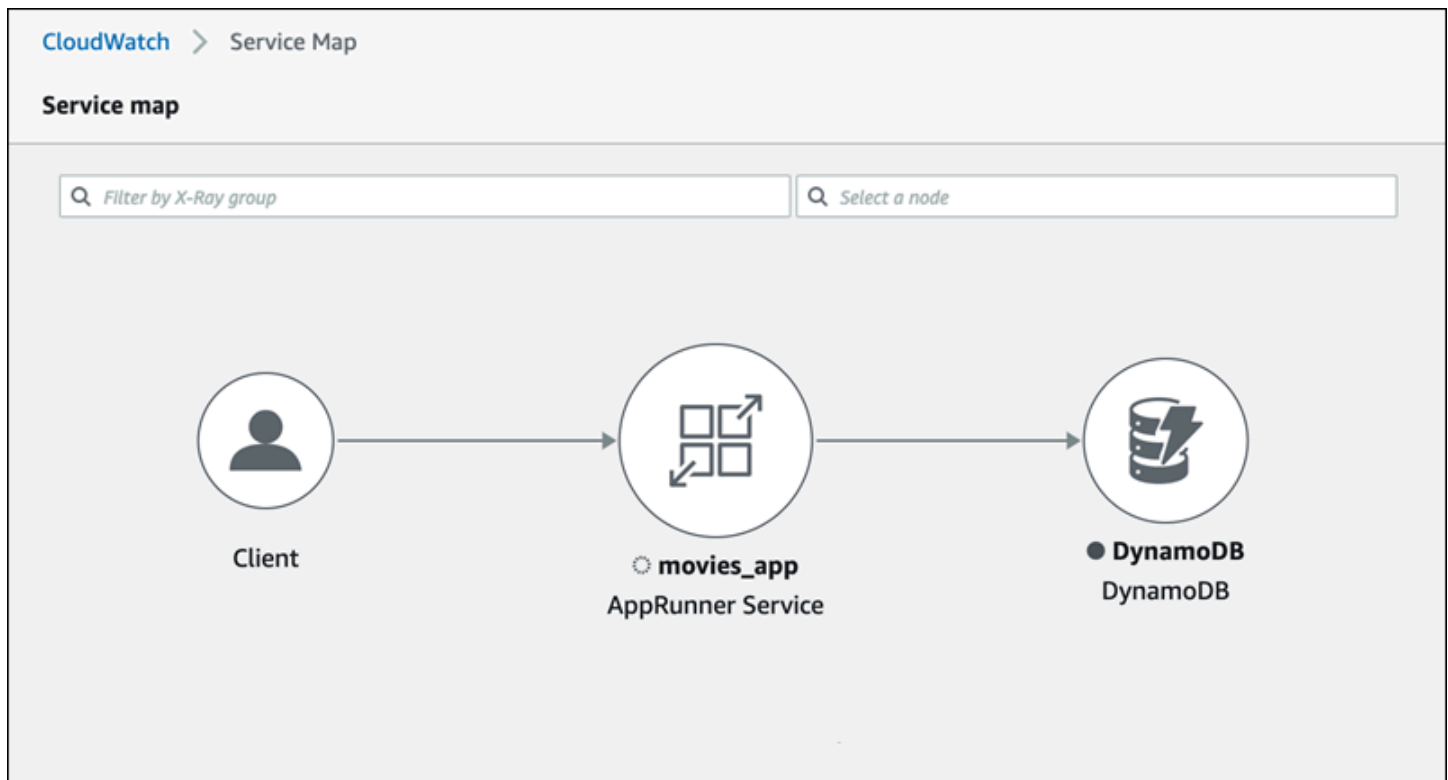
```
        "healthyThreshold": 3,
        "unhealthyThreshold": 5
    },
    "instanceConfiguration": {
        "cpu": "256",
        "memory": "1024"
    },
    "autoScalingConfigurationSummary": {
        "autoScalingConfigurationArn": "arn:aws:apprunner:us-
east-2:123456789012:autoscalingconfiguration/
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"
}
```

Rastreamento para seu aplicativo App Runner com X-Ray

AWS X-Ray é um serviço que coleta dados sobre solicitações atendidas pelo seu aplicativo e fornece ferramentas que você pode usar para visualizar, filtrar e obter informações sobre esses dados para identificar problemas e oportunidades de otimização. Para qualquer solicitação rastreada para seu aplicativo, você pode ver informações detalhadas não apenas sobre a solicitação e a resposta, mas também sobre as chamadas que seu aplicativo faz para AWS recursos downstream, microserviços, bancos de dados e web HTTP. APIs

O X-Ray usa dados de rastreamento dos AWS recursos que alimentam seus aplicativos em nuvem para gerar um gráfico de serviço detalhado. O gráfico de serviço mostra o cliente, o serviço front-end e os serviços back-end chamados pelo serviço front-end para processar solicitações e manter dados. Use o gráfico de serviço para identificar gargalos, picos de latência e outros problemas a fim de resolver o desempenho dos aplicativos.

Para obter mais informações sobre o X-Ray, consulte o [Guia do desenvolvedor do AWS X-Ray](#).



Instrumente seu aplicativo para rastreamento

Instrumente seu aplicativo de serviço App Runner para rastreamento usando [OpenTelemetry](#) uma especificação de telemetria portátil. No momento, o App Runner oferece suporte à [AWS Distro for OpenTelemetry](#) (ADOT), uma OpenTelemetry implementação que coleta e apresenta informações de telemetria usando serviços. AWS X-Ray implementa o componente de rastreamento.

Dependendo do SDK ADOT específico que você usa em seu aplicativo, o ADOT suporta até duas abordagens de instrumentação: automática e manual. Para obter mais informações sobre instrumentação com seu SDK, consulte a [documentação do ADOT](#) e escolha seu SDK no painel de navegação.

Configuração de tempo de execução

A seguir estão as instruções gerais de configuração do tempo de execução para instrumentar seu aplicativo de serviço App Runner para rastreamento.

Para configurar o rastreamento para seu tempo de execução

1. Siga as instruções fornecidas para seu tempo de execução no [AWS Distro for OpenTelemetry](#) (ADOT), para instrumentar seu aplicativo.

2. Instale as OTEL dependências necessárias na build seção do `apprunner.yaml` arquivo se você estiver usando o repositório de código-fonte ou no `Dockerfile` se estiver usando uma imagem de contêiner.
3. Configure suas variáveis de ambiente no `apprunner.yaml` arquivo se você estiver usando o repositório de código-fonte ou no `Dockerfile` se estiver usando uma imagem de contêiner.

Example Variáveis de ambiente

Note

O exemplo a seguir lista as variáveis de ambiente importantes a serem adicionadas ao `apprunner.yaml` arquivo. Adicione essas variáveis de ambiente ao seu `Dockerfile` se você estiver usando uma imagem de contêiner. No entanto, cada tempo de execução pode ter suas próprias idiossincrasias e talvez seja necessário adicionar mais variáveis de ambiente à lista a seguir. Para obter mais informações sobre instruções específicas de seu tempo de execução e exemplos sobre como configurar seu aplicativo para seu tempo de execução, consulte [AWS Distro for OpenTelemetry](#) and go your runtime, em 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` é uma variável de ambiente importante para o App Runner, pois o coletor App Runner Otel não aceita o registro de métricas. Ele só aceita rastreamento de métricas.

Exemplo de configuração de tempo de execução

[O exemplo a seguir demonstra a instrumentação automática de seu aplicativo com o SDK ADOT](#)

[Python](#). O SDK produz automaticamente extensões com dados de telemetria que descrevem os valores usados pelas estruturas Python em seu aplicativo sem adicionar uma única linha de código Python. Você precisa adicionar ou modificar apenas algumas linhas em dois arquivos de origem.

Primeiro, adicione algumas dependências, conforme mostrado no exemplo a seguir.

Example requirements.txt

```
opentelemetry-distro[otlp]>=0.24b0
opentelemetry-sdk-extension-aws~=2.0
opentelemetry-propagator-aws-xray~=1.0
```

Em seguida, instrumente seu aplicativo. A forma de fazer isso depende da fonte do serviço: imagem fonte ou código-fonte.

Source image

Quando sua fonte de serviço é uma imagem, você pode instrumentar diretamente o Dockerfile que controla a criação da imagem do contêiner e a execução do aplicativo na imagem. O exemplo a seguir mostra um Dockerfile instrumentado para um aplicativo Python. As adições de instrumentação são enfatizadas em negrito.

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

Quando sua fonte de serviço é um repositório contendo a fonte do aplicativo, você instrumenta indiretamente sua imagem usando as configurações do arquivo de configuração do App Runner. Essas configurações controlam o Dockerfile que o App Runner gera e usa para criar a imagem para seu aplicativo. O exemplo a seguir mostra um arquivo de configuração instrumentado do App Runner para um aplicativo Python. As adições de instrumentação são enfatizadas em negrito.

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'
```

Adicione permissões X-Ray à sua função de instância de serviço App Runner

Para usar o rastreamento X-Ray com seu serviço App Runner, você precisa fornecer às instâncias do serviço permissões para interagir com o serviço X-Ray. Você faz isso associando uma função de instância ao seu serviço e adicionando uma política gerenciada com permissões X-Ray. Para

obter mais informações sobre uma função de instância do App Runner, consulte [the section called “Função da instância”](#). Adicione a política `AWSXRayDaemonWriteAccess` gerenciada à sua função de instância e atribua-a ao seu serviço durante a criação.

Ative o rastreamento X-Ray para seu serviço App Runner

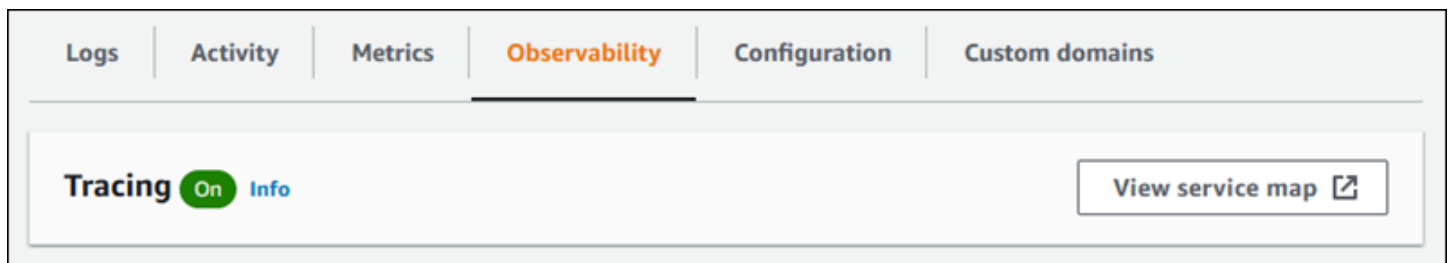
Quando você [cria um serviço](#), o App Runner desativa o rastreamento por padrão. Você pode ativar o rastreamento X-Ray para seu serviço como parte da configuração da observabilidade. Para obter mais informações, consulte [the section called “Gerencie a observabilidade”](#).

Se você usa a API App Runner ou a AWS CLI, o [TraceConfiguration](#) objeto dentro do objeto de [ObservabilityConfiguration](#) recurso contém configurações de rastreamento. Para manter o rastreamento desativado, não especifique um `TraceConfiguration` objeto.

Nos casos do console e da API, não se esqueça de associar sua função de instância discutida na seção anterior ao seu serviço App Runner.

Visualize os dados de rastreamento do X-Ray para seu serviço App Runner

Na guia Observabilidade da [página do painel de serviços](#) no console do App Runner, escolha Exibir mapa do serviço para navegar até o console da Amazon CloudWatch .



Use o CloudWatch console da Amazon para visualizar mapas e rastreamentos de serviços para solicitações atendidas pelo seu aplicativo. Os mapas de serviços mostram informações como latência da solicitação e interações com outros aplicativos e AWS serviços. As anotações personalizadas que você adiciona ao seu código permitem que você pesquise facilmente por rastreamentos. Para obter mais informações, consulte [Usando ServiceLens para monitorar a integridade de seus aplicativos](#) no Guia do CloudWatch usuário da Amazon.

Associando uma ACL AWS WAF da web ao seu serviço

AWS WAF é um firewall de aplicativos da web que você pode usar para proteger seu serviço App Runner. Com as listas de controle de acesso à AWS WAF web (web ACLs), você pode proteger seus endpoints do serviço App Runner contra explorações comuns da web e bots indesejados.

Uma ACL da web fornece um controle refinado sobre todas as solicitações da web recebidas para seu serviço App Runner. Você pode definir regras em uma ACL da web para permitir, bloquear ou monitorar o tráfego da web, para garantir que somente solicitações autorizadas e legítimas cheguem aos seus aplicativos da web e APIs. Você pode personalizar as regras de ACL da web com base em suas necessidades comerciais e de segurança específicas. Para saber mais sobre a segurança da infraestrutura e as melhores práticas para aplicar a rede ACLs, consulte [Controle o tráfego de rede](#) no Guia do usuário da Amazon VPC.

Important

As regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP. Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de [grupo de segurança para endpoints privados](#) em vez de WAF web ACLs.

Fluxo de entrada de solicitações da web

Quando uma ACL AWS WAF da web é associada a um serviço App Runner, as solicitações da web recebidas passam pelo seguinte processo:

1. O App Runner encaminha o conteúdo da solicitação de origem para o AWS WAF
2. AWS WAF inspeciona a solicitação e compara seu conteúdo com as regras que você especificou na sua Web ACL.
3. Com base em sua inspeção, AWS WAF retorna uma block resposta allow ou ao App Runner.
 - Se uma allow resposta for retornada, o App Runner encaminha a solicitação para seu aplicativo.

- Se uma block resposta for retornada, o App Runner impede que a solicitação chegue ao seu aplicativo web. Ele encaminha a block resposta AWS WAF para sua inscrição.

 Note

Por padrão, o App Runner bloqueia a solicitação se nenhuma resposta for retornada AWS WAF.

Para obter mais informações sobre a AWS WAF web ACLs, consulte [Listas de controle de acesso à Web \(web ACLs\)](#) no Guia do AWS WAF desenvolvedor.

 Note

Você paga o AWS WAF preço padrão. Você não incorre em nenhum custo adicional pelo uso AWS WAF da web ACLs para seus serviços do App Runner. Para obter mais informações sobre preços, consulte [AWS WAF Preços](#).

Associando o WAF web ACLs ao seu serviço App Runner

A seguir está o processo de alto nível para associar uma ACL AWS WAF da web ao seu serviço App Runner:

1. Crie uma ACL da web no AWS WAF console. Para obter mais informações, consulte [Criação de uma ACL da web](#) no Guia do AWS WAF desenvolvedor.
2. Atualize suas permissões AWS Identity and Access Management (IAM) para AWS WAF. Para obter mais informações, consulte [Permissões do](#).
3. Associe a Web ACL ao serviço App Runner usando um dos seguintes métodos:
 - Console do App Runner: associe uma ACL da web existente usando o console do App Runner ao [criar](#) ou [atualizar](#) um serviço do App Runner. Para obter instruções, consulte [Gerenciando a AWS WAF web ACLs](#).
 - AWS WAF console: associe a ACL da web usando o AWS WAF console para um serviço existente do App Runner. Para obter mais informações, consulte [Associar ou desassociar uma ACL da web a um recurso da AWS](#) no Guia do desenvolvedor.AWS WAF

- AWS CLI: associe a ACL da web usando o AWS WAF público APIs. Para obter mais informações sobre o AWS WAF público APIs, consulte [AssociateWebACL](#) no Guia de referência da AWS WAF API.

Considerações

- As regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP. Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de [grupo de segurança para endpoints privados](#) em vez de WAF web. ACLs
- Um serviço App Runner só pode ser associado a uma ACL da web. No entanto, você pode associar uma ACL da web a vários serviços do App Runner e a vários AWS recursos. Os exemplos incluem grupos de usuários do Amazon Cognito e recursos do Application Load Balancer.
- Quando você cria uma ACL da web, passa um pequeno tempo até que a ACL da web se propague totalmente e fique disponível para o App Runner. O tempo de propagação pode ser de alguns segundos a alguns minutos. AWS WAF retorna a `WAFUnavailableEntityException` quando você tenta associar uma ACL da web antes que ela seja totalmente propagada.

Se você atualizar o navegador ou sair do console do App Runner antes que a Web ACL seja totalmente propagada, a associação não ocorrerá. No entanto, você pode navegar no console do App Runner.

- AWS WAF retorna um `WAFNonexistantItemException` erro quando você chama um dos seguintes AWS WAF APIs para um serviço App Runner que está em um estado inválido:
 - `AssociateWebACL`
 - `DisassociateWebACL`
 - `GetWebACLForResource`

Os estados inválidos do seu serviço App Runner incluem:

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`

- OPERATION_IN_PROGRESS

 Note

OPERATION_IN_PROGRESS estado é inválido somente se o serviço App Runner estiver sendo excluído.

- Sua solicitação pode resultar em uma carga útil maior do que os limites do que AWS WAF pode ser inspecionado. Para obter mais informações sobre como AWS WAF lidar com solicitações de tamanho grande do App Runner, consulte Tratamento de [componentes de solicitações de tamanho grande no Guia do AWS WAF desenvolvedor para saber como lidar](#) AWS WAF com solicitações de tamanho grande do App Runner.
- Se você não definir regras apropriadas ou se seus padrões de tráfego mudarem, uma Web ACL pode não ser tão eficaz para proteger seu aplicativo.

Permissões

Para trabalhar com uma Web ACL em AWS App Runner, adicione as seguintes permissões do IAM para AWS WAF:

- `apprunner:ListAssociatedServicesForWebAcl`
- `apprunner:DescribeWebAclForService`
- `apprunner:AssociateWebAcl`
- `apprunner:DisassociateWebAcl`

Para obter mais informações sobre as permissões do IAM, consulte [Políticas e permissões no IAM](#) no Guia do usuário do IAM.

Veja a seguir um exemplo da política atualizada do IAM para AWS WAF. Essa política do IAM inclui as permissões necessárias para trabalhar com um serviço App Runner.

Example

```
{
  {
    "Version": "2012-10-17",
    "Statement": [
```



```
{
  "Effect": "Allow",
  "Action": [
    "wafv2:ListResourcesForWebACL",
    "wafv2:GetWebACLForResource",
    "wafv2:AssociateWebACL",
    "wafv2:DisassociateWebACL",
    "apprunner:ListAssociatedServicesForWebAcl",
    "apprunner:DescribeWebAclForService",
    "apprunner:AssociateWebAcl",
    "apprunner:DisassociateWebAcl"
  ],
  "Resource": "*"
}
```

Note

Embora você deva conceder permissões do IAM, as ações listadas são somente com permissão e não correspondem a uma operação de API.

Gerenciando a AWS WAF web ACLs

Gerencie a AWS WAF web ACLs para seu serviço App Runner usando um dos seguintes métodos:

- [the section called “Console do App Runner”](#)
- [the section called “AWS CLI”](#)

Console do App Runner

Ao [criar um serviço](#) ou [atualizar um existente](#) no console do App Runner, você pode associar ou desassociar uma ACL AWS WAF da web.

 Note

- Um serviço App Runner só pode ser associado a uma ACL da web. No entanto, você pode associar uma ACL da web a mais de um serviço do App Runner, além de outros AWS recursos.
- Antes de associar uma Web ACL, certifique-se de atualizar suas permissões do IAM para AWS WAF. Para obter mais informações, consulte [Permissões do](#).

Associando a ACL AWS WAF da web

 Important

As regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP. Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de [grupo de segurança para endpoints privados](#) em vez de WAF web. ACLs

Para associar uma AWS WAF ACL da web

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Com base no fato de você estar criando ou atualizando um serviço, execute uma das seguintes etapas:
 - Se você estiver criando um novo serviço, escolha Criar um serviço App Runner e acesse Configurar serviço.
 - Se você estiver atualizando um serviço existente, escolha a guia Configuração e escolha Editar em Configurar serviço.
3. Vá para Firewall de aplicativos da Web em Segurança.
4. Escolha o botão de alternância Ativar para ver as opções.

Security

Info

Specify an Instance role and an AWS KMS encryption key

Permissions

Select an IAM role with permissions to AWS actions that your service code calls. To create a custom role, use the [IAM console](#)

Instance role

An Instance role is auto-generated for every IAM role that is created for Amazon EC2 using the AWS Management Console. Choose an Instance role to apply the required IAM role to your application code. This grants access permissions to call AWS services.

Choose an Instance role

AWS KMS key

This key is used to encrypt the stored copies of your data.

☒ Use an AWS-owned key

A key that AWS owns and manages for you.

☐ Choose a different AWS KMS key

A key that you own or have permission to use.

Web Application Firewall

Info

Activate WAF to define Web access control list (ACL) to protect against web exploits and bots. Learn more about [WAF and pricing](#).

☒ Activate

Choose a web ACL (0)

Create a web ACL

Choose an existing web ACL or create a new one in AWS WAF console. If you create a new web ACL, click the refresh button to view it in the table below.

Find resource

< 1 >

Name	Description	ID
No web ACL		
No resources to display		
Create a web ACL		

5. Execute uma das seguintes etapas:

- Para associar uma ACL da web existente: escolha a ACL da web necessária na tabela Escolha uma ACL da web para associar ao seu serviço App Runner.

- Para criar uma nova ACL da web: escolha Criar ACL da web para criar uma nova ACL da web usando o console. AWS WAF Para obter mais informações, consulte [Criação de uma ACL da web](#) no Guia do AWS WAF desenvolvedor.
 1. Escolha o botão Atualizar para visualizar a ACL da Web recém-criada na tabela Escolher uma ACL da Web.
 2. Selecione a ACL da web necessária.
- 6. Escolha Avançar se estiver criando um novo serviço ou Salvar alterações se estiver atualizando um serviço existente. A ACL da web selecionada está associada ao seu serviço App Runner.
- 7. Para verificar a associação da Web ACL, escolha a guia Configuração do seu serviço e vá para Configurar serviço. Role até Firewall do aplicativo Web em Segurança para ver os detalhes da ACL da web associada ao seu serviço.

 Note

Quando você cria uma ACL da web, passa um pequeno tempo até que a ACL da web se propague totalmente e fique disponível para o App Runner. O tempo de propagação pode ser de alguns segundos a alguns minutos. AWS WAF retorna a `WAFUnavailableEntityException` quando você tenta associar uma ACL da web antes que ela seja totalmente propagada.

Se você atualizar o navegador ou sair do console do App Runner antes que a Web ACL seja totalmente propagada, a associação não ocorrerá. No entanto, você pode navegar no console do App Runner.

Desassociando uma AWS WAF ACL da web

Você pode desassociar a AWS WAF web ACL que não precisa mais [atualizando seu serviço](#) App Runner.

Para desassociar uma web AWS WAF ACL

1. Abra o [console do App Runner](#) e, na lista Regiões, selecione sua Região da AWS.
2. Vá para a guia Configuração do serviço que você deseja atualizar e escolha Editar em Configurar serviço.
3. Vá para Firewall de aplicativos da Web em Segurança.
4. Desative o botão de alternância Ativar. Você recebe uma mensagem para confirmar a exclusão.

5. Escolha Confirmar. A Web ACL está desassociada do seu serviço App Runner.

Note

- Se você quiser associar seu serviço a outra ACL da Web, selecione uma ACL da Web na tabela Escolher uma ACL da Web. O App Runner desassocia a ACL da web atual e inicia o processo de associação à ACL da web selecionada.
- Se nenhum outro serviço ou recurso do App Runner usar uma ACL da web não associada, considere excluir a ACL da web. Caso contrário, você continuará incorrendo em custos. Para obter mais informações sobre precificação, consulte [Precificação do AWS WAF](#). Para obter instruções sobre como excluir uma ACL da web, consulte [DeleteWebACL na Referência](#) da AWS WAF API.
- Você não pode excluir uma Web ACL associada a outros serviços ativos do App Runner ou a outros recursos.

AWS CLI

Você pode associar ou desassociar uma ACL AWS WAF da web usando o AWS WAF público. APIs O serviço App Runner, ao qual você deseja associar ou desassociar uma ACL da web, deve estar em um estado válido.

AWS WAF retorna um `WAFNonexistentItemException` erro quando você chama um dos seguintes AWS WAF APIs para um serviço App Runner que está em um estado inválido:

- `AssociateWebACL`
- `DisassociateWebACL`
- `GetWebACLForResource`

Os estados inválidos do seu serviço App Runner incluem:

- `CREATE_FAILED`
- `DELETE_FAILED`
- `DELETED`
- `OPERATION_IN_PROGRESS`

Note

OPERATION_IN_PROGRESS estado é inválido somente se o serviço App Runner estiver sendo excluído.

Para obter mais informações sobre o AWS WAF público APIs, consulte o [Guia de referência AWS WAF da API](#).

Note

Atualize suas permissões do IAM para AWS WAF. Para obter mais informações, consulte [Permissões do](#) .

Associando a AWS WAF Web ACL usando AWS CLI

Important

As regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP. Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de [grupo de segurança para endpoints privados](#) em vez de WAF web. ACLs

Para associar uma AWS WAF ACL da web

1. Crie uma ACL AWS WAF da web para seu serviço com seu conjunto preferido de ações de regras Allow ou solicitações Block da web para seu serviço. Para obter mais informações sobre AWS WAF APIs, consulte [CreateWebACL](#) no Guia de referência da AWS WAF API.

Example Criar uma ACL da web - Solicitação

```
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. Associe a ACL da web que você criou ao serviço App Runner usando a API `associate-web-acl` AWS WAF pública. Para obter mais informações sobre AWS WAF APIs, consulte [AssociateWebACL](#) no Guia de referência da AWS WAF API.

Note

Quando você cria uma ACL da web, passa um pequeno tempo até que a ACL da web se propague totalmente e fique disponível para o App Runner. O tempo de propagação pode ser de alguns segundos a alguns minutos. AWS WAF retorna a `WAFUnavailableEntityException` quando você tenta associar uma ACL da web antes que ela seja totalmente propagada.

Se você atualizar o navegador ou sair do console do App Runner antes que a Web ACL seja totalmente propagada, a associação não ocorrerá. No entanto, você pode navegar no console do App Runner.

Example Associando uma ACL da web - Solicitação

```
aws wafv2 associate-web-acl
--resource-arn <apprunner_service_arn>
--web-acl-arn <web_acl_arn>
--region <region>
```

3. Verifique se a ACL da web está associada ao seu serviço App Runner usando a API `get-web-acl-for-resource` AWS WAF pública. Para obter mais informações sobre AWS WAF APIs, consulte [GetWebACLForResource](#) no Guia de referência AWS WAF da API.

Example Verificar a ACL da web para o recurso - Solicitação

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

Se não houver nenhuma web ACLs associada ao seu serviço, você receberá uma resposta em branco.

Excluindo uma ACL AWS WAF da web usando AWS CLI

Você não pode excluir uma ACL AWS WAF da web se ela estiver associada a um serviço do App Runner.

Para excluir uma AWS WAF ACL da web

1. Desassocie a ACL da web do seu serviço App Runner usando a `disassociate-web-acl` AWS WAF API pública. Para obter mais informações sobre AWS WAF APIs, consulte [DisassociateWebACL](#) no Guia de referência da AWS WAF API.

Example Desassociando uma ACL da web - Solicitação

```
aws wafv2 disassociate-web-acl
--resource-arn <apprunner_service_arn>
--region <region>
```

2. Verifique se a ACL da web está desassociada do seu serviço App Runner usando a `get-web-acl-for-resource` AWS WAF API pública.

Example Verifique se a ACL da web está desassociada - Solicitação

```
aws wafv2 get-web-acl-for-resource
--resource-arn <apprunner_service_arn>
--region <region>
```

A ACL da web desassociada não está listada para seu serviço App Runner. Se não houver nenhuma web ACLs associada ao seu serviço, você receberá uma resposta em branco.

3. Exclua a ACL da web desassociada usando a API `delete-web-acl` AWS WAF pública. Para obter mais informações sobre AWS WAF APIs, consulte [DeleteWebACL](#) no Guia de referência da AWS WAF API.

Example Excluir uma ACL da web - Solicitação

```
aws wafv2 delete-web-acl
--name <web_acl_name>
```



```
--scope REGIONAL
--id <web_acl_id>
--lock-token <web_acl_lock_token>
--region <region>
```

4. Verifique se a ACL da web foi excluída usando a API `list-web-acl` AWS WAF pública. Para obter mais informações sobre AWS WAF APIs, consulte [ListWebACLs](#) no Guia de referência AWS WAF da API.

Example Verifique se a ACL da web foi excluída - Solicitação

```
aws wafv2 list-web-acls
--scope REGIONAL
--region <region>
```

A ACL da web excluída não está mais listada.

 Note

Se uma ACL da web estiver associada a outros serviços ativos do App Runner ou a outros recursos, como grupos de usuários do Amazon Cognito, a ACL da web não poderá ser excluída.

Listando os serviços do App Runner associados a uma ACL da web

Uma ACL da web pode ser associada a vários serviços do App Runner e outros recursos. Liste os serviços do App Runner associados a uma ACL da web usando a API `list-resources-for-web-acl` AWS WAF pública. Para obter mais informações sobre AWS WAF APIs, consulte [ListResourcesForWebACL](#) no Guia de referência da AWS WAF API.

Example Listar serviços do App Runner associados a uma ACL da web - Solicitação

```
aws wafv2 list-resources-for-web-acl
--web-acl-arn <WEB_ACL_ARN>
--resource-type APP_RUNNER_SERVICE
--region <REGION>
```

Example Listar serviços do App Runner associados a uma ACL da web - Resposta

O exemplo a seguir ilustra a resposta quando não há serviços do App Runner associados a uma ACL da web.

```
{
  "ResourceArns": []
}
```

Example Listar serviços do App Runner associados a uma ACL da web - Resposta

O exemplo a seguir ilustra a resposta quando há serviços do App Runner associados a uma ACL da web.

```
{
  "ResourceArns": [
    "arn:aws:apprunner:<region>:<aws_account_id>:service/<service_name>/<service_id>"
  ]
}
```

Testando e registrando AWS WAF na web ACLs

Quando você define uma ação de regra como Count em sua ACL da web, AWS WAF adiciona a solicitação a uma contagem de solicitações que correspondem à regra. Para testar uma ACL da web com seu serviço App Runner, defina ações de regra como Count e considere o volume de solicitações que correspondem a cada regra. Por exemplo, você define uma regra para a Block ação que corresponde a um grande número de solicitações que você determina como tráfego normal de usuários. Nesse caso, talvez seja necessário reconfigurar sua regra. Para obter mais informações, consulte [Teste e ajuste de suas AWS WAF proteções](#) no Guia do AWS WAF desenvolvedor.

Você também pode configurar AWS WAF para registrar cabeçalhos de solicitação em um grupo de CloudWatch logs do Amazon Logs, em um bucket do Amazon Simple Storage Service (Amazon S3) ou em um Amazon Data Firehose. Para obter mais informações, consulte [Logging web ACL traffic](#) (Registrar em log o tráfego da ACL da web) no Guia do desenvolvedor do AWS WAF .

Para acessar registros relacionados à ACL da web associada ao seu serviço App Runner, consulte os seguintes campos de registro:

- `httpSourceName`: Contém APPRUNNER

- `httpSourceId`: Contém `customeraccountid-apprunnerserviceid`

Para obter mais informações, consulte [Exemplos de registros](#) no Guia do AWS WAF desenvolvedor.

 Important

As regras de IP de origem para serviços privados do App Runner associados à Web ACLs do WAF não aderem às regras baseadas em IP. Isso ocorre porque atualmente não oferecemos suporte ao encaminhamento de dados IP de origem da solicitação para os serviços privados do App Runner associados ao WAF. Se seu aplicativo App Runner exigir regras de controle de tráfego de entrada IP/CIDR de origem, você deverá usar regras de [grupo de segurança para endpoints privados](#) em vez de WAF web. ACLs

Definindo as opções do serviço App Runner usando um arquivo de configuração

Note

Os arquivos de configuração são aplicáveis somente aos [serviços baseados no código-fonte](#). Você não pode usar arquivos de configuração com serviços [baseados em imagem](#).

Quando você cria um AWS App Runner serviço usando um repositório de código-fonte, AWS App Runner requer informações sobre como criar e iniciar seu serviço. Você pode fornecer essas informações sempre que criar um serviço usando o console ou a API do App Runner. Como alternativa, você pode definir opções de serviço usando um arquivo de configuração. As opções que você especifica em um arquivo se tornam parte do seu repositório de origem, e todas as alterações nessas opções são rastreadas de forma semelhante à forma como as alterações no código-fonte são rastreadas. Você pode usar o arquivo de configuração do App Runner para especificar mais opções do que as suportadas pela API. Você não precisa fornecer um arquivo de configuração se precisar apenas das opções básicas suportadas pela API.

O arquivo de configuração do App Runner é um arquivo YAML nomeado `apprunner.yaml` no [diretório de origem do repositório](#) do seu aplicativo. Ele fornece opções de criação e tempo de execução para seu serviço. Os valores nesse arquivo instruem o App Runner a criar e iniciar seu serviço e fornecem contexto de tempo de execução, como configurações de rede e variáveis de ambiente.

O arquivo de configuração do App Runner não inclui configurações operacionais, como CPU e memória.

Para obter exemplos de arquivos de configuração do App Runner, consulte [the section called “Exemplos”](#). Para obter um guia de referência completo, consulte [the section called “Referência”](#).

Tópicos

- [Exemplos de arquivos de configuração do App Runner](#)
- [Referência do arquivo de configuração do App Runner](#)

Exemplos de arquivos de configuração do App Runner

Note

Os arquivos de configuração são aplicáveis somente aos [serviços baseados no código-fonte](#). Você não pode usar arquivos de configuração com serviços [baseados em imagem](#).

Os exemplos a seguir demonstram os arquivos AWS App Runner de configuração. Alguns são mínimos e contêm apenas as configurações necessárias. Outros estão completos, incluindo todas as seções do arquivo de configuração. Para obter uma visão geral dos arquivos de configuração do App Runner, consulte [Arquivo de configuração do App Runner](#).

Exemplos de arquivos de configuração

Arquivo de configuração mínimo

Com um arquivo de configuração mínimo, o App Runner faz as seguintes suposições:

- Nenhuma variável de ambiente personalizada é necessária durante a construção ou execução.
- A versão de tempo de execução mais recente é usada.
- O número da porta padrão e a variável de ambiente da porta são usados.

Example apprunner.yaml

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

Arquivo de configuração completo

Este exemplo mostra o uso de todas as chaves de configuração no formato `apprunner.yaml` original com um tempo de execução gerenciado.

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"

```

Arquivo de configuração completo — (usa compilação revisada)

Este exemplo mostra o uso de todas as chaves de configuração no `apprunner.yaml` com um tempo de execução gerenciado.

O `pre-run` parâmetro só é compatível com a versão revisada do App Runner. Não insira esse parâmetro no arquivo de configuração se o aplicativo usar versões de tempo de execução compatíveis com a compilação original do App Runner. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Note

Como esse exemplo é para o Python 3.11, usamos os comandos `pip3` `python3`. Para obter mais informações, consulte [Explicações para versões específicas de tempo de execução](#) o tópico da plataforma 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"
```

Para exemplos de arquivos específicos de configuração de tempo de execução gerenciado, consulte o subtópico de tempo de execução específico abaixo [Serviço baseado em código](#).

Referência do arquivo de configuração do App Runner

Note

Os arquivos de configuração são aplicáveis somente aos [serviços baseados no código-fonte](#). Você não pode usar arquivos de configuração com serviços [baseados em imagem](#).

Este tópico é um guia de referência abrangente sobre a sintaxe e a semântica de um arquivo de AWS App Runner configuração. Para obter uma visão geral dos arquivos de configuração do App Runner, consulte [Arquivo de configuração do App Runner](#).

O arquivo de configuração do App Runner é um arquivo YAML. Dê um `apprunner.yaml` nome e coloque-o no [diretório de origem](#) do repositório do seu aplicativo.

Visão geral da estrutura

O arquivo de configuração do App Runner é um arquivo YAML. Dê um `apprunner.yaml` nome e coloque-o no [diretório de origem](#) do repositório do seu aplicativo.

O arquivo de configuração do App Runner contém as seguintes partes principais:

- Seção superior — Contém teclas de nível superior
- Seção de construção — configura o estágio de construção
- Seção Executar — Configura o estágio de execução

Seção superior

As chaves na parte superior do arquivo fornecem informações gerais sobre o arquivo e o tempo de execução do serviço. As seguintes chaves estão disponíveis:

- **version**— Obrigatório. A versão do arquivo de configuração do App Runner. O ideal é usar a versão mais recente.

Sintaxe

```
version: version
```

Example

```
version: 1.0
```

- **runtime**— Obrigatório. O nome do tempo de execução que seu aplicativo usa. Para saber mais sobre os tempos de execução disponíveis para as diferentes plataformas de programação que o App Runner oferece, consulte. [Serviço baseado em código](#)

Note

A convenção de nomenclatura de um tempo de execução gerenciado é *<language-name><major-version>*.

Sintaxe

```
runtime: runtime-name
```

Example

```
runtime: python3
```

Seção de construção

A seção de compilação configura o estágio de construção da implantação do serviço App Runner. Você pode especificar comandos de construção e variáveis de ambiente. Os comandos de compilação são obrigatórios.

A seção começa com a `build:` chave e tem as seguintes subchaves:

- `commands`— Obrigatório. Especifica os comandos que o App Runner executa durante várias fases de compilação. Inclui as seguintes subchaves:
 - `pre-build`— Opcional. Os comandos que o App Runner executa antes da compilação. Por exemplo, instale npm dependências ou teste bibliotecas.
 - `build`— Obrigatório. Os comandos que o App Runner executa para criar seu aplicativo. Por exemplo, use `pipenv`.
 - `post-build`— Opcional. Os comandos que o App Runner executa após a compilação. Por exemplo, use o Maven para empacotar artefatos de construção em um arquivo JAR ou WAR ou executar um teste.

Sintaxe

```
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**— Opcional. Especifica variáveis de ambiente personalizadas para o estágio de construção. Definido como mapeamentos escalares de nome-valor. Você pode se referir a essas variáveis pelo nome nos seus comandos de compilação.

 Note

Há duas env entradas distintas em dois locais diferentes nesse arquivo de configuração. Um conjunto está na seção Construir e o outro na seção Executar.

- O env conjunto na seção Build pode ser referenciado pelos `pre-run` comandos `pre-build`, `buildpost-build`, e durante o processo de compilação.

Importante: observe que os `pre-run` comandos estão localizados na seção Executar desse arquivo, embora possam acessar somente as variáveis de ambiente definidas na seção Construir.

- O env conjunto na seção Executar pode ser referenciado pelo `run` comando no ambiente de execução.

Sintaxe

```
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"
```

Seção de execução

A seção de execução configura o estágio de execução do contêiner da implantação do aplicativo App Runner. Você pode especificar a versão do tempo de execução, os comandos de pré-execução (somente no formato revisado), o comando de início, a porta de rede e as variáveis de ambiente.

A seção começa com a `run:` chave e tem as seguintes subchaves:

- `runtime-version`— Opcional. Especifica uma versão de tempo de execução que você deseja bloquear para o serviço App Runner.

Por padrão, somente a versão principal está bloqueada. O App Runner usa as versões secundárias e de patch mais recentes que estão disponíveis para o tempo de execução em cada implantação ou atualização de serviço. Se você especificar versões principais e secundárias, ambas ficarão bloqueadas e o App Runner atualizará somente as versões de patch. Se você especificar versões principais, secundárias e de patch, seu serviço será bloqueado em uma versão de tempo de execução específica e o App Runner nunca a atualizará.

Sintaxe

```
run:  
  runtime-version: major[.minor[.patch]]
```

Note

Os tempos de execução de algumas plataformas têm componentes de versão diferentes. Consulte os tópicos específicos da plataforma para obter detalhes.

Example

```
runtime: python3  
run:  
  runtime-version: 3.7
```

- `pre-run`— Opcional. Somente para uso [revisado da compilação](#). Especifica os comandos que o App Runner executa após copiar seu aplicativo da imagem de compilação para a imagem de execução. Você pode inserir comandos aqui para modificar a imagem de execução fora do `/app` diretório. Por exemplo, se você precisar instalar dependências globais adicionais que residem fora

do /app diretório, insira os comandos necessários nesta subseção para fazer isso. Para obter mais informações sobre o processo de criação do App Runner, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Note

- Importante — Embora os pre-run comandos estejam listados na seção Executar, eles só podem referenciar as variáveis de ambiente definidas na seção Construir desse arquivo de configuração. Eles não podem referenciar as variáveis de ambiente definidas nesta seção Executar.
- O pre-run parâmetro só é compatível com a versão revisada do App Runner. Não insira esse parâmetro no arquivo de configuração se o aplicativo usar versões de tempo de execução compatíveis com a compilação original do App Runner. Para obter mais informações, consulte [Versões de tempo de execução gerenciadas e a compilação do App Runner](#).

Sintaxe

```
run:
  pre-run:
    - command
    - ...
```

- **command**— Obrigatório. O comando que o App Runner usa para executar seu aplicativo depois de concluir a criação do aplicativo.

Sintaxe

```
run:
  command: command
```

- **network**— Opcional. Especifica a porta que seu aplicativo escuta. A tabela inclui os seguintes campos:
 - **port**— Opcional. Se especificado, esse é o número da porta que seu aplicativo escuta. O padrão é 8080.
 - **env**— Opcional. Se especificado, o App Runner passa o número da porta para o contêiner nessa variável de ambiente, além de (não em vez de) passar o mesmo número de porta na

variável de ambiente padrão, `PORT`. Em outras palavras, se você especificar `env`, o App Runner passará o número da porta em duas variáveis de ambiente.

Sintaxe

```
run:
  network:
    port: port-number
    env: env-variable-name
```

Example

```
run:
  network:
    port: 8000
    env: MY_APP_PORT
```

- `env`— Opcional. Definição de variáveis de ambiente personalizadas para o estágio de execução. Definido como mapeamentos escalares de nome-valor. Você pode se referir a essas variáveis pelo nome em seu ambiente de tempo de execução.

Note

Há duas `env` entradas distintas em dois locais diferentes nesse arquivo de configuração. Um conjunto está na seção `Construir` e o outro na seção `Executar`.

- O `env` conjunto na seção `Build` pode ser referenciado pelos `pre-run` comandos `pre-build`, `buildpost-build`, e durante o processo de compilação.

Importante: observe que os `pre-run` comandos estão localizados na seção `Executar` desse arquivo, embora possam acessar somente as variáveis de ambiente definidas na seção `Construir`.

- O `env` conjunto na seção `Executar` pode ser referenciado pelo `run` comando no ambiente de execução.

Sintaxe

```
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"
```

A API App Runner

A interface de programação de AWS App Runner aplicativos (API) é uma RESTful API para fazer solicitações ao serviço App Runner. Você pode usar a API para criar, listar, descrever, atualizar e excluir recursos do App Runner em sua Conta da AWS.

Você pode chamar a API diretamente no código do seu aplicativo ou usar um dos AWS SDKs.

Para obter informações completas de referência da API, consulte a [Referência AWS App Runner da API](#).

Para obter mais informações sobre ferramentas para AWS desenvolvedores, consulte [Ferramentas para desenvolver AWS](#).

Tópicos

- [Usando o AWS CLI para trabalhar com o App Runner](#)
- [Usando AWS CloudShell para trabalhar com AWS App Runner](#)

Usando o AWS CLI para trabalhar com o App Runner

Para scripts de linha de comando, use o [AWS CLI](#) para fazer chamadas para o serviço App Runner. Para obter informações AWS CLI de referência completas, consulte o [apprunner](#) na Referência de AWS CLI comandos.

AWS CloudShell permite que você pule a instalação do AWS CLI em seu ambiente de desenvolvimento e, em AWS Management Console vez disso, o use no. Além de evitar a instalação, você também não precisa configurar credenciais e não precisa especificar a região. Sua AWS Management Console sessão fornece esse contexto para AWS CLI o. Para obter mais informações sobre CloudShell e para obter um exemplo de uso, consulte [the section called “Usando AWS CloudShell”](#).

Usando AWS CloudShell para trabalhar com AWS App Runner

AWS CloudShell é um shell pré-autenticado baseado em navegador que você pode iniciar diretamente do. AWS Management Console Você pode executar AWS CLI comandos em AWS serviços (inclusive AWS App Runner) usando seu shell preferido (Bash PowerShell ou Z shell). E você pode fazer isso sem precisar baixar ou instalar ferramentas de linha de comando.

Você [inicia a AWS CloudShell partir do AWS Management Console](#), e AWS as credenciais que você usou para entrar no console estão automaticamente disponíveis em uma nova sessão de shell. Essa pré-autenticação de AWS CloudShell usuários permite que você ignore a configuração de credenciais ao interagir com AWS serviços como o App Runner usando a AWS CLI versão 2 (pré-instalada no ambiente computacional do shell).

Tópicos

- [Obtendo permissões do IAM para AWS CloudShell](#)
- [Interagindo com o App Runner usando AWS CloudShell](#)
- [Verificando seu serviço App Runner usando AWS CloudShell](#)

Obtendo permissões do IAM para AWS CloudShell

Usando os recursos de gerenciamento de acesso fornecidos por AWS Identity and Access Management, os administradores podem conceder permissões aos usuários do IAM para que eles possam acessar AWS CloudShell e usar os recursos do ambiente.

A maneira mais rápida de um administrador conceder acesso aos usuários é por meio de uma política AWS gerenciada. Uma [política gerenciada pela AWS](#) é uma política independente que é criada e administrada pela AWS. A seguinte política AWS gerenciada para CloudShell pode ser anexada às identidades do IAM:

- `AWSCloudShellFullAccess`: concede permissão para uso AWS CloudShell com acesso total a todos os recursos.

Se você quiser limitar o escopo das ações que um usuário do IAM pode realizar AWS CloudShell, crie uma política personalizada que use a política `AWSCloudShellFullAccess` gerenciada como modelo. Para obter mais informações sobre como limitar as ações que estão disponíveis para os usuários em CloudShell, consulte [Gerenciamento de AWS CloudShell acesso e uso com políticas do IAM](#) no Guia do AWS CloudShell usuário.

Note

Sua identidade do IAM também exige uma política que conceda permissão para fazer chamadas para o App Runner. Para obter mais informações, consulte [the section called “App Runner e IAM”](#).

Interagindo com o App Runner usando AWS CloudShell

Depois AWS CloudShell de iniciar a partir do AWS Management Console, você pode começar imediatamente a interagir com o App Runner usando a interface de linha de comando.

No exemplo a seguir, você recupera informações sobre um dos seus serviços do App Runner usando o AWS CLI in. CloudShell

Note

Ao usar o AWS CLI in AWS CloudShell, você não precisa baixar ou instalar nenhum recurso adicional. Além disso, como você já está autenticado no shell, não precisará configurar as credenciais antes de fazer chamadas.

Example Recuperando informações do serviço App Runner usando AWS CloudShell

1. A partir do AWS Management Console, você pode iniciar CloudShell escolhendo as seguintes opções disponíveis na barra de navegação:
 - Escolha o CloudShell ícone.
 - Comece a digitar **cloudshell** na caixa de pesquisa e escolha a CloudShell opção ao vê-la nos resultados da pesquisa.
2. Para listar todos os serviços atuais do App Runner em sua AWS conta na AWS região da sessão do console, digite o seguinte comando na linha de CloudShell comando:

```
$ aws apprunner list-services
```

A saída lista informações resumidas de seus serviços.

```
{
  "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. Para obter uma descrição detalhada de um serviço específico do App Runner, digite o seguinte comando na linha de CloudShell comando, usando um dos ARNs recuperados na etapa anterior:

```

$ aws apprunner describe-service --service-arn arn:aws:apprunner:us-east-2:123456789012:service/my-app-1/8fe1e10304f84fd2b0df550fe98a71fa

```

A saída lista uma descrição detalhada do serviço que você especificou.

```

{
  "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
  }
}

```

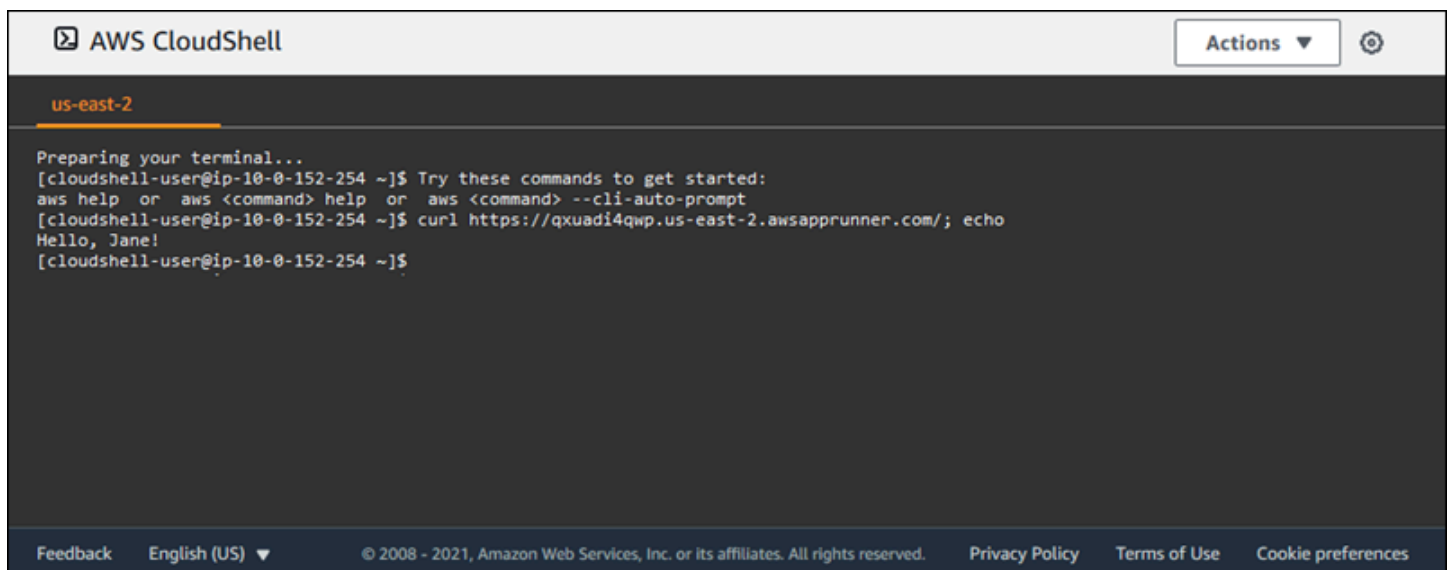
Verificando seu serviço App Runner usando AWS CloudShell

Quando você [cria um serviço do App Runner](#), o App Runner cria um domínio padrão para o site do seu serviço e o mostra no console (ou o retorna no resultado da chamada da API). Você pode usar CloudShell para fazer chamadas para seu site e verificar se ele está funcionando corretamente.

Por exemplo, depois de criar um serviço App Runner conforme descrito em [Conceitos básicos](#), execute o seguinte comando em CloudShell:

```
$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo
```

A saída deve mostrar o conteúdo esperado da página.



The screenshot shows the AWS CloudShell interface. At the top, there's a header bar with the AWS CloudShell logo and an 'Actions' dropdown menu. Below the header, the region 'us-east-2' is displayed. The main terminal area shows the following text:

```
Preparing your terminal...  
[cloudshell-user@ip-10-0-152-254 ~]$ Try these commands to get started:  
aws help or aws <command> help or aws <command> --cli-auto-prompt  
[cloudshell-user@ip-10-0-152-254 ~]$ curl https://qxuadi4qwp.us-east-2.awsapprunner.com/; echo  
Hello, Jane!  
[cloudshell-user@ip-10-0-152-254 ~]$
```

At the bottom of the terminal, there's a footer bar containing links for 'Feedback', 'English (US)', copyright information '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.', and links for 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

Solução de problemas

Este capítulo fornece etapas de solução de problemas para erros e problemas comuns que você pode encontrar ao usar seu AWS App Runner serviço. As mensagens de erro podem aparecer no console, na API ou na guia Registros da sua página de serviço.

Para obter mais orientações sobre solução de problemas e respostas a perguntas comuns de suporte, acesse a [Central de Conhecimento da](#) .

Tópicos

- [Quando o serviço não consegue criar](#)
- [Nomes de domínios personalizados](#)
- [Erro de roteamento de solicitação HTTP/HTTPS](#)
- [Quando o serviço não consegue se conectar ao Amazon RDS ou ao serviço downstream](#)
- [Quando não há endereços IP suficientes para iniciar instâncias ou escalar](#)

Quando o serviço não consegue criar

Se sua tentativa de criar um serviço App Runner falhar, o serviço entrará em um `CREATE_FAILED` status. Esse status aparece como Falha na criação no console. Um serviço pode falhar na criação devido a problemas relacionados a um ou mais dos seguintes:

- Código da sua aplicação
- O processo de construção
- Configuração
- Cotas de recurso
- Problemas temporários com o subjacente Serviços da AWS que seu serviço usa

Para solucionar uma falha na criação de um serviço, recomendamos que você faça o seguinte.

1. Leia os eventos e registros do serviço para descobrir o que causou a falha na criação do serviço.
2. Faça as alterações necessárias em seu código ou configuração.
3. Se você atingiu sua cota de serviço, exclua um ou mais serviços.

4. Se você atingiu outra cota de recursos, talvez consiga aumentá-la se ela for ajustável.
5. Tente reconstruir o serviço novamente depois de concluir todas as etapas acima. Para obter informações sobre como reconstruir seu serviço, consulte [the section called “Serviço com falha na reconstrução”](#).

Note

Uma das cotas de recursos ajustáveis que podem estar causando um problema é o recurso Fargate On-Demand vCPU.

A contagem de recursos da vCPU determina o número de instâncias que o App Runner pode fornecer ao seu serviço. Esse é um valor de cota ajustável para a contagem de recursos da vCPU Fargate On-Demand que reside no serviço. AWS Fargate Para ver as configurações de cota de vCPU para sua conta ou para solicitar um aumento de cota, use o console Service Quotas no. AWS Management Console Para obter mais informações, consulte as [cotas AWS Fargate de serviço](#) no Amazon Elastic Container Service Developer Guide.

Important

Você não incorre em nenhuma cobrança adicional além da tentativa inicial de criação de um serviço com falha. Mesmo que o serviço com falha não seja utilizável, ele ainda conta para sua cota de serviço. O App Runner não exclui automaticamente o serviço com falha, então certifique-se de excluí-lo quando terminar de analisar a falha.

Nomes de domínios personalizados

Esta seção aborda como você pode solucionar problemas e resolver vários erros que podem ocorrer ao se vincular a um domínio personalizado.

Note

[Para aumentar a segurança de seus aplicativos App Runner, o domínio*.awsapprunner.com é registrado na Lista Pública de Sufixos \(PSL\)](#). Para maior segurança, recomendamos que você use cookies com um __Host - prefixo se precisar definir cookies confidenciais no nome de domínio padrão para seus aplicativos App Runner. Essa prática ajudará a defender seu

domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

Obtendo o erro Create Fail para domínio personalizado

- Verifique se esse erro é devido a um problema com os registros CAA. Se não houver registros CAA em nenhum lugar na árvore DNS, você receberá uma mensagem e `fail open` AWS Certificate Manager emitirá um certificado para verificar o domínio personalizado. Isso permite que o App Runner aceite o domínio personalizado. Se você estiver usando certificações CAA nos registros DNS, certifique-se de que pelo menos os registros CAA de um domínio incluam `amazon.com`. Caso contrário, o ACM não emitirá um certificado. Como resultado, o domínio personalizado do App Runner não foi criado.

O exemplo a seguir usa a ferramenta de pesquisa de DNS DiG para mostrar registros CAA sem uma entrada obrigatória. O exemplo usa `example.com` como domínio personalizado. Execute os comandos a seguir no exemplo para verificar os registros do 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...
```

- Corrija os registros do domínio e garanta que pelo menos um registro CAA `amazon.com` inclua.
- Tente vincular novamente o domínio personalizado ao App Runner.

Para obter instruções sobre como resolver erros de CAA, consulte o seguinte:

- [Problemas de Autorização da Autoridade de Certificação \(CAA\)](#)
- [Como faço para resolver erros de CAA ao emitir ou renovar um certificado ACM?](#)

Erro pendente de validação do certificado DNS para domínio personalizado

- Verifique se você pulou uma etapa importante na configuração do domínio personalizado. Além disso, verifique se você configurou incorretamente um registro DNS usando uma ferramenta de pesquisa de DNS, como DiG. Em particular, verifique os seguintes erros:
 - Qualquer etapa perdida.
 - Caracteres não suportados, como aspas duplas nos registros DNS.
- Corrija os erros.
- Tente vincular novamente o domínio personalizado ao App Runner.

Para obter instruções sobre como resolver erros de validação do CAA, consulte o seguinte.

- [Validação de DNS](#)
- [the section called “Nomes de domínios personalizados”](#)

Comandos básicos de solução de problemas

- Confirme se um serviço pode ser encontrado.

```
aws apprunner list-services
```

- Descreva um serviço e verifique seu status.

```
aws apprunner describe-service --service-arn
```

- Verifique o status do domínio personalizado.

```
aws apprunner describe-custom-domains --service-arn
```

- Liste todas as operações em andamento.

```
aws apprunner list-operations --service-arn
```

Renovação de certificado de domínio personalizado

Quando você adiciona um domínio personalizado ao seu serviço, o App Runner fornece um conjunto de registros CNAME que você adiciona ao seu servidor DNS. Esses registros CNAME incluem registros de certificados. O App Runner usa AWS Certificate Manager (ACM) para verificar o domínio. O App Runner valida esses registros DNS para garantir a propriedade contínua desse domínio. Se você remover os registros CNAME da sua zona DNS, o App Runner não poderá mais validar os registros DNS e o certificado de domínio personalizado não será renovado automaticamente.

Esta seção aborda como resolver os seguintes problemas de renovação de certificados de domínio personalizados:

- [the section called “O CNAME é removido do servidor DNS”](#).
- [the section called “O certificado expirou”](#).

O CNAME é removido do servidor DNS

- Recupere seus registros CNAME usando a [DescribeCustomDomains](#) API ou a partir das configurações de domínio personalizado no console do App Runner. Para obter informações sobre o armazenamento CNAMEs, consulte [CertificateValidationRecords](#).
- Adicione os registros CNAME de validação do certificado ao seu servidor DNS. O App Runner pode então validar que você é proprietário do domínio. Depois de adicionar os registros CNAME, pode levar até 30 minutos para que os registros DNS sejam propagados. Também pode levar várias horas para que o App Runner e o ACM tentem novamente o processo de renovação do certificado. Para obter instruções sobre como adicionar registros CNAME, consulte [the section called “Gerenciar domínios personalizados”](#).

O certificado expirou

- Desassocie (desvincule) e depois associe (vincule) o domínio personalizado do seu serviço App Runner usando o console ou a API do App Runner. O App Runner cria novos registros CNAME de validação de certificado.

- Adicione os novos registros CNAME de validação de certificado ao seu servidor DNS.

Para obter instruções sobre como desassociar (desvincular) e associar (vincular) o domínio personalizado, consulte [the section called “Gerenciar domínios personalizados”](#)

Como faço para verificar se o certificado foi renovado com sucesso?

Você pode verificar o status dos registros do certificado para verificar se o certificado foi renovado com sucesso. Você pode verificar o status dos certificados usando ferramentas como curl.

Para obter mais informações sobre a renovação do certificado, consulte os links a seguir:

- [Por que meu certificado ACM está marcado como inelegível para renovação?](#)
- [Renovação gerenciada para certificados ACM](#)
- [Validação de DNS](#)

Erro de roteamento de solicitação HTTP/HTTPS

Esta seção aborda como você pode solucionar problemas e resolver erros que podem ocorrer ao rotear o tráfego HTTP/HTTPS para os endpoints do serviço App Runner.

Erro 404 Não encontrado ao enviar tráfego HTTP/HTTPS para endpoints do serviço App Runner

- Verifique se o Host Header está apontando para a URL do serviço na solicitação HTTP, pois o App Runner usa as informações do cabeçalho do host para rotear as solicitações. A maioria dos clientes, como curl, e os navegadores da Web apontam automaticamente o cabeçalho do host para a URL do serviço. Se seu cliente não definir o URL do serviço como o Host Header, você receberá um 404 Not Found erro.

Exemplo cabeçalho de host incorreto

```
$ ~ curl -I -H "host: foobar.com" https://testservice.awsapprunner.com/  
HTTP/1.1 404 Not Found  
transfer-encoding: chunked
```

Example Cabeçalho correto do host

```
$ ~ 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
```

- Verifique se seu cliente está configurando corretamente o indicador de nome do servidor (SNI) para roteamento de solicitações para serviços públicos ou privados. Para terminação de TLS e roteamento de solicitações, o App Runner usa o SNI definido na conexão HTTPS.

Quando o serviço não consegue se conectar ao Amazon RDS ou ao serviço downstream

Pode haver um problema de configuração de rede com seu serviço se ele não conseguir se conectar a um banco de dados do Amazon RDS ou a outro aplicativo ou serviço downstream. Este tópico explica algumas etapas para determinar se há algum problema com a configuração da rede e as opções para corrigi-lo. Para saber mais sobre a configuração do tráfego de saída para o App Runner, consulte [Habilitando o acesso à VPC para tráfego de saída](#)

Note

Para visualizar a configuração do seu conector VPC, no painel de navegação esquerdo do console do App Runner, selecione Configuração de rede. Em seguida, selecione a guia Tráfego de saída. Selecione um conector VPC. A próxima página exibe detalhes sobre o conector VPC. Nessa página, você pode visualizar e detalhar o seguinte: sub-redes, grupos de segurança e serviços do App Runner que usam a VPC.

Para restringir a causa da incapacidade do seu aplicativo de se conectar a outro serviço downstream

1. Certifique-se de que as sub-redes usadas nos conectores VPC sejam sub-redes privadas. Se um conector estiver configurado com uma sub-rede pública, seu serviço encontrará erros, porque o hiperplano subjacente ENIs (interfaces de rede elásticas) de cada sub-rede não tem um espaço IP público.

Se seus conectores VPC estiverem usando sub-redes públicas, você tem as seguintes opções para corrigir essa configuração:

- a. Crie uma nova sub-rede privada e use-a em vez da sub-rede pública para o conector VPC. Para obter mais informações, consulte [Sub-redes para sua VPC no Guia do usuário](#) da Amazon VPC.
 - b. Roteie a sub-rede pública existente por meio de gateways NAT. Para obter mais informações, consulte os [gateways NAT no Guia](#) do usuário da Amazon VPC.
2. Verifique se as regras de entrada e saída do grupo de segurança para o conector VPC estão corretas. No painel de navegação esquerdo do console do App Runner, selecione Configuração de rede > Tráfego de saída. Selecione o conector VPC na lista. A próxima página lista os grupos de segurança que você pode selecionar para inspecionar.
 3. Verifique se as regras de entrada e saída do grupo de segurança estão corretas para a instância do RDS ou outro serviço downstream ao qual você está tentando se conectar. Para obter mais informações, consulte o guia de serviço do serviço downstream ao qual seu aplicativo App Runner está tentando se conectar.
 4. Para confirmar que não há outro tipo de problema de configuração de rede fora das configurações do App Runner, tente se conectar ao RDS ou ao serviço downstream fora do App Runner:
 - a. Em uma EC2 instância da Amazon na mesma VPC, tente se conectar à instância ou serviço do RDS.
 - b. Se você estiver tentando se conectar a um endpoint de VPC de serviço, verifique a conectividade acessando o mesmo endpoint de uma EC2 instância na mesma VPC.
 5. Se algum dos testes de conexão na Etapa 4 falhar, é mais do que provável que haja um problema fora das configurações do App Runner com outro recurso em sua AWS conta. Entre em contato com o AWS Support para obter assistência para isolar e corrigir ainda mais o problema com suas outras configurações de rede.
 6. Se você se conectar com êxito à instância do RDS ou ao serviço downstream seguindo as instruções na Etapa 4, continue com as instruções nesta etapa. Verificaremos se o tráfego está entrando no ENI ativando e inspecionando os registros de fluxo do Hyperplane ENI.

 Note

Para poder concluir essas etapas e obter as informações necessárias do log de fluxo ENI, a tentativa de conexão com o RDS ou o serviço downstream deve ocorrer após a inicialização bem-sucedida do serviço App Runner. Seu aplicativo deve realizar a operação de conexão ao RDS ou ao serviço downstream quando estiver em um estado em execução. Caso contrário, ENIs isso poderia ser limpo como parte dos fluxos de trabalho de reversão do App Runner. Essa abordagem garante que eles ENIs permaneçam disponíveis para investigações adicionais.

- a. No AWS console, inicie o EC2 console.
- b. No painel de navegação esquerdo, no agrupamento Rede e Segurança, selecione Interfaces de Rede.
- c. Role até as colunas Tipo de interface e Descrição para localizá-las ENIs nas sub-redes associadas ao conector VPC. Eles terão os seguintes padrões de nomenclatura.
 - Tipo de interface: fargate
 - Descrição: começa com AWSAppRunner ENI(exemplo: AWSAppRunner ENI - abcde123-abcd-1234-1234-abcde1233456)
- d. Use as caixas de seleção no início das linhas para selecionar as ENIs que se aplicam.
- e. No menu Ações, selecione Criar registro de fluxo.
- f. Insira as informações nas instruções e selecione Criar flog de fluxo na parte inferior da página.
- g. Inspecione o registro de fluxo gerado.
 - Se o tráfego estava entrando na ENI quando você estava testando a conexão, o problema não está relacionado à configuração da ENI. Pode haver problemas de configuração de rede com outro recurso em sua AWS conta além dos serviços do App Runner. Entre em contato com o AWS Support para obter mais assistência.
 - Se o tráfego não estava entrando na ENI quando você estava testando a conexão, recomendamos que você entre em contato com o AWS Support para ver se há algum problema conhecido com o serviço Fargate.
- h. Use a ferramenta Reachability Analyzer de rede. Essa ferramenta ajuda a determinar configurações incorretas de rede identificando componentes de bloqueio quando uma fonte

no caminho da rede virtual não está acessível. Para obter mais informações, consulte [O que é o Reachability Analyzer?](#) no Guia do Amazon VPC Reachability Analyzer.

Insira a ENI do App Runner como origem e a ENI do RDS como destino.

7. Se você não conseguir restringir ainda mais o problema ou se ainda não conseguir se conectar ao RDS ou ao serviço downstream após concluir as etapas anteriores, recomendamos que entre em contato com o AWS Support para obter mais assistência.

Quando não há endereços IP suficientes para iniciar instâncias ou escalar

Note

Para serviços públicos, o App Runner não cria uma interface de rede elástica (ENI) no seu VPCs, portanto, seus serviços públicos não são afetados por essa alteração.

Este guia ajuda a resolver erros de exaustão de IP que você pode encontrar nos serviços do App Runner com acesso à VPC para tráfego de saída ativado.

O App Runner iniciará instâncias nas sub-redes associadas ao seu conector VPC. O App Runner cria 1 ENI por instância na sub-rede em que sua instância é executada. Cada ENI usa um IP privado nessa sub-rede. As sub-redes têm um número fixo de IPs disponíveis, dependendo do bloco CIDR associado a essa sub-rede. Se o App Runner não conseguir encontrar sub-redes suficientes IPs para criar uma ENI, ele falhará ao iniciar novas instâncias para seu serviço App Runner. Isso pode levar a problemas com a expansão de seus serviços. Nesses casos, você verá os registros de eventos do App Runner indicando que o App Runner não consegue encontrar sub-redes disponíveis. IPs Você pode atualizar seus serviços com as instruções abaixo para resolver esses erros.

Como atualizar seus serviços para ter mais serviços disponíveis IPs

O número de endereços IP disponíveis em uma sub-rede é baseado no bloco CIDR associado a essa sub-rede. Os blocos CIDR associados a uma sub-rede não podem ser atualizados após a criação. Os conectores VPC do App Runner também não podem ser atualizados depois de criados. Para fornecer mais IPs aos seus serviços do App Runner com acesso à VPC para tráfego de saída ativado:

1. Crie novas sub-redes com um bloco CIDR maior.
2. Crie um novo conector VPC com a (s) nova (s) sub-rede (s).
3. Atualize seu serviço App Runner para usar o novo conector VPC.

Cálculo IPs necessário para seus serviços

Antes de tentar criar novas sub-redes com blocos CIDR maiores, determine o número necessário em seus serviços do IPs App Runner. Recomendamos calcular o número IPs necessário em seu conector da seguinte forma:

1. Para cada serviço com acesso à VPC para tráfego de saída ativado, observe o [tamanho máximo \(máximo de instâncias\)](#) na configuração de auto scaling.
2. Some os valores em todos os serviços.
3. Duplique essa soma para contabilizar as novas instâncias lançadas durante implantações azul-esverdeadas.

Exemplo

Considere dois serviços A e B usando o mesmo conector VPC.

1. O serviço A tem o tamanho máximo configurado como 25.
2. O serviço B tem tamanho máximo configurado como 15.

Obrigatório IPs = $2 \times (25 + 15) = 80$

Certifique-se de que suas sub-redes tenham pelo menos 80 disponíveis IPs combinadas.

Criar nova (s) sub-rede (s)

1. Determine o tamanho do bloco CIDR necessário para IPv4 usar essa fórmula (observe que 5 IPs são reservados pela AWS: Dimensionamento de [sub-rede](#))

```
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. Crie uma nova sub-rede usando a AWS EC2 CLI.

```
aws ec2 create-subnet --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Exemplo (cria uma sub-rede com 4.096): IPs

```
aws ec2 create-subnet --vpc-id my-vpc-id --cidr-block 10.0.0.0/20
```

3. Crie um novo conector VPC. Consulte: [Gerenciar o acesso à VPC](#)
4. Atualize seus serviços com tráfego de saída para a VPC habilitado para usar esse novo conector VPC. O App Runner começará a usar as novas sub-redes assim que seu serviço for atualizado.

Note

VPCs também são limitados ao número de disponíveis IPs que podem ser alocados às sub-redes por blocos CIDR. Se você não conseguir criar sub-redes com blocos CIDR maiores, talvez seja necessário atualizar sua VPC com blocos CIDR secundários antes de criar as novas sub-redes.

Anexando blocos CIDR secundários à sua VPC

Associe o bloco CIDR secundário a essa VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id <my-vpc-id> --cidr-block <cidr-block>
```

Exemplo de :

```
aws ec2 associate-vpc-cidr-block --vpc-id my-vpc-id --cidr-block 10.1.0.0/16
```

Verificação

Depois de atualizar seu serviço. Você pode usar o seguinte para realizar a verificação de sua correção

1. Monitore registros de eventos: monitore os [registros](#) de eventos do serviço App Runner para validar que nenhum novo erro de indisponibilidade de IP ou ENI apareça
2. Verifique o escalonamento do serviço:
 1. Amplie totalmente o serviço alterando a contagem mínima de instâncias em sua configuração de escalonamento automático
 2. Verifique se todas as novas instâncias foram iniciadas sem erros relacionados ao IP
 3. Monitore vários eventos de escalabilidade para garantir um desempenho consistente
3. Banner do console: se você estiver usando o AWS Management Console, confirme se o App Runner não exibe mais um banner alertando sobre insuficiente IPs.
4. Utilização de VPC e IP de sub-rede:
 1. Use o painel do VPC ou os comandos da CLI para verificar a utilização do endereço IP em suas novas sub-redes.
 2. Confirme se ainda há uma boa margem de disponibilidade IPs após a expansão do seu serviço

Armadilhas comuns

Ao abordar a exaustão de IP nos serviços do App Runner, esteja ciente desses possíveis problemas:

1. Planejamento inadequado de endereços IP: subestimar as necessidades futuras de IP pode levar a problemas recorrentes de exaustão. Conduza um planejamento completo da capacidade, considerando o potencial crescimento do serviço e os cenários de pico de uso.
2. Ignorando o uso de IP em toda a VPC: lembre-se de que outros serviços da AWS dentro da mesma VPC também consomem endereços IP. Considere os requisitos de IP de todos os serviços ao planejar suas configurações de VPC e sub-rede.
3. Negligenciar a atualização de serviços: depois de criar novas sub-redes ou conectores VPC, certifique-se de atualizar seus serviços do App Runner para usar as novas configurações. Não fazer isso resultará no uso contínuo do intervalo de IP esgotado.

4. Entendendo mal as sobreposições de blocos CIDR: ao adicionar blocos CIDR secundários a uma VPC, certifique-se de que eles não se sobreponham aos blocos existentes. A sobreposição de blocos CIDR pode causar conflitos de roteamento e ambiguidade no endereço IP.
5. Excedendo os limites da VPC: lembre-se de que uma VPC pode ter no máximo 5 blocos CIDR (1 primário e 4 secundário). Planeje a expansão do seu espaço de endereço IP dentro dessas restrições.
6. Ignorando a distribuição AZ da sub-rede: ao criar novas sub-redes, certifique-se de que elas sejam distribuídas em várias zonas de disponibilidade para obter alta disponibilidade e tolerância a falhas.
7. Ignorando os limites de ENI: lembre-se de que há limites para o número ENIs que pode ser anexado às instâncias. Verifique se os limites da sua conta da AWS estão alinhados com o uso planejado da interface de rede.

Ao estar ciente dessas armadilhas, você pode gerenciar com mais eficiência seus recursos de VPC e evitar problemas de exaustão de IP nos serviços do App Runner.

Recursos adicionais

1. [Documentação do AWS VPC](#)
2. [Entendendo os blocos CIDR](#)
3. [Conectores VPC do App Runner](#)

Glossário

1. ENI: Elastic Network Interface, uma interface de rede virtual na AWS.
2. CIDR: Roteamento entre domínios sem classe, um método para alocar endereços IP.
3. Conector VPC: um recurso que permite que o App Runner se conecte à sua VPC.

Segurança no App Runner

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como segurança da nuvem e segurança na nuvem:

- **Segurança da nuvem** — AWS é responsável por proteger a infraestrutura que executa AWS os serviços no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Auditores terceirizados testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de [AWS](#) de . Para saber mais sobre os programas de conformidade que se aplicam AWS App Runner, consulte [AWS Serviços no escopo do programa de conformidade AWS](#) .
- **Segurança na nuvem** — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o App Runner. Os tópicos a seguir mostram como configurar o App Runner para atender aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que ajudam a monitorar e proteger seus recursos do App Runner.

Tópicos

- [Proteção de dados no App Runner](#)
- [Gerenciamento de identidade e acesso para App Runner](#)
- [Registro e monitoramento no App Runner](#)
- [Validação de conformidade para o App Runner](#)
- [Resiliência no App Runner](#)
- [Segurança da infraestrutura em AWS App Runner](#)
- [Usando o App Runner com VPC endpoints](#)
- [Análise de configuração e vulnerabilidade no App Runner](#)
- [Melhores práticas de segurança para o App Runner](#)

Proteção de dados no App Runner

O modelo de [responsabilidade AWS compartilhada modelo](#) se aplica à proteção de dados em AWS App Runner. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as [Data Privacy FAQ](#). Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and RGPD](#) no Blog de segurança da AWS .

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como [trabalhar com CloudTrail trilhas](#) no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para obter mais informações sobre os endpoints FIPS disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sigilosas, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o App Runner ou outro Serviços da AWS usando o console, a API ou AWS SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre

usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Tópicos

- [Proteção de dados usando criptografia](#)
- [Privacidade do tráfego entre redes](#)

Proteção de dados usando criptografia

AWS App Runner lê a fonte do aplicativo (imagem de origem ou código-fonte) de um repositório que você especifica e a armazena para implantação em seu serviço. Para obter mais informações, consulte [Arquitetura e conceitos](#).

A proteção de dados se refere à proteção de dados em trânsito (à medida que viajam de e para o App Runner) e em repouso (enquanto são armazenados em data AWS centers).

Para mais informações sobre a proteção de dados, consulte [the section called “Proteção de dados”](#).

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Criptografia em trânsito

Você pode obter proteção de dados em trânsito de duas maneiras: criptografar a conexão usando Transport Layer Security (TLS) ou usar criptografia do lado do cliente (onde o objeto é criptografado antes de ser enviado). Ambos os métodos são válidos para proteger os dados do aplicativo. Para proteger a conexão, criptografe-a usando TLS sempre que seu aplicativo, seus desenvolvedores e administradores e seus usuários finais enviarem ou receberem objetos. O App Runner configura seu aplicativo para receber tráfego via TLS.

A criptografia do lado do cliente não é um método válido para proteger a imagem ou o código de origem que você fornece ao App Runner para implantação. O App Runner precisa acessar a fonte do seu aplicativo, portanto, ele não pode ser criptografado. Portanto, certifique-se de proteger a conexão entre seu ambiente de desenvolvimento ou implantação e o App Runner.

Criptografia em repouso e gerenciamento de chaves

Para proteger os dados do seu aplicativo em repouso, o App Runner criptografa todas as cópias armazenadas da imagem de origem ou do pacote de origem do aplicativo. Ao criar um serviço App Runner, você pode fornecer um AWS KMS key. Se você fornecer uma, o App Runner usará a chave fornecida para criptografar sua fonte. Se você não fornecer um, o App Runner usa um Chave gerenciada pela AWS em vez disso.

Para obter detalhes sobre os parâmetros de criação do serviço App Runner, consulte [CreateService](#). Para obter informações sobre AWS Key Management Service (AWS KMS), consulte o [Guia do AWS Key Management Service desenvolvedor](#).

Privacidade do tráfego entre redes

O App Runner usa a Amazon Virtual Private Cloud (Amazon VPC) para criar limites entre os recursos em seu aplicativo App Runner e controlar o tráfego entre eles, sua rede local e a Internet. Para obter mais informações sobre a segurança da Amazon VPC, consulte Privacidade do [tráfego entre redes na Amazon VPC no Guia do usuário da Amazon VPC](#).

Para obter informações sobre como associar seu aplicativo App Runner a uma Amazon VPC personalizada, consulte [the section called “Tráfego de saída”](#)

Para obter informações sobre como proteger solicitações para o App Runner usando um VPC endpoint, consulte [the section called “Endpoints da VPC”](#)

Para mais informações sobre a proteção de dados, consulte [the section called “Proteção de dados”](#).

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Gerenciamento de identidade e acesso para App Runner

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) para usar os recursos do App Runner. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Tópicos

- [Público](#)
- [Autenticar com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [Como o App Runner funciona com o IAM](#)
- [Exemplos de políticas baseadas em identidade do App Runner](#)
- [Usando funções vinculadas ao serviço para o App Runner](#)
- [AWS políticas gerenciadas para AWS App Runner](#)
- [Solução de problemas de identidade e acesso do App Runner](#)

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no App Runner.

Usuário do serviço — Se você usa o serviço App Runner para fazer seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do App Runner para fazer seu trabalho, talvez precise de permissões adicionais.

Compreenda como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não conseguir acessar um recurso no App Runner, consulte [Solução de problemas de identidade e acesso do App Runner](#).

Administrador de serviços — Se você é responsável pelos recursos do App Runner em sua empresa, provavelmente tem acesso total ao App Runner. É seu trabalho determinar quais recursos e recursos do App Runner seus usuários do serviço devem acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM com o App Runner, consulte [Como o App Runner funciona com o IAM](#).

Administrador do IAM — Se você for administrador do IAM, talvez queira saber detalhes sobre como criar políticas para gerenciar o acesso ao App Runner. Para ver exemplos de políticas baseadas em identidade do App Runner que você pode usar no IAM, consulte. [Exemplos de políticas baseadas em identidade do App Runner](#)

Autenticar com identidades

A autenticação é como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte [Como fazer login Conta da AWS](#) no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte [Versão 4 do AWS Signature para solicitações de API](#) no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser necessário fornecer informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte [Autenticação multifator](#) no Guia do usuário do AWS IAM Identity Center e [Usar a autenticação multifator da AWS no IAM](#) no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário-raiz para tarefas diárias. Proteja as credenciais do usuário-raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário-raiz, consulte [Tarefas que exigem credenciais de usuário-raiz](#) no Guia do Usuário do IAM.

Usuários e grupos do IAM

Um [usuário do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, é recomendável contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte [Alternar as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo](#) no Guia do Usuário do IAM.

Um [grupo do IAM](#) é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdminse conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte [Casos de uso para usuários do IAM](#) no Guia do usuário do IAM.

Perfis do IAM

Uma [função do IAM](#) é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Para assumir temporariamente uma função do IAM no AWS Management Console, você pode [alternar de um usuário para uma função do IAM \(console\)](#). Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para usar perfis, consulte [Métodos para assumir um perfil](#) no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- **Acesso de usuário federado:** para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para ter mais informações sobre perfis para federação, consulte [Criar um perfil para um provedor de identidade de terceiros \(federação\)](#) no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a

um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte [Conjuntos de Permissões](#) no Guia do Usuário do AWS IAM Identity Center .

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos na Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal da chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
- Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte [Sessões de acesso direto](#).
- Perfil de serviço: um perfil de serviço é um [perfil do IAM](#) que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte [Criar um perfil para delegar permissões a um AWS service \(Serviço da AWS\)](#) no Guia do Usuário do IAM.
- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode presumir o perfil de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.
- Aplicativos em execução na Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma EC2 instância

e fazendo solicitações AWS CLI de AWS API. Isso é preferível a armazenar chaves de acesso na EC2 instância. Para atribuir uma AWS função a uma EC2 instância e disponibilizá-la para todos os aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém a função e permite que os programas em execução na EC2 instância recebam credenciais temporárias. Para obter mais informações, consulte [Usar uma função do IAM para conceder permissões a aplicativos executados em EC2 instâncias da Amazon](#) no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. Você deve [especificar uma entidade principal](#) em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais diretores (membros da conta, usuários ou funções) têm permissões para acessar um recurso. ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o AWS WAF Amazon VPC são exemplos de serviços que oferecem suporte. ACLs Para saber mais ACLs, consulte a [visão geral da lista de controle de acesso \(ACL\)](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade.

As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.

- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as suas contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada um Usuário raiz da conta da AWS. Para obter mais informações sobre Organizations e SCPs, consulte [Políticas de controle de serviços](#) no Guia AWS Organizations do Usuário.
- Políticas de controle de recursos (RCPs) — RCPs são políticas JSON que você pode usar para definir o máximo de permissões disponíveis para recursos em suas contas sem atualizar as políticas do IAM anexadas a cada recurso que você possui. O RCP limita as permissões para recursos nas contas dos membros e pode afetar as permissões efetivas para identidades, incluindo a Usuário raiz da conta da AWS, independentemente de pertencerem à sua organização. Para obter mais informações sobre Organizations e RCPs, incluindo uma lista Serviços da AWS desse suporte RCPs, consulte [Políticas de controle de recursos \(RCPs\)](#) no Guia AWS Organizations do usuário.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte [Lógica de avaliação de políticas](#) no Guia do usuário do IAM.

Como o App Runner funciona com o IAM

Antes de usar o IAM para gerenciar o acesso AWS App Runner, você deve entender quais recursos do IAM estão disponíveis para uso com o App Runner. Para ter uma visão geral de como o App Runner e outros AWS serviços funcionam com o IAM, consulte [AWS Serviços que funcionam com o IAM no Guia do](#) usuário do IAM.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Tópicos

- [Políticas baseadas em identidade do App Runner](#)
- [Políticas baseadas em recursos do App Runner](#)
- [Autorização com base nas tags do App Runner](#)
- [Permissões de usuário do App Runner](#)
- [Funções do App Runner IAM](#)

Políticas baseadas em identidade do App Runner

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. O App Runner oferece suporte a ações, recursos e chaves de condição específicos. Para conhecer todos os elementos usados em uma política JSON, consulte [Referência de elementos de política JSON do IAM](#) no Guia do usuário do IAM.

Ações

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de política no App Runner usam o seguinte prefixo antes da ação: `apprunner:` Por exemplo, para conceder permissão a alguém para executar uma EC2 instância da Amazon com

a operação de EC2 RunInstances API da Amazon, você inclui a `ec2:RunInstances` ação na política dessa pessoa. As instruções de política devem incluir um elemento `Action` ou `NotAction`. O App Runner define seu próprio conjunto de ações que descrevem as tarefas que você pode realizar com esse serviço.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": [  
    "apprunner:CreateService",  
    "apprunner:CreateConnection"  
]
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "apprunner:Describe*"
```

Para ver uma lista das ações do App Runner, consulte [Ações definidas por AWS App Runner](#) na Referência de Autorização de Serviço.

Recursos

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu [nome do recurso da Amazon \(ARN\)](#). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Os recursos do App Runner têm a seguinte estrutura de ARN:

```
arn:aws:apprunner:region:account-id:resource-type/resource-name[/resource-id]
```


Para obter mais informações sobre o formato de ARNs, consulte [Amazon Resource Names \(ARNs\) e AWS Service Namespaces](#) no. Referência geral da AWS

Por exemplo, para especificar o my-service serviço em sua declaração, use o seguinte ARN:

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/my-service"
```

Para especificar todos os serviços que pertencem a uma conta específica, use o caractere curinga (*):

```
"Resource": "arn:aws:apprunner:us-east-1:123456789012:service/*"
```

Algumas ações do App Runner, como aquelas para criar recursos, não podem ser executadas em um recurso específico. Nesses casos, você deve utilizar o caractere curinga (*).

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do App Runner e seus ARNs, consulte [Recursos definidos por AWS App Runner](#) na Referência de Autorização de Serviço. Para saber com quais ações é possível especificar o ARN de cada atributo, consulte [Ações definidas pelo AWS App Runner](#).

Chaves de condição

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento Condition (ou bloco Condition) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento Condition é opcional. É possível criar expressões condicionais que usem [agentes de condição](#), como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos de Condition em uma declaração ou várias chaves em um único elemento de Condition, a AWS os avaliará usando uma operação lógica AND. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma OR operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um recurso somente se ele estiver

marcado com seu nome de usuário do IAM. Para obter mais informações, consulte [Elementos da política do IAM: variáveis e tags](#) no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as [chaves de contexto de condição AWS global](#) no Guia do usuário do IAM.

O App Runner suporta o uso de algumas chaves de condição globais. Para ver todas as chaves de condição AWS globais, consulte [Chaves de contexto de condição AWS global](#) no Guia do usuário do IAM.

O App Runner define um conjunto de chaves de condição específicas do serviço. Além disso, o App Runner oferece suporte ao controle de acesso baseado em tags, que é implementado usando chaves de condição. Para obter detalhes, consulte [the section called “Autorização com base nas tags do App Runner”](#).

Para ver uma lista das chaves de condição do App Runner, consulte [Chaves de condição AWS App Runner](#) na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte [Ações definidas por AWS App Runner](#).

Exemplos

Para ver exemplos de políticas baseadas em identidade do App Runner, consulte. [Exemplos de políticas baseadas em identidade do App Runner](#)

Políticas baseadas em recursos do App Runner

O App Runner não oferece suporte a políticas baseadas em recursos.

Autorização com base nas tags do App Runner

Você pode anexar tags aos recursos do App Runner ou passar tags em uma solicitação para o App Runner. Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as `apprunner:ResourceTag/key-name`, `aws:RequestTag/key-name` ou chaves de condição `aws:TagKeys`. Para obter mais informações sobre a marcação de recursos do App Runner, consulte. [the section called “Configuração”](#)

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um recurso baseado em tags desse recurso, consulte [Controle do acesso aos serviços do App Runner com base em tags](#).

Permissões de usuário do App Runner

Para usar o App Runner, os usuários do IAM precisam de permissões para as ações do App Runner. Uma forma comum de conceder permissões aos usuários é anexar uma política aos usuários ou grupos do IAM. Para obter mais informações sobre o gerenciamento de permissões de usuário, consulte [Alteração de permissões para um usuário do IAM](#) no Guia do usuário do IAM.

O App Runner fornece duas políticas gerenciadas que você pode anexar aos seus usuários.

- `AWSAppRunnerReadOnlyAccess`— Concede permissões para listar e visualizar detalhes sobre os recursos do App Runner.
- `AWSAppRunnerFullAccess`— Concede permissões para todas as ações do App Runner.

Para um controle mais granular das permissões do usuário, você pode criar uma política personalizada e anexá-la aos seus usuários. Para obter detalhes, consulte [Criação de políticas do IAM](#) no Guia do usuário do IAM.

Para obter exemplos de políticas de usuário, consulte [the section called “Políticas de usuário”](#).

`AWSAppRunnerReadOnlyAccess`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*"
      ],
      "Resource": "*"
    }
  ]
}
```

`AWSAppRunnerFullAccess`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": [
        "arn:aws:iam::*:role/aws-service-role/apprunner.amazonaws.com/
AWSServiceRoleForAppRunner",
        "arn:aws:iam::*:role/aws-service-role/networking.apprunner.amazonaws.com/
AWSServiceRoleForAppRunnerNetworking"
    ],
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": [
                "apprunner.amazonaws.com",
                "networking.apprunner.amazonaws.com"
            ]
        }
    }
},
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "iam:PassedToService": "apprunner.amazonaws.com"
        }
    }
},
{
    "Sid": "AppRunnerAdminAccess",
    "Effect": "Allow",
    "Action": "apprunner:*",
    "Resource": "*"
}
]
}

```

Funções do App Runner IAM

Uma [função do IAM](#) é uma entidade dentro da sua Conta da AWS que tem permissões específicas.

Perfis vinculados ao serviço

[As funções vinculadas ao serviço](#) permitem que AWS os serviços acessem recursos em outros serviços para concluir uma ação em seu nome. Os perfis vinculados a serviço aparecem em sua

conta do IAM e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados a serviço.

O App Runner oferece suporte a funções vinculadas a serviços. Para obter informações sobre como criar ou gerenciar funções vinculadas ao serviço do App Runner, consulte [the section called “Uso de perfis vinculados ao serviço”](#)

Perfis de serviço

Esse atributo permite que um serviço assuma um [perfil de serviço](#) em seu nome. O perfil permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e são de propriedade da conta. Isso significa que um usuário do IAM pode alterar as permissões para essa função. Porém, fazer isso pode alterar a funcionalidade do serviço.

O App Runner oferece suporte a algumas funções de serviço.

Função de acesso

A função de acesso é uma função que o App Runner usa para acessar imagens no Amazon Elastic Container Registry (Amazon ECR) em sua conta. É necessário acessar uma imagem no Amazon ECR e não no Amazon ECR Public. Antes de criar um serviço com base em uma imagem no Amazon ECR, use o IAM para criar uma função de serviço e usar a política `AWSAppRunnerServicePolicyForECRAccess` gerenciada nela. Em seguida, você pode passar essa função para o App Runner ao chamar a [CreateService](#) API no [AuthenticationConfiguration](#) membro do [SourceConfiguration](#) parâmetro ou ao usar o console do App Runner para criar um serviço.

`AWSAppRunnerServicePolicyForECRAccess`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetAuthorizationToken"
      ]
    }
  ]
}
```

```
    ],  
    "Resource": "*"    
  }  
]  
}
```

Note

Se você criar sua própria política personalizada para sua função de acesso, não se esqueça de especificar "Resource": "*" para a `ecr:GetAuthorizationToken` ação. Os tokens podem ser usados para acessar qualquer registro do Amazon ECR ao qual você tenha acesso.

Ao criar sua função de acesso, não se esqueça de adicionar uma política de confiança que declare o diretor do serviço App Runner `build.apprunner.amazonaws.com` como uma entidade confiável.

Política de confiança para uma função de acesso

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "build.apprunner.amazonaws.com"  
      },  
      "Action": "sts:AssumeRole"  
    }  
  ]  
}
```

Se você usar o console do App Runner para criar um serviço, o console poderá criar automaticamente uma função de acesso para você e escolhê-la para o novo serviço. O console também lista outras funções em sua conta, e você pode selecionar uma função diferente, se quiser.

Função da instância

O papel da instância é um papel opcional que o App Runner usa para fornecer permissões às ações de AWS serviço que as instâncias de computação do seu serviço precisam. Você precisa

fornece uma função de instância ao App Runner se o código do aplicativo chamar AWS actions (APIs). Incorpore as permissões necessárias na função da instância ou crie sua própria política personalizada e use-a na função da instância. Não temos como prever quais chamadas seu código usa. Portanto, não fornecemos uma política gerenciada para essa finalidade.

Antes de criar um serviço App Runner, use o IAM para criar uma função de serviço com as políticas personalizadas ou incorporadas necessárias. Em seguida, você pode passar essa função para o App Runner como a função da instância ao chamar a [CreateService](#) API no `InstanceRoleArn` membro do [InstanceConfiguration](#) parâmetro ou ao usar o console do App Runner para criar um serviço.

Ao criar sua função de instância, não se esqueça de adicionar uma política de confiança que declare o diretor do serviço App Runner `tasks.apprunner.amazonaws.com` como uma entidade confiável.

Política de confiança para uma função de instância

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "tasks.apprunner.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Se você usar o console do App Runner para criar um serviço, o console listará as funções em sua conta e você poderá selecionar a função que criou para essa finalidade.

Para obter informações sobre a criação de um serviço, consulte [the section called “Criação”](#).

Exemplos de políticas baseadas em identidade do App Runner

Por padrão, os usuários e funções do IAM não têm permissão para criar ou modificar AWS App Runner recursos. Eles também não podem realizar tarefas usando a AWS API, AWS Management Console, AWS CLI, ou. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos

especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos do IAM que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte [Criar políticas na guia JSON](#) no Manual do usuário do IAM.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Tópicos

- [Práticas recomendadas de política](#)
- [Políticas de usuário](#)
- [Controle do acesso aos serviços do App Runner com base em tags](#)

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do App Runner em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte [Políticas gerenciadas pela AWS](#) ou [Políticas gerenciadas pela AWS para funções de trabalho](#) no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte [Políticas e permissões no IAM](#) no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS

CloudFormation. Para obter mais informações, consulte [Elementos da política JSON do IAM: condição](#) no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para obter mais informações, consulte [Validação de políticas do IAM Access Analyzer](#) no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte [Configuração de acesso à API protegido por MFA](#) no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.

Políticas de usuário

Para acessar o console do App Runner, os usuários do IAM devem ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do App Runner em seu Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas exigidas, o console não funcionará conforme planejado para os usuários com essa política.

O App Runner fornece duas políticas gerenciadas que você pode anexar aos seus usuários.

- `AWSAppRunnerReadOnlyAccess`— Concede permissões para listar e visualizar detalhes sobre os recursos do App Runner.
- `AWSAppRunnerFullAccess`— Concede permissões para todas as ações do App Runner.

Para garantir que os usuários possam usar o console do App Runner, anexe, no mínimo, a política `AWSAppRunnerReadOnlyAccess` gerenciada aos usuários. Em vez disso, você pode anexar a política `AWSAppRunnerFullAccess` gerenciada ou adicionar permissões adicionais específicas para permitir que os usuários criem, modifiquem e excluam recursos. Para obter mais informações, consulte [Adicionar permissões a um usuário](#) no Guia do usuário do IAM.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente às ações que correspondam à operação de API que você deseja permitir que os usuários realizem.

Os exemplos a seguir demonstram políticas de usuário personalizadas. Você pode usá-los como pontos de partida para definir suas próprias políticas de usuário personalizadas. Copie o exemplo e/ou remova ações, defina o escopo dos recursos e adicione condições.

Exemplo: política de usuário de gerenciamento de console e conexão

Este exemplo de política permite o acesso ao console e permite a criação e o gerenciamento de conexões. Ele não permite a criação e o gerenciamento do serviço App Runner. Ele pode ser anexado a um usuário cuja função é gerenciar o acesso do serviço App Runner aos ativos do código-fonte.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apprunner:List*",
        "apprunner:Describe*",
        "apprunner:CreateConnection",
        "apprunner>DeleteConnection"
      ],
      "Resource": "*"
    }
  ]
}
```

Exemplo: políticas de usuário que usam chaves de condição

Os exemplos nesta seção demonstram permissões condicionais que dependem de algumas propriedades de recursos ou parâmetros de ação.

Este exemplo de política permite criar um serviço App Runner, mas nega o uso de uma conexão chamada. prod

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "AllowCreateAppRunnerServiceWithNonProdConnections",
    "Effect": "Allow",
    "Action": "apprunner:CreateService",
    "Resource": "*",
    "Condition": {
      "ArnNotLike": {
        "apprunner:ConnectionArn": "arn:aws:apprunner:*:*:connection/prod/*"
      }
    }
  }
]
}

```

Este exemplo de política permite atualizar um serviço do App Runner nomeado preprod somente com uma configuração de escalonamento automático chamada. preprod

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatePreProdAppRunnerServiceWithPreProdASConfig",
      "Effect": "Allow",
      "Action": "apprunner:UpdateService",
      "Resource": "arn:aws:apprunner:*:*:service/preprod/*",
      "Condition": {
        "ArnLike": {
          "apprunner:AutoScalingConfigurationArn":
            "arn:aws:apprunner:*:*:autoscalingconfiguration/preprod/*"
        }
      }
    }
  ]
}

```

Controle do acesso aos serviços do App Runner com base em tags

Você pode usar condições em sua política baseada em identidade para controlar o acesso aos recursos do App Runner com base em tags. Este exemplo mostra como você pode criar uma política que permita excluir um serviço do App Runner. No entanto, a permissão será concedida somente

se a tag `Owner` tiver o valor do nome desse usuário. Essa política também concede as permissões necessárias concluir essa ação no console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListServicesInConsole",
      "Effect": "Allow",
      "Action": "apprunner:ListServices",
      "Resource": "*"
    },
    {
      "Sid": "DeleteServiceIfOwner",
      "Effect": "Allow",
      "Action": "apprunner:DeleteService",
      "Resource": "arn:aws:apprunner:*:*:service/*",
      "Condition": {
        "StringEquals": {"apprunner:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

É possível anexar essa política aos usuários do IAM na sua conta. Se um usuário chamado `richard-roe` tentar excluir um serviço do App Runner, o serviço deverá ser marcado como `Owner=richard-roe` ou `owner=richard-roe`. Caso contrário, ele terá o acesso negado. A chave da tag de condição `Owner` corresponde a `Owner` e a `owner` porque os nomes das chaves de condição não fazem distinção entre maiúsculas e minúsculas. Para obter mais informações, consulte [IAM JSON Policy Elements: Condition](#) (Elementos da política JSON do IAM: Condição) no Guia do usuário do IAM.

Usando funções vinculadas ao serviço para o App Runner

AWS App Runner usa funções [vinculadas ao serviço AWS Identity and Access Management](#) (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao App Runner. As funções vinculadas ao serviço são predefinidas pelo App Runner e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Tópicos

- [Usando funções para gerenciamento](#)

- [Usando funções para networking](#)

Usando funções para gerenciamento

AWS App Runner usa funções [vinculadas ao serviço AWS Identity and Access Management](#) (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao App Runner. As funções vinculadas ao serviço são predefinidas pelo App Runner e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Uma função vinculada ao serviço facilita a configuração do App Runner porque você não precisa adicionar manualmente as permissões necessárias. O App Runner define as permissões de suas funções vinculadas ao serviço e, a menos que seja definido de outra forma, somente o App Runner pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos do App Runner porque você não pode remover inadvertidamente a permissão para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com perfis vinculados a serviços, consulte [Serviços da AWS compatíveis com o IAM](#) e procure os serviços que contenham Sim na coluna Service-Linked Role. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculadas ao serviço para App Runner

O App Runner usa a função vinculada ao serviço chamada. `AWSServiceRoleForAppRunner`

A função permite que o App Runner execute as seguintes tarefas:

- Envie registros para grupos de CloudWatch registros do Amazon Logs.
- Crie regras do Amazon CloudWatch Events para assinar os push de imagens do Amazon Elastic Container Registry (Amazon ECR).
- Envie informações de rastreamento para. AWS X-Ray

A função `AWSServiceRoleForAppRunner` vinculada ao serviço confia nos seguintes serviços para assumir a função:

- `apprunner.amazonaws.com`

As políticas de permissões da função AWSService RoleForAppRunner vinculada ao serviço contêm todas as permissões que o App Runner precisa para concluir ações em seu nome.

AppRunnerServiceRolePolicy política gerenciada

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:PutRetentionPolicy"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:log-group:/aws/apprunner/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/apprunner/*:log-stream:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events>DeleteRule",
        "events:RemoveTargets",
        "events:DescribeRule",
        "events:EnableRule",
        "events:DisableRule"
      ],
      "Resource": "arn:aws:events:*:*:rule/AWSAppRunnerManagedRule*"
    }
  ]
}
```

Política de rastreamento por raio-X

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "xray:PutTraceSegments",
        "xray:PutTelemetryRecords",
        "xray:GetSamplingRules",
        "xray:GetSamplingTargets",
        "xray:GetSamplingStatisticSummaries"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para mais informações, consulte [Permissões de perfil vinculado ao serviço](#) no Guia do usuário do IAM.

Criação de uma função vinculada ao serviço para o App Runner

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria um serviço App Runner na AWS Management Console, na ou na AWS API AWS CLI, o App Runner cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um serviço App Runner, o App Runner cria a função vinculada ao serviço para você novamente.

Editando uma função vinculada ao serviço para o App Runner

O App Runner não permite que você edite a função vinculada ao AWSService RoleForAppRunner serviço. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte [Editar uma função vinculada a serviço](#) no Guia do usuário do IAM.

Excluindo uma função vinculada ao serviço para o App Runner

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar seu perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir um perfil vinculado ao serviço, você deverá excluir qualquer recurso usado pelo perfil.

No App Runner, isso significa excluir todos os serviços do App Runner em sua conta. Para saber mais sobre como excluir os serviços do App Runner, consulte [the section called “Exclusão”](#)

Note

Se o serviço App Runner estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSService RoleForAppRunner vinculada ao serviço. Para obter mais informações, consulte [Excluir um perfil vinculado ao serviço](#) no Guia do usuário do IAM.

Regiões suportadas para funções vinculadas ao serviço App Runner

O App Runner oferece suporte ao uso de funções vinculadas ao serviço em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte [endpoints e cotas do AWS App Runner](#) na Referência geral da AWS.

Usando funções para networking

AWS App Runner usa funções [vinculadas ao serviço AWS Identity and Access Management](#) (IAM). Uma função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao App Runner. As funções vinculadas ao serviço são predefinidas pelo App Runner e incluem todas as permissões que o serviço exige para chamar outros AWS serviços em seu nome.

Uma função vinculada ao serviço facilita a configuração do App Runner porque você não precisa adicionar manualmente as permissões necessárias. O App Runner define as permissões de suas funções vinculadas ao serviço e, a menos que seja definido de outra forma, somente o App Runner pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos do App Runner porque você não pode remover inadvertidamente a permissão para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com perfis vinculados a serviços, consulte [Serviços da AWS compatíveis com o IAM](#) e procure os serviços que contenham Sim na coluna Service-Linked Role. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculadas ao serviço para App Runner

O App Runner usa a função vinculada ao serviço chamada.

`AWSServiceRoleForAppRunnerNetworking`

A função permite que o App Runner execute as seguintes tarefas:

- Conecte uma VPC ao seu serviço App Runner e gerencie as interfaces de rede.

A função `AWSServiceRoleForAppRunnerNetworking` vinculada ao serviço confia nos seguintes serviços para assumir a função:

- `networking.apprunner.amazonaws.com`

A política de permissões de função nomeada `AppRunnerNetworkingServiceRolePolicy` contém todas as permissões que o App Runner precisa para concluir ações em seu nome.

`AppRunnerNetworkingServiceRolePolicy`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
```

```

        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "ec2:CreateNetworkInterface",
    "Resource": "*",
    "Condition": {
        "ForAllValues:StringEquals": {
            "aws:TagKeys": [
                "AWSAppRunnerManaged"
            ]
        }
    }
},
{
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
        "StringEquals": {
            "ec2:CreateAction": "CreateNetworkInterface"
        },
        "StringLike": {
            "aws:RequestTag/AWSAppRunnerManaged": "*"
        }
    }
},
{
    "Effect": "Allow",
    "Action": "ec2:DeleteNetworkInterface",
    "Resource": "*",
    "Condition": {
        "Null": {
            "ec2:ResourceTag/AWSAppRunnerManaged": "false"
        }
    }
}
]

```

```
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para mais informações, consulte [Permissões de perfil vinculado ao serviço](#) no Guia do usuário do IAM.

Criação de uma função vinculada ao serviço para o App Runner

Não é necessário criar manualmente um perfil vinculado ao serviço. Quando você cria um conector VPC na, na ou na AWS API AWS Management Console AWS CLI, o App Runner cria a função vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um conector VPC, o App Runner cria a função vinculada ao serviço para você novamente.

Editando uma função vinculada ao serviço para o App Runner

O App Runner não permite que você edite a função vinculada ao AWSService RoleForAppRunnerNetworking serviço. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte [Editar uma função vinculada a serviço](#) no Guia do usuário do IAM.

Excluindo uma função vinculada ao serviço para o App Runner

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar seu perfil vinculado ao serviço para excluí-la manualmente.

Limpar uma função vinculada ao serviço

Antes de usar o IAM para excluir um perfil vinculado ao serviço, você deverá excluir qualquer recurso usado pelo perfil.

No App Runner, isso significa desassociar os conectores VPC de todos os serviços do App Runner em sua conta e excluir os conectores VPC. Para obter mais informações, consulte [the section called “Tráfego de saída”](#).

 Note

Se o serviço App Runner estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Excluir manualmente a função vinculada ao serviço

Use o console do IAM AWS CLI, o ou a AWS API para excluir a função AWSServiceRoleForAppRunnerNetworking vinculada ao serviço. Para obter mais informações, consulte [Excluir um perfil vinculado ao serviço](#) no Guia do usuário do IAM.

Regiões suportadas para funções vinculadas ao serviço App Runner

O App Runner oferece suporte ao uso de funções vinculadas ao serviço em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte [endpoints e cotas do AWS App Runner](#) na Referência geral da AWS.

AWS políticas gerenciadas para AWS App Runner

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as [políticas gerenciadas pelo cliente](#) que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para obter mais informações, consulte [Políticas gerenciadas pela AWS](#) no Guia do usuário do IAM.

Atualizações do App Runner nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas do App Runner desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações nessa página, assine o feed RSS na página de histórico de documentos do App Runner.

Alteração	Descrição	Data
AWSAppRunnerReadOnlyAccess – Nova política	O App Runner adicionou uma nova política para permitir que os usuários listem e visualizem detalhes sobre os recursos do App Runner.	24 de fevereiro de 2022
AWSAppRunnerFullAccess : atualizar para uma política existente	O App Runner atualizou a lista de recursos da <code>iam::CreateServiceLinkedRole</code> ação para permitir a criação da função <code>AWSServiceRoleForAppRunnerNetworking</code> vinculada ao serviço.	8 de fevereiro de 2022
AppRunnerNetworkingServiceRolePolicy – Nova política	O App Runner adicionou uma nova política para permitir que o App Runner faça chamadas para a Amazon Virtual Private Cloud para anexar uma VPC ao seu serviço App Runner e gerenciar interfaces de rede em nome dos serviços do App Runner. A política é usada na função <code>AWSServiceRoleForAppRunnerNetworking</code> vinculada ao serviço.	8 de fevereiro de 2022
AWSAppRunnerFullAccess – Nova política	O App Runner adicionou uma nova política para permitir que os usuários realizem todas as ações do App Runner.	10 de janeiro de 2022

Alteração	Descrição	Data
AppRunnerServiceRolePolicy – Nova política	O App Runner adicionou uma nova política para permitir que o App Runner faça chamadas para o Amazon CloudWatch Logs e o Amazon CloudWatch Events em nome dos serviços do App Runner. A política é usada na função <code>AWSServiceRoleForAppRunner</code> vinculada ao serviço.	1 de março de 2021
AWSAppRunnerServicePolicyForECRAccess – Nova política	O App Runner adicionou uma nova política para permitir que o App Runner acesse imagens do Amazon Elastic Container Registry (Amazon ECR) em sua conta.	1 de março de 2021
O App Runner começou a monitorar as alterações	O App Runner começou a monitorar as mudanças em suas políticas AWS gerenciadas.	1 de março de 2021

Solução de problemas de identidade e acesso do App Runner

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com AWS App Runner um IAM.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Tópicos

- [Não estou autorizado a realizar uma ação no App Runner](#)
- [Quero permitir que pessoas fora da minha acessem meus Recursos da AWS no App Runner](#)

Não estou autorizado a realizar uma ação no App Runner

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. Seu administrador é a pessoa que forneceu suas credenciais de AWS login.

O exemplo de erro a seguir ocorre quando um usuário do IAM chamado `marymajor` tenta usar o console para ver detalhes sobre um serviço do App Runner, mas não tem `apprunner:DescribeService` permissões.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
apprunner:DescribeService on resource: my-example-service
```

Nesse caso, Mary pede ao administrador que atualize suas políticas para permitir que ela acesse o *my-example-service* recurso usando a `apprunner:DescribeService` ação.

Quero permitir que pessoas fora da minha acessem meus Conta da AWS recursos do App Runner

É possível criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o App Runner é compatível com esses recursos, consulte [Como o App Runner funciona com o IAM](#).
- Para saber como fornecer acesso aos seus recursos em todas as Contas da AWS que você possui, consulte Como [fornecer acesso a um usuário do IAM em outra Conta da AWS que você possui](#) no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como [fornecer acesso Contas da AWS a terceiros](#) no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte [Conceder acesso a usuários autenticados externamente \(federação de identidades\)](#) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

Registro e monitoramento no App Runner

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do seu AWS App Runner serviço. A coleta de dados de monitoramento de todas as

partes da AWS solução permite que você depure com mais facilidade uma falha, caso ela ocorra. O App Runner se integra a várias AWS ferramentas para monitorar seus serviços do App Runner e responder a possíveis incidentes.

CloudWatch Alarmes da Amazon

Com CloudWatch os alarmes da Amazon, você pode observar uma métrica de serviço durante um período de tempo especificado por você. Se a métrica exceder um determinado limite por um determinado número de períodos, você receberá uma notificação.

O App Runner coleta uma variedade de métricas sobre o serviço como um todo e as instâncias (unidades de escalabilidade) que executam seu serviço web. Para obter mais informações, consulte [Métricas \(CloudWatch\)](#).

Logs de aplicações

O App Runner coleta a saída do código do seu aplicativo e a transmite para o Amazon Logs. CloudWatch O que está nessa saída depende de você. Por exemplo, você pode incluir registros detalhados das solicitações feitas ao seu serviço web. Esses registros de log podem ser úteis em auditorias de segurança e acesso. Para obter mais informações, consulte [Registros \(CloudWatch Registros\)](#).

AWS CloudTrail registros de ações

O App Runner é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou AWS serviço no App Runner. CloudTrail captura todas as chamadas de API para o App Runner como eventos. Você pode visualizar os eventos mais recentes no CloudTrail console e criar uma trilha para permitir a entrega contínua de CloudTrail eventos para um bucket do Amazon Simple Storage Service (Amazon S3). Para obter mais informações, consulte [Ações de API \(CloudTrail\)](#).

Validação de conformidade para o App Runner

Audidores terceirizados avaliam a segurança e a conformidade AWS App Runner como parte de vários programas de AWS conformidade. Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte [Serviços da AWS Escopo por Programa de Conformidade](#) [Serviços da AWS](#) e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de [AWS conformidade Programas AWS](#) de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte [Baixar relatórios em AWS Artifact](#).

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. AWS fornece os seguintes recursos para ajudar na conformidade:

- [Governança e conformidade de segurança](#): esses guias de implementação de solução abordam considerações sobre a arquitetura e fornecem etapas para implantar recursos de segurança e conformidade.
- [Referência de serviços qualificados para HIPAA](#): lista os serviços qualificados para HIPAA. Nem todos Serviços da AWS são elegíveis para a HIPAA.
- AWS Recursos de <https://aws.amazon.com/compliance/resources/> de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- [AWS Guias de conformidade do cliente](#) — Entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as melhores práticas de proteção Serviços da AWS e mapeiam as diretrizes para controles de segurança em várias estruturas (incluindo o Instituto Nacional de Padrões e Tecnologia (NIST), o Conselho de Padrões de Segurança do Setor de Cartões de Pagamento (PCI) e a Organização Internacional de Padronização (ISO)).
- [Avaliação de recursos com regras](#) no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- [AWS Security Hub](#) — Isso AWS service (Serviço da AWS) fornece uma visão abrangente do seu estado de segurança interno AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a [Referência de controles do Security Hub](#).
- [Amazon GuardDuty](#) — Isso AWS service (Serviço da AWS) detecta possíveis ameaças às suas cargas de trabalho Contas da AWS, contêineres e dados monitorando seu ambiente em busca de atividades suspeitas e maliciosas. GuardDuty pode ajudá-lo a atender a vários requisitos de conformidade, como o PCI DSS, atendendo aos requisitos de detecção de intrusões exigidos por determinadas estruturas de conformidade.
- [AWS Audit Manager](#) — Isso AWS service (Serviço da AWS) ajuda você a auditar continuamente seu AWS uso para simplificar a forma como você gerencia o risco e a conformidade com as regulamentações e os padrões do setor.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Resiliência no App Runner

A infraestrutura AWS global é construída em torno Regiões da AWS de zonas de disponibilidade. Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte [Infraestrutura AWS global](#).

AWS App Runner gerencia e automatiza o uso da infraestrutura AWS global em seu nome. Ao usar o App Runner, você se beneficia dos mecanismos de disponibilidade e tolerância a falhas oferecidos AWS .

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Segurança da infraestrutura em AWS App Runner

Como serviço gerenciado, AWS App Runner é protegido pelos procedimentos AWS globais de segurança de rede descritos no whitepaper [Amazon Web Services: Visão geral dos processos de segurança](#).

Você usa chamadas de API AWS publicadas para acessar o App Runner pela rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.2 ou posterior. Os clientes também devem ter compatibilidade com conjuntos de criptografia com perfect forward secrecy (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece compatibilidade com esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou é possível usar o [AWS Security Token Service](#) (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Usando o App Runner com VPC endpoints

Seu AWS aplicativo pode integrar AWS App Runner serviços com outros Serviços da AWS que são executados em uma VPC a partir da [Amazon Virtual Private Cloud](#) (Amazon VPC). Partes do seu aplicativo podem fazer solicitações ao App Runner de dentro da VPC. Por exemplo, você pode usar AWS CodePipeline para implantar continuamente em seu serviço App Runner. Uma forma de melhorar a segurança do seu aplicativo é enviar essas solicitações do App Runner (e solicitações para outros Serviços da AWS) por meio de um VPC endpoint.

Usando um VPC endpoint, você pode conectar de forma privada sua VPC a serviços de endpoint de VPC compatíveis Serviços da AWS e baseados em. AWS PrivateLink Você não precisa de um gateway de internet, dispositivo NAT, conexão VPN ou AWS Direct Connect conexão.

Os recursos em sua VPC não usam endereços IP públicos para interagir com os recursos do App Runner. O tráfego entre sua VPC e o App Runner não sai da rede Amazon. Para obter mais informações sobre VPC endpoints, consulte [VPC](#) endpoints no Guia.AWS PrivateLink

Note

Por padrão, o aplicativo web em seu serviço App Runner é executado em uma VPC que o App Runner fornece e configura. Essa VPC é pública. Isso significa que ele está conectado à internet. Opcionalmente, você pode associar seu aplicativo a uma VPC personalizada. Para obter mais informações, consulte [the section called “Tráfego de saída”](#).

Você pode configurar seus serviços para acessar a Internet AWS APIs, inclusive quando o serviço está conectado a uma VPC. Para obter instruções sobre como habilitar o acesso público à Internet para o tráfego de saída da VPC, consulte. [the section called “Considerações ao selecionar uma sub-rede”](#)

O App Runner não oferece suporte à criação de um VPC endpoint para seu aplicativo.

Configurando um VPC endpoint para o App Runner

Para criar a interface VPC endpoint para o serviço App Runner em sua VPC, siga o procedimento [Criar um endpoint de interface](#) no Guia.AWS PrivateLink Em Service Name (Nome do serviço), escolha com `.amazonaws.region.apprunner`.

Considerações sobre privacidade da rede VPC

Important

Usar um VPC endpoint para o App Runner não garante que todo o tráfego da sua VPC permaneça fora da Internet. A VPC pode ser pública. Além disso, algumas partes da sua solução podem não usar VPC endpoints para fazer AWS chamadas de API. Por exemplo, Serviços da AWS pode ligar para outros serviços usando seus endpoints públicos. Se a privacidade do tráfego for necessária para a solução em sua VPC, leia esta seção.

Para garantir a privacidade do tráfego de rede em sua VPC, considere o seguinte:

- Ativar nome DNS — partes do seu aplicativo ainda podem enviar solicitações ao App Runner pela Internet usando o endpoint `apprunner.region.amazonaws.com` público. Se sua VPC estiver configurada com acesso à Internet, essas solicitações serão bem-sucedidas sem nenhuma indicação para você. Você pode evitar isso garantindo que a opção Ativar nome DNS esteja ativada ao criar o endpoint. Por padrão, está definido como verdadeiro. Isso adiciona uma entrada DNS em sua VPC que mapeia o endpoint público de serviço para o VPC endpoint de interface.
- Configure endpoints VPC para serviços adicionais — sua solução pode enviar solicitações para outras pessoas. Serviços da AWS Por exemplo, AWS CodePipeline pode enviar solicitações para AWS CodeBuild. Configure endpoints VPC para esses serviços e habilite nomes DNS nesses endpoints.
- Configurar uma VPC privada — Se possível (se sua solução não precisar de acesso à Internet), configure sua VPC como privada, o que significa que ela não tem conexão com a Internet. Isso garante que um VPC endpoint ausente cause um erro visível, para que você possa adicionar o endpoint ausente.

Usar políticas de endpoint para controlar o acesso com VPC endpoints

As políticas de VPC endpoint são compatíveis com o App Runner. Por padrão, o acesso total ao App Runner é permitido por meio do endpoint da interface. As políticas de endpoint da VPC podem ser usadas para controlar quais diretores da AWS podem acessar o endpoint do App Runner. Como alternativa, você pode associar um grupo de segurança às interfaces de rede do endpoint para controlar o tráfego para o App Runner por meio do endpoint da interface.

Integração com o endpoint de interface

Suporte ao App Runner AWS PrivateLink, que fornece conectividade privada ao App Runner e elimina a exposição do tráfego à Internet. Para permitir que seu aplicativo envie solicitações ao App Runner usando AWS PrivateLink, configure um tipo de VPC endpoint conhecido como endpoint de interface. Para mais informações, consulte [Endpoints da VPC de interface\(AWS PrivateLink\)](#) no Guia AWS PrivateLink .

Análise de configuração e vulnerabilidade no App Runner

AWS e nossos clientes compartilham a responsabilidade de alcançar um alto nível de segurança e conformidade de componentes de software. Para obter mais informações, consulte o [modelo de responsabilidade AWS compartilhada](#).

Imagens de contêineres de patches

A correção da imagem do contêiner faz parte da responsabilidade do cliente no modelo de segurança compartilhada. O proprietário da imagem é responsável por atualizar e corrigir regularmente a imagem do contêiner. Recomendamos estabelecer um cronograma de rotina para verificar e aplicar atualizações às imagens do contêiner. Para obter mais informações sobre como escanear suas imagens em busca de vulnerabilidades, consulte a documentação do [AWS App Runner](#)

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Melhores práticas de segurança para o App Runner

AWS App Runner fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As práticas recomendadas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes no seu ambiente, trate-as como considerações úteis, não como requisitos.

Para outros tópicos de segurança do App Runner, consulte [Segurança](#).

Práticas recomendadas de segurança preventiva

Os controles de segurança preventiva tentam evitar incidentes antes que ocorram.

Implemente o acesso de privilégio mínimo

O App Runner fornece políticas gerenciadas AWS Identity and Access Management (IAM) para [usuários do IAM](#) e a [função de acesso](#). Essas políticas gerenciadas especificam todas as permissões que podem ser necessárias para a operação correta do serviço App Runner.

O aplicativo pode não exigir todas as permissões em nossas políticas gerenciadas. Você pode personalizá-los e conceder somente as permissões necessárias para que seus usuários e seu serviço App Runner realizem suas tarefas. Isso é especialmente relevante para políticas de usuário, em que diferentes funções de usuário podem ter diferentes necessidades de permissão. A implementação do acesso de privilégio mínimo é fundamental para reduzir o risco de segurança e o impacto que pode resultar de erros ou usuários mal-intencionados.

Verifique suas imagens em busca de vulnerabilidades

Você pode usar os Amazon ECRs APIs para ajudar a identificar vulnerabilidades de software nas imagens do seu contêiner. Para obter mais informações, consulte a [documentação do Amazon ECR](#).

Práticas recomendadas de segurança de detecção

Os controles de segurança de detecção identificam violações de segurança depois de ocorrerem. Eles podem ajudar a detectar uma possível ameaça ou incidente de segurança.

Implementar o monitoramento

O monitoramento é uma parte importante para manter a confiabilidade, a segurança, a disponibilidade e o desempenho de suas soluções App Runner. AWS fornece várias ferramentas e serviços para ajudá-lo a monitorar seus AWS serviços.

Veja a seguir alguns exemplos de itens a serem monitorados:

- CloudWatch Métricas da Amazon para App Runner — Defina alarmes para as principais métricas do App Runner e para as métricas personalizadas do seu aplicativo. Para obter detalhes, consulte [Métricas \(CloudWatch\)](#).
- AWS CloudTrail entradas — acompanhe ações que possam afetar a disponibilidade, como `PauseService` ou `DeleteConnection`. Para obter mais detalhes, consulte [Ações de API \(CloudTrail\)](#).

AWS Glossário

Para obter a AWS terminologia mais recente, consulte o [AWS glossário](#) na Glossário da AWS Referência.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.