Manual do usuário

AWS App Mesh

AWS App Mesh: Manual do usuário

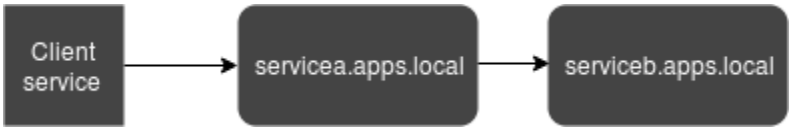
Table of Contents

O que é o AWS App Mesh?

O AWS App Mesh é uma malha de serviços que facilita o monitoramento e o controle de serviços. Uma malha de serviços é uma camada de infraestrutura dedicada a lidar com a comunicação entre serviços, geralmente por meio de uma variedade de proxies de rede leves implantados junto com o código do aplicativo. O App Mesh padroniza o modo como os serviços se comunicam, oferecendo a você visibilidade completa e ajudando a garantir alta disponibilidade para as aplicações. O App Mesh oferece controles do tráfego de rede e visibilidade consistentes para cada serviço em uma aplicação.

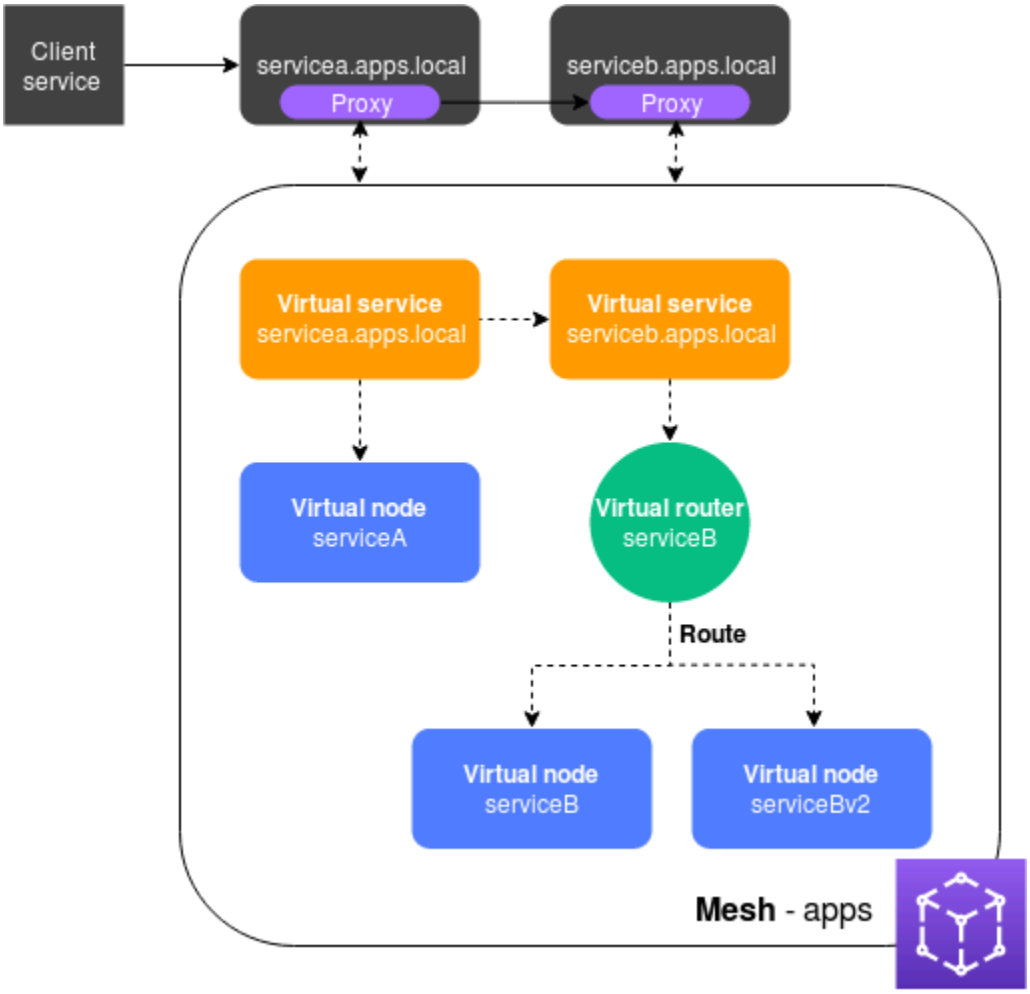
Adicionar o App Mesh a um aplicativo de exemplo

Considere o seguinte exemplo simples de aplicativo que não usa o App Mesh. Os dois serviços podem ser executados no AWS Fargate, no Amazon Elastic Container Service (Amazon ECS), no Amazon Elastic Kubernetes Service (Amazon EKS), no Kubernetes em instâncias do Amazon Elastic Compute Cloud (Amazon EC2) ou em instâncias do Amazon EC2 com o Docker.



Nesta ilustração, `serviceA` e `serviceB`, ambos são detectáveis por meio do namespace `apps.local`. Digamos, por exemplo, que você decida implantar uma nova versão do `serviceb.apps.local` nomeado `servicebv2.apps.local`. Em seguida, você deseja direcionar uma porcentagem do tráfego de `servicea.apps.local` para `serviceb.apps.local` e uma porcentagem para `servicebv2.apps.local`. Quando tiver certeza de que `servicebv2` está funcionando bem, você deseja enviar 100% do tráfego para ele.

O App Mesh pode ajudá-lo a fazer isso sem alterar nenhum código de aplicativo ou nomes de serviço registrados. Se você usar o App Mesh com esse aplicativo de exemplo, sua malha poderá se parecer com a ilustração a seguir.



Nessa configuração, os serviços não se comunicam mais diretamente entre si. Em vez disso, eles se comunicam entre si por meio de um proxy. O proxy implantado com o serviço `servicea.apps.local` lê a configuração do App Mesh e envia tráfego para `serviceb.apps.local` ou `servicebv2.apps.local` com base na configuração.

Componentes do App Mesh

O App Mesh tem os seguintes componentes, ilustrados no exemplo anterior:

- Malha de serviços: uma malha de serviços é um limite lógico para o tráfego de rede entre os serviços que residem nela. No exemplo, a malha é nomeada `apps` e contém todos os outros recursos da malha. Para obter mais informações, consulte Malhas de serviço.
- Serviços virtuais: um serviço virtual é uma abstração de um serviço real que é fornecido por um nó virtual direta ou indiretamente por meio de um roteador virtual. Na ilustração, dois serviços virtuais representam os dois serviços reais. Os nomes dos serviços virtuais são os nomes detectáveis dos

serviços reais. Quando um serviço virtual e um serviço real têm o mesmo nome, vários serviços podem se comunicar entre si usando os mesmos nomes que usavam antes da implementação do App Mesh. Para obter mais informações, consulte Serviços virtuais.

- Nós virtuais: um nó virtual atua como um ponteiro lógico para um serviço detectável, como um serviço do Amazon ECS ou Kubernetes. Para cada serviço virtual, você terá pelo menos um nó virtual. Na ilustração, o serviço virtual `servicea.apps.local` obtém informações de configuração para o nó virtual nomeado `serviceA`. O nó virtual `serviceA` é configurado com o nome `servicea.apps.local` para descoberta de serviços. O serviço virtual `serviceb.apps.local` é configurado para rotear o tráfego para os nós virtuais `serviceB` e `serviceBv2` por meio de um roteador virtual nomeado `serviceB`. Para obter mais informações, consulte Nós virtuais.
- Roteadores virtuais e rotas: os roteadores virtuais manipulam o tráfego de um ou mais serviços virtuais dentro da malha. Uma rota está associada a um roteador virtual. A rota é usada para atender às solicitações do roteador virtual e distribuir o tráfego para os nós virtuais associados. Na ilustração anterior, o roteador virtual `serviceB` tem uma rota que direciona uma porcentagem do tráfego para o nó virtual `serviceB` e uma porcentagem do tráfego para o nó virtual `serviceBv2`. Você pode definir a porcentagem de tráfego roteado para um determinado nó virtual e alterá-la ao longo do tempo. Você pode rotear o tráfego com base em critérios como cabeçalhos HTTP, caminhos de URL ou nomes de serviços e métodos gRPC. Você pode configurar políticas de novas tentativas para tentar novamente uma conexão se houver um erro na resposta. Por exemplo, na ilustração, a política de novas tentativas da rota pode especificar que uma conexão com `serviceb.apps.local` seja tentada novamente cinco vezes, com dez segundos entre as tentativas de repetição, se `serviceb.apps.local` retornar tipos específicos de erros. Para obter mais informações, consulte Roteadores virtuais e Rotas.
- Proxy: você configura seus serviços para usar o proxy depois de criar sua malha e seus recursos. O proxy lê a configuração do App Mesh e direciona o tráfego de forma adequada. Na ilustração, toda a comunicação de `servicea.apps.local` até `serviceb.apps.local` passa pelo proxy implantado em cada serviço. Os serviços se comunicam entre si usando os mesmos nomes de descoberta de serviços usados antes da introdução do App Mesh. Como o proxy lê a configuração do App Mesh, você pode controlar como os dois serviços se comunicam entre si. Quando quiser alterar a configuração do App Mesh, você não precisa alterar nem reimplantar os próprios serviços ou os proxies. Para obter mais informações, consulte Imagem do Envoy.

Como começar a usar

Para usar o App Mesh, você deve ter um serviço existente em execução no AWS Fargate, no Amazon ECS, no Amazon EKS, no Kubernetes no Amazon EC2 ou no Amazon EC2 com Docker.

Para começar a usar tags no App Mesh, consulte os guias a seguir:

- Conceitos básicos do App Mesh e do Amazon ECS
- Conceitos básicos do App Mesh e do Kubernetes
- Conceitos básicos do App Mesh e do Amazon EC2

Acessar o App Mesh

Você pode trabalhar com App Mesh das seguintes formas:

AWS Management Console

O console é uma interface baseada em navegador que você pode usar para gerenciar os recursos do App Mesh. Você pode abrir o console do App Mesh em https://console.aws.amazon.com/appmesh/.

AWS CLI

Fornece comandos para um conjunto amplo de produtos da AWS e é compatível com Windows, Mac e Linux. Para começar a usar, consulte oAWS Command Line Interface User Guide (Guia do usuário da AWS Command Line Interface). Para obter mais informações sobre comandos para App Mesh, consulte appmesh na Referência de comandos da AWS CLI.

AWS Tools for Windows PowerShell

Fornece comandos para um conjunto amplo de produtos da AWS para os usuários que usam script no ambiente do PowerShell. Para começar a usar, consulte o Guia do usuário do AWS Tools for Windows PowerShell. Para obter mais informações sobre cmdlets para o App Mesh, consulte App Mesh na Referência de Cmdlets para ferramentas da AWS no PowerShell.

AWS CloudFormation

Permite que você crie um modelo que descreva os recursos da AWS que deseja. Usando o modelo, o AWS CloudFormation provisiona e configura os recursos para você. Para começar a usar, consulte oAWS CloudFormation User Guide (Guia do usuário da AWS CloudFormation).

Para obter mais informações sobre os tipos de recursos do App Mesh, consulte Referência do tipo de recurso do App Mesh na Referência do modelo do AWS CloudFormation.

AWS SDKs

Também fornecemos SDKs que permitem que você acesse o App Mesh com diversas linguagens de programação. Os SDKs cuidam automaticamente de tarefas como:

- Assinar criptograficamente suas solicitações de serviço
- Recuperar solicitações
- Lidar com respostas de erro

Para mais informações sobre os SDKs disponíveis, consulte Ferramentas para a Amazon Web Services.

Para obter mais informações sobre como usar as APIs do App Mesh, consulte a Referência da API do AWS App Mesh.

Conceitos básicos do App Mesh

Você pode usar o App Mesh com aplicativos que você implanta no Amazon ECS, Kubernetes (que você implanta em suas próprias instâncias do Amazon EC2 ou executados no Amazon EKS) e Amazon EC2. Para começar a usar o App Mesh, selecione um dos serviços nos quais você tem aplicativos implantados e que deseja usar com o App Mesh. Você sempre pode permitir que aplicativos em outros serviços também funcionem com o App Mesh depois de concluir um dos guias de conceitos básicos.

Tópicos

Introdução ao AWS App Mesh Amazon ECS

Este tópico ajuda você a usar AWS App Mesh com um serviço real que está sendo executado no Amazon ECS. Esse tutorial aborda os atributos básicos de vários tipos de recursos do App Mesh.

Cenário

Para ilustrar como usar o App Mesh, suponha que você tenha um aplicativo com as seguintes características:

- Consiste em dois serviços chamados `serviceA` e `serviceB`.
- Ambos os serviços estão registrados em um namespace chamado `apps.local`.
- O `ServiceA` se comunica com o `serviceB` por HTTP/2, porta 80.
- Você já implantou a versão 2 do `serviceB` e a registrou com o nome de `serviceBv2` no namespace `apps.local`.

Você tem os seguintes requisitos:

- Você deseja enviar 75% do tráfego do `serviceA` para o `serviceB` e 25% do tráfego do `serviceBv2` para garantir que o `serviceBv2` esteja livre de bugs antes de enviar 100% do tráfego do `serviceA` para ele.
- Você quer poder ajustar facilmente a ponderação do tráfego para que 100% do tráfego vá para o `serviceBv2` quando for comprovado que ele é confiável. Depois que todo o tráfego estiver sendo enviado para o `serviceBv2`, você deseja descontinuar o `serviceB`.
- Você não quer ter que alterar nenhum código de aplicativo ou registro de descoberta de serviços existente para que seus serviços reais atendam aos requisitos anteriores.

Para atender às suas necessidades, você decide criar uma malha de serviços do App Mesh com serviços virtuais, nós virtuais, um roteador virtual e uma rota. Depois de implementar a malha, você atualiza os serviços que usam o proxy do Envoy. Assim que forem atualizados, os serviços se comunicarão entre si por meio do proxy Envoy em vez de diretamente entre si.

Pré-requisitos

- Uma compreensão existente dos conceitos do App Mesh. Para ter mais informações, consulte O que é o AWS App Mesh?.
- Uma compreensão existente dos conceitos do Amazon ECSs. Para obter mais informações, consulte O que é o Amazon ECS no Guia do desenvolvedor do Amazon Elastic Container Service.
- O App Mesh oferece suporte a serviços Linux registrados com DNS ou ambos. AWS Cloud Map Para usar este guia de conceitos básicos, recomendamos que você tenha três serviços existentes registrados no DNS. Para os procedimentos nesse tópico, espera-se que os serviços existentes sejam nomeados `serviceA`, `serviceB` e `serviceBv2`; e que todos os serviços sejam detectáveis por meio de um namespace chamado `apps.local`.

  É possível criar uma malha de serviço e seus recursos mesmo que os serviços não existam, mas não será possível usar a malha enquanto não tiver implantado serviços reais. Para mais informações sobre a descoberta de serviços no Amazon ECS, consulte Descoberta de serviços. Para criar um serviço do Amazon ECS com descoberta de serviços, consulte Tutorial: Como criar um serviço usando a descoberta de serviços. Se você ainda não tem serviços em execução, você pode Criar um serviço Amazon ECS com a descoberta de serviços.

Etapa 1: Criar uma malha e um serviço virtual

Uma malha de serviços é um limite lógico para o tráfego de rede entre os serviços que residem nela. Para ter mais informações, consulte Malhas de serviço. Um serviço virtual é uma abstração de um serviço real. Para ter mais informações, consulte Serviços virtuais.

Crie os recursos da a seguir:

- Uma malha chamada `apps`, uma vez que todos os serviços no cenário estão registrados no namespace `apps.local`.
- Um serviço virtual chamado `serviceb.apps.local`, uma vez que o serviço virtual representa um serviço que é detectável com esse nome e você não quer alterar o código para fazer referência a outro nome. Um serviço virtual chamado `servicea.apps.local` será adicionado em uma etapa posterior.

Você pode usar a AWS CLI versão 1.18.116 AWS Management Console ou superior ou 2.0.38 ou superior para concluir as etapas a seguir. Se estiver usando o AWS CLI, use o `aws --version` comando para verificar sua AWS CLI versão instalada. Se você não tiver a versão 1.18.116 ou superior ou a versão 2.0.38 ou superior instalada, será necessário instalar ou atualizar a AWS CLI. Selecione a guia da ferramenta que deseja usar.

AWS Management Console

1. Abra o assistente de primeira execução do console do App Mesh em https://console.aws.amazon.com/appmesh/get-started.
2. Em Mesh name (Nome da malha), insira **`apps`**.
3. Em Virtual service name (Nome do serviço virtual), insira **`serviceb.apps.local`**.
4. Para continuar, escolha Avançar.

AWS CLI

1. Crie uma malha com o comando `create-mesh`.

   ```
   aws appmesh create-mesh --mesh-name apps
   ```

2. Crie um serviço virtual com o comando `create-virtual-service`.

```
aws appmesh create-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local --spec {}
```

Etapa 2: Criar um nó virtual

Um nó virtual funciona como um apontador lógico para um serviço real. Para ter mais informações, consulte Nós virtuais.

Crie um nó virtual chamado `serviceB`, uma vez que um dos nós virtuais representa o serviço real chamado `serviceB`. O serviço real que o nó virtual representa é detectável por meio do DNS com um nome de host de `serviceb.apps.local`. Como alternativa, você pode descobrir serviços reais usando. AWS Cloud Map O nó virtual escutará o tráfego usando o protocolo HTTP/2 na porta 80. Outros protocolos, assim como verificações de integridade, também são compatíveis. Você criará nós virtuais para `serviceA` e `serviceBv2` em uma etapa posterior.

AWS Management Console

1. Em Virtual node name (Nome do nó virtual), insira **`serviceB`**.
2. Em Service discovery method (Método de descoberta de serviços), escolha DNS e insira **`serviceb.apps.local`** para DNS hostname (Nome de host do DNS).
3. Em Listener configuration (Configuração do Listener), escolha http2 para Protocol (Protocolo) e digite **`80`** para Port (Porta).
4. Para continuar, escolha Avançar.

AWS CLI

1. Crie um arquivo denominado `create-virtual-node-serviceb.json` com o conteúdo a seguir:

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
```

```
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceB.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceB"
}
```

2. Crie o nó virtual com o create-virtual-nodecomando usando o arquivo JSON como entrada.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-serviceb.json
```

Etapa 3: Criar um roteador virtual e uma rota

Os roteadores virtuais cuidam do tráfego de um ou mais serviços virtuais dentro da malha. Para obter mais informações, consulte Roteadores virtuais e Rotas.

Crie os recursos da a seguir:

- Um roteador virtual denominado `serviceB`, uma vez que o serviço virtual do `serviceB.apps.local` não inicia a comunicação de saída com nenhum outro serviço. Lembre-se de que o serviço virtual criado anteriormente é uma abstração do serviço `serviceb.apps.local` real. O serviço virtual envia tráfego para o roteador virtual. O roteador virtual recebe o tráfego usando o protocolo HTTP/2 na porta 80. Outros protocolos também são compatíveis.
- Uma rota chamada `serviceB`. Ela roteia 100% de seu tráfego para o nó virtual do `serviceB`. O peso será alterado em uma etapa posterior, depois de adicionar o nó virtual do `serviceBv2`. Embora não seja abordado neste guia, é possível adicionar critérios de filtro adicionais para a rota e adicionar uma política de novas tentativas para fazer com que o proxy Envoy faça várias tentativas de enviar tráfego para um nó virtual quando ele tiver um problema de comunicação.

AWS Management Console

1. Em Virtual router name (Nome do roteador virtual), insira **`serviceB`**.

2. Em Listener configuration (Configuração do Listener), escolha http2 para Protocol (Protocolo) e especifique **`80`** para Port (Porta).
3. Em Route name (Nome da rota), insira **`serviceB`**.
4. Em Route type (Tipo de rota), escolha http2.
5. Para o nome do nó virtual em Configuração de destino, selecione `serviceB` e digite **`100`** para Peso.
6. Em Configuração de correspondência, escolha um Método.
7. Para continuar, escolha Avançar.

AWS CLI

1. Crie um roteador virtual.

   a. Crie um arquivo denominado `create-virtual-router.json` com o conteúdo a seguir:

   ```
   {
       "meshName": "apps",
       "spec": {
           "listeners": [
               {
                   "portMapping": {
                       "port": 80,
                       "protocol": "http2"
                   }
               }
           ]
       },
       "virtualRouterName": "serviceB"
   }
   ```

   b. Crie o roteador virtual com o create-virtual-routercomando usando o arquivo JSON como entrada.

   ```
   aws appmesh create-virtual-router --cli-input-json file://create-virtual-router.json
   ```

2. Crie uma rota.

a. Crie um arquivo denominado `create-route.json` com o conteúdo a seguir:

```
{
    "meshName" : "apps",
    "routeName" : "serviceB",
    "spec" : {
        "httpRoute" : {
            "action" : {
                "weightedTargets" : [
                    {
                        "virtualNode" : "serviceB",
                        "weight" : 100
                    }
                ]
            },
            "match" : {
                "prefix" : "/"
            }
        }
    },
    "virtualRouterName" : "serviceB"
}
```

b. Crie a rota com o comando create-route usando o arquivo JSON como entrada.

```
aws appmesh create-route --cli-input-json file://create-route.json
```

Etapa 4: revisar e criar

Revise as configurações em relação às instruções anteriores.

AWS Management Console

Se precisar fazer alterações em qualquer seção, selecione Edit (Editar). Quando estiver satisfeito com as configurações, escolha Create mesh (Criar malha).

A tela Status mostra todos os recursos de malha que foram criados. Você pode ver no console os recursos criados selecionando View mesh (Exibir malha).

AWS CLI

Revise as configurações da malha criada com o comando describe-mesh.

```
aws appmesh describe-mesh --mesh-name apps
```

Revise as configurações do serviço virtual que você criou com o describe-virtual-servicecomando.

```
aws appmesh describe-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local
```

Revise as configurações do nó virtual que você criou com o describe-virtual-nodecomando.

```
aws appmesh describe-virtual-node --mesh-name apps --virtual-node-name serviceB
```

Revise as configurações do roteador virtual que você criou com o describe-virtual-routercomando.

```
aws appmesh describe-virtual-router --mesh-name apps --virtual-router-name serviceB
```

Revise as configurações da rota criada com o comando describe-route.

```
aws appmesh describe-route --mesh-name apps \
    --virtual-router-name serviceB  --route-name serviceB
```

Etapa 5: Criar recursos adicionais

Para concluir o cenário, é necessário:

- Criar um nó virtual chamado `serviceBv2` e outro chamado `serviceA`. Ambos os nós virtuais escutam solicitações por meio da porta 80 do HTTP/2. Para o nó virtual `serviceA`, configure um back-end do `serviceb.apps.local`. Todo o tráfego de saída do nó virtual `serviceA` é enviado para o serviço virtual chamado `serviceb.apps.local`. Embora não seja abordado neste guia, também é possível especificar um caminho de arquivo para gravar logs de acesso para um nó virtual.
- Crie um serviço virtual adicional chamado `servicea.apps.local`, que enviará todo o tráfego diretamente para o nó virtual do `serviceA`.
- Atualizar a rota do `serviceB` criada em uma etapa anterior para enviar 75% de seu tráfego para o nó virtual do `serviceB` e 25% de seu tráfego para o nó virtual do `serviceBv2`. Com o passar

do tempo, você poderá continuar a modificar os pesos até que o `serviceBv2` receba 100% do tráfego. Depois que todo o tráfego for enviado para o `serviceBv2`, você poderá descontinuar o nó virtual do `serviceB` e o serviço real. Conforme você altera os pesos, o código não exigirá nenhuma modificação, porque os nomes de serviço `serviceb.apps.local` virtual e real não são alterados. Lembre-se de que o serviço virtual `serviceb.apps.local` envia tráfego para o roteador virtual, que roteia o tráfego para os nós virtuais. Os nomes de descoberta de serviço para os nós virtuais podem ser alterados a qualquer momento.

AWS Management Console

1. No painel de navegação à esquerda, selecione Meshes (Malhas).
2. Selecione a malha `apps` criada em uma etapa anterior.
3. No painel de navegação esquerdo, selecione Virtual nodes (Nós virtuais).
4. Selecione Create nó virtual (Criar nó virtual).
5. Em Virtual node name (Nome do nó virtual), insira **`serviceBv2`**, em Service discovery method (Método de descoberta de serviço), escolha DNS e, em DNS hostname (Nome de host do DNS), insira **`servicebv2.apps.local`**.
6. Em Listener configuration (Configuração do Listener), selecione http2 para Protocol (Protocolo) e digite **`80`** para Port (Porta).
7. Selecione Create nó virtual (Criar nó virtual).
8. Selecione Create nó virtual (Criar nó virtual) novamente. Digite **`serviceA`** para o o Virtual node name (Nome do nó virtual). Em Service discovery method (Método de descoberta de serviços), escolha DNS e, para DNS hostname (Nome de host do DNS), insira **`servicea.apps.local`**.
9. Para Enter a virtual service name (Digite um nome de serviço virtual) em New backend (Novo back-end), digite **`serviceb.apps.local`**.
10. Em Listener configuration (Configuração do Listener), escolha http2 para Protocol (Protocolo), digite **`80`** para Port (Porta) e escolha Create virtual node (Criar nó virtual).
11. No painel de navegação esquerdo, selecione Virtual routers (Roteadores virtuais) e, depois, selecione o roteador virtual `serviceB` na lista.
12. Em Routes (Rotas), selecione a rota chamada `ServiceB` criada em uma etapa anterior e escolha Edit (Editar).
13. Em Targets (Destinos), Virtual node name (Nome do nó virtual), altere o valor de Weight (Peso) de `serviceB` para **`75`**.

14. Escolha Adicionar destino, depois escolha `serviceBv2` na lista suspensa e defina o valor de Peso como **25**.
15. Escolha Salvar.
16. No painel de navegação esquerdo, selecione Virtual services (Serviços virtuais) e escolha Create virtual service (Criar serviço virtual).
17. Insira **`servicea.apps.local`** para Virtual service name (Nome do serviço virtual), selecione Virtual node para Provider (Provedor), selecione `serviceA` para Virtual node (Nó virtual) e escolha Create virtual service (Criar serviço virtual).

AWS CLI

1. Crie o nó virtual `serviceBv2`.

   a. Crie um arquivo denominado `create-virtual-node-servicebv2.json` com o conteúdo a seguir:

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceBv2.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceBv2"
}
```

   b. Crie o nó virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicebv2.json
```

2. Crie o nó virtual `serviceA`.

   a. Crie um arquivo denominado `create-virtual-node-servicea.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "spec" : {
      "backends" : [
         {
            "virtualService" : {
               "virtualServiceName" : "serviceb.apps.local"
            }
         }
      ],
      "listeners" : [
         {
            "portMapping" : {
               "port" : 80,
               "protocol" : "http2"
            }
         }
      ],
      "serviceDiscovery" : {
         "dns" : {
            "hostname" : "servicea.apps.local"
         }
      }
   },
   "virtualNodeName" : "serviceA"
}
```

   b. Crie o nó virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicea.json
```

3. Atualize o serviço virtual `serviceb.apps.local` criado em uma etapa anterior para enviar seu tráfego para o roteador virtual `serviceB`. Quando o serviço virtual foi criado originalmente, ele não enviava tráfego para nenhum lugar, já que o roteador virtual `serviceB` ainda não tinha sido criado.

a. Crie um arquivo denominado `update-virtual-service.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualRouter" : {
            "virtualRouterName" : "serviceB"
         }
      }
   },
   "virtualServiceName" : "serviceb.apps.local"
}
```

b. Atualize o serviço virtual com o update-virtual-servicecomando.

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

4. Atualize a rota `serviceB` criada em uma etapa anterior.

a. Crie um arquivo denominado `update-route.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "routeName" : "serviceB",
   "spec" : {
      "http2Route" : {
         "action" : {
            "weightedTargets" : [
               {
                  "virtualNode" : "serviceB",
                  "weight" : 75
               },
               {
                  "virtualNode" : "serviceBv2",
                  "weight" : 25
               }
            ]
         },
         "match" : {
```

```
            "prefix" : "/"
         }
      }
   },
   "virtualRouterName" : "serviceB"
}
```

b. Atualize a rota com o comando update-route.

```
aws appmesh update-route --cli-input-json file://update-route.json
```

5. Crie o serviço virtual `serviceA`.

a. Crie um arquivo denominado `create-virtual-servicea.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualNode" : {
            "virtualNodeName" : "serviceA"
         }
      }
   },
   "virtualServiceName" : "servicea.apps.local"
}
```

b. Crie o serviço virtual.

```
aws appmesh create-virtual-service --cli-input-json file://create-virtual-
servicea.json
```

Resumo da malha

Antes de criar a malha de serviço, você tinha três serviços reais chamados `servicea.apps.local`, `serviceb.apps.local` e `servicebv2.apps.local`. Além dos serviços reais, agora você tem uma malha de serviços que contém os seguintes recursos que representam os serviços reais:

- Dois serviços virtuais. O proxy envia todo o tráfego do serviço virtual `servicea.apps.local` para o serviço virtual `serviceb.apps.local` por meio de um roteador virtual.
- Três nós virtuais chamados `serviceA`, `serviceB` e `serviceBv2`. O proxy Envoy usa as informações de descoberta de serviço configuradas para os nós virtuais para pesquisar os endereços IP dos serviços reais.
- Um roteador virtual com uma rota que instrui o proxy Envoy a rotear 75% do tráfego de entrada para o nó virtual `serviceB` e 25% do tráfego para o nó virtual `serviceBv2`.

Etapa 6: Atualizar os serviços

Depois de criar a malha, é necessário concluir as seguintes tarefas:

- Autorize o proxy Envoy implantado com cada tarefa do Amazon ECS para ler a configuração de um ou mais nós virtuais. Para mais informações sobre como autorizar o proxy, consulte Autorização de proxy.
- Atualize cada uma das definições de tarefa do Amazon ECS existentes para usar o proxy Envoy.

Credenciais

O contêiner Envoy exige AWS Identity and Access Management credenciais para assinar solicitações enviadas ao serviço App Mesh. Para tarefas do Amazon ECS implantadas com o tipo de execução do Amazon EC2, as credenciais podem vir da função da instância ou de um perfil do IAM da tarefa. As tarefas do Amazon ECS implantadas com o Fargate em contêineres Linux não têm acesso ao servidor de metadados do Amazon EC2 que fornece credenciais de perfil do IAM da instância. Para fornecer as credenciais, você deve associar uma função de tarefa do IAM a qualquer tarefa implantada com o Fargate no tipo de contêiner Linux.

Se uma tarefa for implantada com o tipo de inicialização do Amazon EC2 e o acesso for bloqueado para o servidor de metadados do Amazon EC2, conforme descrito na anotação Importante em Perfil do IAM para tarefas, um perfil do IAM de tarefa também deverá ser associada à tarefa. A função que você atribui à instância ou à tarefa deve ter uma política do IAM anexada a ela, conforme descrito em Autorização de proxy.

Para atualizar sua definição de tarefa usando o AWS CLI

Você usa o AWS CLI comando `register-task-definition`Amazon ECS. O exemplo de definição de tarefa abaixo mostra como configurar o App Mesh para seu serviço.

> **Note**
>
> A configuração do App Mesh para Amazon ECS por meio do console não está disponível.

Json da definição de tarefas

Configuração do proxy

Para configurar seu serviço do Amazon ECS para usar o App Mesh, a definição de tarefa do serviço deverá ter a seção de configuração de proxy a seguir. Defina a configuração de proxy `type` como `APPMESH` e `containerName` como `envoy`. Defina os seguintes valores de propriedade da maneira adequada.

`IgnoredUID`

O proxy Envoy não roteia o tráfego de processos que usam esse ID de usuário. É possível escolher qualquer ID de usuário que você quiser para esse valor de propriedade, mas esse ID deve ser o mesmo que o ID de `user` do contêiner do Envoy na definição de tarefa. Essa correspondência permite que o Envoy ignore seu próprio tráfego sem usar o proxy. Nossos exemplos usam *`1337`* para fins históricos.

`ProxyIngressPort`

Essa é a porta de entrada para o contêiner do proxy Envoy. Defina este valor como `15000`.

`ProxyEgressPort`

Essa é a porta de saída para o contêiner do proxy Envoy. Defina este valor como `15001`.

`AppPorts`

Especifique as portas de entrada em que seus contêineres de aplicativos recebem. Neste exemplo, o contêiner do aplicativo escuta na porta *`9080`*. A porta especificada deve corresponder à porta configurada no listener do nó virtual.

`EgressIgnoredIPs`

O Envoy não faz a intermediação do tráfego para esses endereços IP. Defina esse valor como `169.254.170.2,169.254.169.254`, que ignora o servidor de metadados do EC2 e o endpoint de metadados da tarefa do ECS. O endpoint de metadados fornece perfis do IAM para credenciais de tarefas. É possível adicionar outros endereços.

`EgressIgnoredPorts`

É possível adicionar uma lista de portas separada por vírgulas. O Envoy não faz a intermediação do tráfego para essas portas. Mesmo que você não liste nenhuma porta, a porta 22 será ignorada.

> **Note**
>
> O número máximo de portas de saída que podem ser ignoradas é 15.

```
"proxyConfiguration": {
 "type": "APPMESH",
 "containerName": "envoy",
 "properties": [{
   "name": "IgnoredUID",
   "value": "1337"
  },
  {
   "name": "ProxyIngressPort",
   "value": "15000"
  },
  {
   "name": "ProxyEgressPort",
   "value": "15001"
  },
  {
   "name": "AppPorts",
   "value": "9080"
  },
  {
   "name": "EgressIgnoredIPs",
   "value": "169.254.170.2,169.254.169.254"
  },
  {
   "name": "EgressIgnoredPorts",
   "value": "22"
  }
 ]
}
```

Dependência de contêiner de aplicativos do Envoy

Os contêineres de aplicativos em suas definições de tarefas devem aguardar o bootstrap e o início do proxy Envoy para que possam ser iniciados. Para garantir que isso aconteça, você define uma seção `dependsOn` em cada definição de contêiner de aplicativos para aguardar que o contêiner do Envoy seja relatado como `HEALTHY`. O código a seguir mostra um exemplo de definição de contêiner de aplicativos com essa dependência. Todas as propriedades no exemplo a seguir são necessárias. Alguns dos valores de propriedade também são necessários, mas alguns são *substituíveis*.

```
{
 "name": "appName",
 "image": "appImage",
 "portMappings": [{
  "containerPort": 9080,
  "hostPort": 9080,
  "protocol": "tcp"
 }],
 "essential": true,
 "dependsOn": [{
  "containerName": "envoy",
  "condition": "HEALTHY"
 }]
}
```

Definição de contêiner do Envoy

Suas definições de tarefas do Amazon ECS devem conter uma imagem de contêiner do App Mesh Envoy.

Todas as regiões suportadas, exceto `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`. Você pode substituir o *Código de região* por qualquer Região que não seja `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`.

Padrão

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

me-south-1

Padrão

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

ap-east-1

Padrão

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

ap-southeast-3

Padrão

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-
fips
```

eu-south-1

Padrão

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

il-central-1

Padrão

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-
fips
```

af-south-1

Padrão

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

Public repository

Padrão

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

Important

Somente a versão v1.9.0.0-prod ou posterior é compatível para uso com o App Mesh.

Você deve usar a imagem do contêiner do App Mesh Envoy até que a equipe do projeto Envoy mescle as alterações compatíveis com o App Mesh. Para obter detalhes adicionais, consulte a edição do GitHub roteiro.

Todas as propriedades no exemplo a seguir são necessárias. Alguns dos valores de propriedade também são necessários, mas alguns são *substituíveis*.

> **Note**
>
> - A definição de contêiner do Envoy deve ser marcada como `essential`.
> - Recomendamos a alocação de `512` unidades de CPU e pelo menos `64` MiB de memória para o contêiner do Envoy. No Fargate, o mínimo que você poderá definir é `1024` MiB de memória.
> - O nome do nó virtual para o serviço do Amazon ECS deve ser definido como o valor da propriedade `APPMESH_RESOURCE_ARN`. Essa propriedade requer uma versão `1.15.0` ou posterior da imagem do Envoy. Para ter mais informações, consulte Envoy.
> - O valor da configuração de `user` deve corresponder ao valor `IgnoredUID` da configuração de proxy da definição de tarefa. Neste exemplo, usamos *`1337`*.
> - A verificação de integridade mostrada aqui aguarda o contêiner do Envoy ser inicializado corretamente antes de relatar ao Amazon ECS que o contêiner do Envoy está íntegro e pronto para que os contêineres de aplicativos sejam iniciados.
> - Por padrão, o App Mesh usa o nome do recurso especificado em `APPMESH_RESOURCE_ARN` quando o Envoy está se referindo a si mesmo em métricas e rastreamentos. É possível substituir esse comportamento definindo a variável de ambiente `APPMESH_RESOURCE_CLUSTER` com seu próprio nome. Essa propriedade requer uma versão `1.15.0` ou posterior da imagem do Envoy. Para ter mais informações, consulte Envoy.

O código a seguir mostra um exemplo de definição de contêiner do Envoy.

```
{
 "name": "envoy",
 "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-
prod",
 "essential": true,
 "environment": [{
```

```
  "name": "APPMESH_RESOURCE_ARN",
  "value": "arn:aws:appmesh:us-west-2:111122223333:mesh/apps/virtualNode/serviceB"
 }],
 "healthCheck": {
  "command": [
   "CMD-SHELL",
   "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
  ],
  "startPeriod": 10,
  "interval": 5,
  "timeout": 2,
  "retries": 3
 },
 "user": "1337"
}
```

Exemplos de definições de tarefa

O exemplo de definições de tarefa do Amazon ECS a seguir mostra como mesclar os exemplos acima em uma definição de tarefa para `taskB`. São fornecidos exemplos de criação de tarefas para ambos os tipos de inicialização do Amazon ECS, com ou sem o uso do AWS X-Ray. Altere os valores *substituíveis* conforme apropriado a fim de criar definições de tarefa para as tarefas chamadas `taskBv2` e `taskA` do cenário. Substitua o nome da malha e o nome do nó virtual pelo valor `APPMESH_RESOURCE_ARN` e uma lista de portas em que seu aplicativo escuta pelo valor `AppPorts` da configuração de proxy. Por padrão, o App Mesh usa o nome do recurso especificado em `APPMESH_RESOURCE_ARN` quando o Envoy está se referindo a si mesmo em métricas e rastreamentos. É possível substituir esse comportamento definindo a variável de ambiente `APPMESH_RESOURCE_CLUSTER` com seu próprio nome. Todas as propriedades nos exemplos a seguir são necessárias. Alguns dos valores de propriedade também são necessários, mas alguns são *substituíveis*.

Se você estiver executando uma tarefa do Amazon ECS conforme descrito na seção Credenciais, será necessário adicionar um perfil do IAM de tarefa existente aos exemplos.

> **Important**
>
> O Fargate deve usar um valor de porta maior que 1024.

Example JSON para definição de tarefas do Amazon ECS: Fargate em contêineres Linux

```
{

   "family" : "taskB",
   "memory" : "1024",
   "cpu" : "0.5 vCPU",
   "proxyConfiguration" : {
      "containerName" : "envoy",
      "properties" : [
         {
            "name" : "ProxyIngressPort",
            "value" : "15000"
         },
         {
            "name" : "AppPorts",
            "value" : "9080"
         },
         {
            "name" : "EgressIgnoredIPs",
            "value" : "169.254.170.2,169.254.169.254"
         },
         {
            "name": "EgressIgnoredPorts",
            "value": "22"
         },
         {
            "name" : "IgnoredUID",
            "value" : "1337"
         },
         {
            "name" : "ProxyEgressPort",
            "value" : "15001"
         }
      ],
      "type" : "APPMESH"
   },
   "containerDefinitions" : [
      {
         "name" : "appName",
         "image" : "appImage",
         "portMappings" : [
            {
               "containerPort" : 9080,
```

```
                "protocol" : "tcp"
            }
         ],
         "essential" : true,
         "dependsOn" : [
            {
               "containerName" : "envoy",
               "condition" : "HEALTHY"
            }
         ]
      },
      {
         "name" : "envoy",
         "image" : "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod",
         "essential" : true,
         "environment" : [
            {
               "name" : "APPMESH_VIRTUAL_NODE_NAME",
               "value" : "mesh/apps/virtualNode/serviceB"
            }
         ],
         "healthCheck" : {
            "command" : [
               "CMD-SHELL",
               "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
            ],
            "interval" : 5,
            "retries" : 3,
            "startPeriod" : 10,
            "timeout" : 2
         },
         "memory" : 500,
         "user" : "1337"
      }
   ],
   "requiresCompatibilities" : [ "FARGATE" ],
   "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
   "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
   "networkMode" : "awsvpc"
}
```

Example Definição de tarefas JSON para Amazon ECS com - AWS X-Ray Fargate em contêineres Linux

O X-Ray permite que você colete dados sobre solicitações que um aplicativo atende e fornece ferramentas que você pode usar para visualizar o fluxo de tráfego. O uso do driver X-Ray para Envoy permite que o Envoy relate informações de rastreamento ao X-Ray. Para habilitar o rastreamento do X-Ray, use a configuração do Envoy. Com base na configuração, o Envoy envia dados de rastreamento para o daemon do X-Ray em execução como um contêiner sidecar, e o daemon encaminha os rastreamentos para o serviço X-Ray. Assim que os rastreamentos são publicados no X-Ray, você pode usar o console do X-Ray para visualizar o gráfico de chamada de serviço e solicitar detalhes de rastreamento. O seguinte JSON representa uma definição de tarefa para habilitar a integração do X-Ray.

```
{


   "family" : "taskB",
   "memory" : "1024",
   "cpu" : "512",
   "proxyConfiguration" : {
      "containerName" : "envoy",
      "properties" : [
         {
            "name" : "ProxyIngressPort",
            "value" : "15000"
         },
         {
            "name" : "AppPorts",
            "value" : "9080"
         },
         {
            "name" : "EgressIgnoredIPs",
            "value" : "169.254.170.2,169.254.169.254"
         },
         {
            "name": "EgressIgnoredPorts",
            "value": "22"
         },
         {
            "name" : "IgnoredUID",
            "value" : "1337"
         },
```

```
            {
               "name" : "ProxyEgressPort",
               "value" : "15001"
            }
         ],
         "type" : "APPMESH"
      },
      "containerDefinitions" : [
         {
            "name" : "appName",
            "image" : "appImage",
            "portMappings" : [
               {
                  "containerPort" : 9080,
                  "protocol" : "tcp"
               }
            ],
            "essential" : true,
            "dependsOn" : [
               {
                  "containerName" : "envoy",
                  "condition" : "HEALTHY"
               }
            ]
         },
         {

            "name" : "envoy",
            "image" : "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod",
            "essential" : true,
            "environment" : [
               {
                  "name" : "APPMESH_VIRTUAL_NODE_NAME",
                  "value" : "mesh/apps/virtualNode/serviceB"
               },
               {
                  "name": "ENABLE_ENVOY_XRAY_TRACING",
                  "value": "1"
               }
            ],
            "healthCheck" : {
               "command" : [
                  "CMD-SHELL",
```

```
                "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
            ],
            "interval" : 5,
            "retries" : 3,
            "startPeriod" : 10,
            "timeout" : 2
         },
         "memory" : 500,
         "user" : "1337"
      },
      {
         "name" : "xray-daemon",
         "image" : "amazon/aws-xray-daemon",
         "user" : "1337",
         "essential" : true,
         "cpu" : "32",
         "memoryReservation" : "256",
         "portMappings" : [
            {
               "containerPort" : 2000,
               "protocol" : "udp"
            }
         ]
      }
   ],
   "requiresCompatibilities" : [ "FARGATE" ],
   "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
   "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
   "networkMode" : "awsvpc"
}
```

Example JSON para definição de tarefa do Amazon ECS: tipo de inicialização do EC2

```
{
  "family": "taskB",
  "memory": "256",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "envoy",
    "properties": [
      {
        "name": "IgnoredUID",
        "value": "1337"
```

```
      },
      {
        "name": "ProxyIngressPort",
        "value": "15000"
      },
      {
        "name": "ProxyEgressPort",
        "value": "15001"
      },
      {
        "name": "AppPorts",
        "value": "9080"
      },
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      },
      {
        "name": "EgressIgnoredPorts",
        "value": "22"
      }
    ]
  },
  "containerDefinitions": [
    {
      "name": "appName",
      "image": "appImage",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
```

```
      "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/apps/virtualNode/serviceB"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    }
  ],
  "requiresCompatibilities" : [ "EC2" ],
  "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
  "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Example Definição de tarefa JSON para Amazon ECS com AWS X-Ray - tipo de execução EC2

```
{
  "family": "taskB",
  "memory": "256",
   "cpu" : "1024",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "envoy",
    "properties": [
      {
        "name": "IgnoredUID",
        "value": "1337"
      },
      {
```

```
        "name": "ProxyIngressPort",
        "value": "15000"
      },
      {
        "name": "ProxyEgressPort",
        "value": "15001"
      },
      {
        "name": "AppPorts",
        "value": "9080"
      },
      {
        "name": "EgressIgnoredIPs",
        "value": "169.254.170.2,169.254.169.254"
      },
      {
        "name": "EgressIgnoredPorts",
        "value": "22"
      }
    ]
  },
  "containerDefinitions": [
    {
      "name": "appName",
      "image": "appImage",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod",
```

```
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/apps/virtualNode/serviceB"
        },
        {
         "name": "ENABLE_ENVOY_XRAY_TRACING",
         "value": "1"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | grep state | grep -q LIVE"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      },
      "user": "1337"
    },
    {
      "name": "xray-daemon",
      "image": "amazon/aws-xray-daemon",
      "user": "1337",
      "essential": true,
      "cpu": 32,
      "memoryReservation": 256,
      "portMappings": [
        {
          "containerPort": 2000,
          "protocol": "udp"
        }
      ]
    }
  ],
  "requiresCompatibilities" : [ "EC2" ],
  "taskRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskRole",
  "executionRoleArn" : "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Tópicos avançados

Implantações canário usando o App Mesh

As implantações e lançamentos canário ajudam você a alternar o tráfego entre uma versão antiga de um aplicativo e uma versão recém-implantada. Ele também monitora a integridade da versão recém-implantada. Se houver algum problema com a nova versão, a implantação canário poderá redirecionar automaticamente o tráfego para a versão antiga. As implantações canário oferecem a capacidade de alternar o tráfego entre as versões do aplicativo com mais controle.

Para mais informações sobre como implementar implantações canário para o Amazon ECS usando o App Mesh, consulte Criar um pipeline com implantações canário para o Amazon ECS usando o App Mesh

> **Note**
>
> Para ver mais exemplos e orientações sobre o App Mesh, consulte o repositório de exemplos do App Mesh.

Introdução ao AWS App Mesh Kubernetes

Ao se integrar AWS App Mesh ao Kubernetes usando o controlador App Mesh para Kubernetes, você gerencia recursos do App Mesh, como malhas, serviços virtuais, nós virtuais, roteadores virtuais e rotas por meio do Kubernetes. Também é possível adicionar automaticamente imagens de contêiner do arquivo associado do App Mesh às especificações do pod do Kubernetes. Este tutorial orienta a instalação do controlador do App Mesh para Kubernetes de forma a permitir essa integração.

O controlador é acompanhado pela implantação das seguintes definições de recursos personalizados do Kubernetes: `meshes`, `virtual services`, `virtual nodes` e `virtual routers`. O controlador observa a criação, a modificação e a exclusão dos recursos personalizados e faz alterações nos recursos the section called “Malhas”, the section called “Serviços virtuais”, the section called “Nós virtuais”, the section called “Gateways virtuais”, the section called “Rotas de gateway” e the section called “Roteadores virtuais” (incluindo the section called “Rotas”) correspondentes do App Mesh por meio da API do App Mesh. Para saber mais ou contribuir com o controlador, consulte o GitHubprojeto.

O controlador também instala um webhook que injeta os seguintes contêineres em pods do Kubernetes etiquetados com um nome que você especificar.

- Proxy do App Mesh Envoy: o Envoy usa a configuração definida no ambiente de gerenciamento do App Mesh para determinar para onde enviar o tráfego do seu aplicativo.
- Gerenciador de rotas de proxy do App Mesh: atualiza as regras `iptables` em um namespace de rede do pod que roteia o tráfego de entrada e saída pelo Envoy. Esse contêiner é executado como um contêiner de init do Kubernetes dentro do pod.

Pré-requisitos

- Uma compreensão existente dos conceitos do App Mesh. Para ter mais informações, consulte O que é o AWS App Mesh?.
- Uma compreensão existente dos conceitos de Kubernetes. Para obter mais informações, consulte O que é o Kubernetes na documentação do Kubernetes.
- Um cluster do Kubernetes existente. Se você não tiver um cluster existente, consulte Conceitos básicos do Amazon EKS no Guia do usuário do Amazon EKS. Se você estiver executando seu próprio cluster Kubernetes no Amazon EC2, certifique-se de que o Docker esteja autenticado no repositório Amazon ECR em que a imagem do Envoy está. Para obter mais informações, consulte Imagem do Envoy, Autenticação do registro no Guia do usuário do Amazon Elastic Container Registry e Como extrair uma imagem de um registro privado na documentação do Kubernetes.
- O App Mesh oferece suporte a serviços Linux registrados com DNS ou ambos. AWS Cloud Map Para usar este guia de conceitos básicos, recomendamos que você tenha três serviços existentes registrados no DNS. Para os procedimentos nesse tópico, espera-se que os serviços existentes sejam nomeados `serviceA`, `serviceB` e `serviceBv2`; e que todos os serviços sejam detectáveis por meio de um namespace chamado `apps.local`.

  É possível criar uma malha de serviço e seus recursos mesmo que os serviços não existam, mas não será possível usar a malha enquanto não tiver implantado serviços reais.
- A AWS CLI versão 1.18.116 ou posterior ou 2.0.38 ou posterior instalada. Para instalar ou atualizar o AWS CLI, consulte Instalando AWS CLI o.
- Um cliente do `kubectl` que está configurado para se comunicar com o cluster do Kubernetes. Se você estiver usando o Amazon Elastic Kubernetes Service, use as instruções para instalar `kubectl` e configurar um arquivo `kubeconfig`.

- Ter o Helm versão 3.0 ou posterior instalado. Se você não tiver o Helm instalado, consulte Como usar o Helm com o Amazon EKS no Guia do usuário do Amazon EKS.
- Atualmente, o Amazon EKS só oferece suporte `IPv4_ONLY` e `IPv6_ONLY` somente às preferências de IP, porque o Amazon EKS atualmente só oferece suporte a pods capazes de servir somente tráfego `IPv4` ou somente tráfego `IPv6`.

Supõe-se nas etapas restantes que os serviços reais sejam nomeados `serviceA`, `serviceB` e `serviceBv2` e que todos os serviços sejam detectáveis por meio de um namespace chamado `apps.local`.

Etapa 1: Instalar os componentes de integração

Instale os componentes de integração uma vez em cada cluster que hospeda os pods a serem usados com o App Mesh.

Como instalar os componentes de integração

1. As etapas restantes deste procedimento exigem um cluster sem uma versão de pré-lançamento do controlador instalada. Se você instalou uma versão de pré-lançamento ou não tem certeza se você a tem, baixe e execute um script para verificar se uma versão de pré-lançamento está instalada no seu cluster.

   ```
   curl -o pre_upgrade_check.sh https://raw.githubusercontent.com/aws/eks-charts/
   master/stable/appmesh-controller/upgrade/pre_upgrade_check.sh
   sh ./pre_upgrade_check.sh
   ```

   Se o script retornar `Your cluster is ready for upgrade. Please proceed to the installation instructions`, você poderá avançar para a próxima etapa. Se for diferente, você precisará concluir as etapas de atualização antes de continuar. Para obter mais informações sobre como atualizar uma versão de pré-lançamento, consulte Atualizar em. GitHub
2. Adicione o repositório `eks-charts` ao Helm.

   ```
   helm repo add eks https://aws.github.io/eks-charts
   ```
3. Instale as definições de recursos personalizados (CRD) do aplicativo Mesh Kubernetes.

   ```
   kubectl apply -k "https://github.com/aws/eks-charts/stable/appmesh-controller/crds?
   ref=master"
   ```

4. Crie um namespace do Kubernetes para o controlador.

```
kubectl create ns appmesh-system
```

5. Defina as seguintes variáveis para uso nas etapas mais adiante. Substitua *cluster-name* e *Region-code* pelos valores do cluster existente.

```
export CLUSTER_NAME=cluster-name
export AWS_REGION=Region-code
```

6. (Opcional) Se você deseja executar o controlador no Fargate, é necessário criar um perfil do Fargate. Se você não tiver o `eksctl` instalado, consulte Como instalar ou atualizar `eksctl` no Guia do usuário do Amazon EKS. Se preferir criar o perfil usando o console, consulte Como criar um perfil do Fargate no Guia do usuário do Amazon EKS.

```
eksctl create fargateprofile --cluster $CLUSTER_NAME --name appmesh-system --
namespace appmesh-system
```

7. Crie um provedor de identidade OpenID Connect (OIDC) para o cluster. Se você não tiver o `eksctl` instalado, poderá instalá-lo com as instruções em Como instalar ou atualizar `eksctl` no Guia de usuário do Amazon EKS. Se você preferir criar o provedor usando o console, consulte Como habilitar perfis do IAM para contas de serviço no cluster no Guia do usuário do Amazon EKS.

```
eksctl utils associate-iam-oidc-provider \
    --region=$AWS_REGION \
    --cluster $CLUSTER_NAME \
    --approve
```

8. Crie uma função do IAM, AWSAppMeshFullAccessanexe as políticas AWSCloudMapFullAccess AWS gerenciadas a ela e vincule-a à conta de serviço do `appmesh-controller` Kubernetes. A função permite que o controlador adicione, remova e altere os recursos do App Mesh.

> **Note**
>
> O comando cria uma função AWS do IAM com um nome gerado automaticamente. Não é possível especificar o nome do perfil do IAM criado.

```
eksctl create iamserviceaccount \
    --cluster $CLUSTER_NAME \
    --namespace appmesh-system \
    --name appmesh-controller \
    --attach-policy-arn  arn:aws:iam::aws:policy/
AWSCloudMapFullAccess,arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
    --override-existing-serviceaccounts \
    --approve
```

Se você preferir criar a conta de serviço usando o AWS Management Console ou AWS CLI, consulte Criação de uma função e política do IAM para sua conta de serviço no Guia do usuário do Amazon EKS. Se você usar AWS Management Console ou AWS CLI para criar a conta, também precisará mapear a função para uma conta de serviço do Kubernetes. Para obter mais informações, consulte Como especificar um perfil do IAM para sua conta de serviço no Guia do usuário do Amazon EKS.

9. Implante o controlador do App Mesh. Para obter uma lista de todas as opções de configuração, consulte Configuração ativada GitHub.

   1. Para implantar o controlador do App Mesh em um cluster privado, você precisa primeiro habilitar o App Mesh e os endpoints da Amazon VPC de descoberta de serviços na sub-rede privada vinculada. Você também precisa definir o `accountId`.

      ```
      --set accountId=$AWS_ACCOUNT_ID
      ```

      Para habilitar o rastreamento do X-Ray em um cluster privado, habilite os endpoints da Amazon VPC do X-Ray e do Amazon ECR. O controlador usa `public.ecr.aws/xray/aws-xray-daemon:latest` por padrão, portanto extraia essa imagem para o local e envie-a para o seu repositório pessoal do ECR.

      > **Note**
      >
      > No momento, os endpoints da Amazon VPC não oferecem suporte aos repositórios públicos do Amazon ECR.

      O exemplo a seguir mostra a implantação do controlador com configurações do X-Ray.

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
    --namespace appmesh-system \
    --set region=$AWS_REGION \
    --set serviceAccount.create=false \
    --set serviceAccount.name=appmesh-controller \
    --set accountId=$AWS_ACCOUNT_ID \
    --set log.level=debug \
    --set tracing.enabled=true \
    --set tracing.provider=x-ray \
    --set xray.image.repository=your-account-id.dkr.ecr.your-
region.amazonaws.com/your-repository \
    --set xray.image.tag=your-xray-daemon-image-tag
```

Verifique se o daemon do X-Ray foi injetado com sucesso ao vincular a implantação do aplicativo ao seu nó virtual ou gateway.

Para obter mais informações, consulte Clusters privados no Guia do usuário do Amazon EKS.

2. Implante o controlador do App Mesh para outros clusters. Para obter uma lista de todas as opções de configuração, consulte Configuração ativada GitHub.

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
    --namespace appmesh-system \
    --set region=$AWS_REGION \
    --set serviceAccount.create=false \
    --set serviceAccount.name=appmesh-controller
```

> **Note**
>
> Se sua família de cluster do Amazon EKS for `IPv6`, defina o nome do cluster ao implantar o controlador do App Mesh adicionando a seguinte opção ao comando `--set clusterName=$CLUSTER_NAME` anterior.

> **Important**
>
> Se seu cluster estiver em `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` ou `af-south-1`, você precisará adicionar as seguintes opções ao comando anterior:

Substitua o *ID da conta* e o *código da região* por um dos conjuntos de valores apropriados.

- Para a imagem associada:
  - ```
    --set image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/amazon/appmesh-controller
    ```
  - 772975370895.dkr. ecr.me-south-1.amazonaws.com /:v1.27.3.0-prod aws-appmesh-envoy
  - 856666278305.dkr. ecr.ap-east-1.amazonaws.com /:v1.27.3.0-prod aws-appmesh-envoy
  - 909464085924.dkr. ecr.ap-southeast-3.amazonaws.com /:v1.27.3.0-prod aws-appmesh-envoy
  - 422531588944.dkr. ecr.eu-south-1.amazonaws.com /:v1.27.3.0-prod aws-appmesh-envoy
  - 564877687649.dkr. ecr.il-central-1.amazonaws.com /:v1.27.3.0-prod aws-appmesh-envoy
  - 924023996002.dkr. ecr.af-south-1.amazonaws.com /:v1.27.3.0-prod aws-appmesh-envoy
- Os URIs de imagem mais antigos podem ser encontrados no GitHublog de alterações. As AWS contas nas quais as imagens estão presentes mudaram de versão`v1.5.0`. As versões mais antigas das imagens são hospedadas em contas da AWS encontradas nos registros de imagens de contêineres da Amazon no Amazon Elastic Kubernetes Service.
- Para a imagem do controlador:
  - ```
    --set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
    ```
  - 772975370895.dkr.ecr.me-south-1.amazonaws.com/amazon/appmesh-controller:v1.12.3
  - 856666278305.dkr.ecr.ap-east-1.amazonaws.com/amazon/appmesh-controller:v1.12.3
  - 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/amazon/appmesh-controller:v1.12.3

    - 422531588944.dkr.ecr.eu-south-1.amazonaws.com/amazon/appmesh-controller:v1.12.3
    - 564877687649.dkr.ecr.il-central-1.amazonaws.com/amazon/appmesh-controller:v1.12.3
    - 924023996002.dkr.ecr.af-south-1.amazonaws.com/amazon/appmesh-controller:v1.12.3
  - Para a imagem associada de inicialização:
    - `--set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy`
    - 772975370895.dkr. ecr.me-south-1.amazonaws.com/-manager:v7-prod aws-appmesh-proxy-route
    - 856666278305.dkr. ecr.ap-east-1.amazonaws.com/-manager:v7-prod aws-appmesh-proxy-route
    - 909464085924.dkr. ecr.ap-southeast-3.amazonaws.com/-manager:v7-prod aws-appmesh-proxy-route
    - 422531588944.dkr. ecr.eu-south-1.amazonaws.com/-manager:v7-prod aws-appmesh-proxy-route
    - 564877687649.dkr. ecr.il-central-1.amazonaws.com/-manager:v7-prod aws-appmesh-proxy-route
    - 924023996002.dkr. ecr.af-south-1.amazonaws.com/-manager:v7-prod aws-appmesh-proxy-route

> **Important**
>
> Somente a versão v1.9.0.0-prod ou posterior é compatível para uso com o App Mesh.

10. Confirme se a versão do controlador é `v1.4.0` ou posterior. Você pode revisar o registro de alterações GitHub.

```
kubectl get deployment appmesh-controller \
    -n appmesh-system \
    -o json  | jq -r ".spec.template.spec.containers[].image" | cut -f2 -d ':'
```

> **Note**
>
> Se visualizar o log para o contêiner em execução, você poderá ver uma linha que inclui o seguinte texto, que pode ser ignorado com segurança.
>
> ```
> Neither -kubeconfig nor -master was specified. Using the inClusterConfig.
>  This might not work.
> ```

Etapa 2: Como implantar recursos do App Mesh

Quando você implanta um aplicativo no Kubernetes, você também cria os recursos personalizados do Kubernetes para que o controlador possa criar os recursos do App Mesh correspondentes. O procedimento a seguir ajuda a implantar recursos do App Mesh com alguns atributos deles. Você pode encontrar exemplos de manifestos para implantar outros recursos de recursos do App Mesh nas `v1beta2` subpastas de muitas das pastas de recursos listadas nas orientações do App Mesh em. GitHub

> **Important**
>
> Depois que o controlador criar um recurso do App Mesh, recomendamos fazer apenas alterações ou excluir o recurso do App Mesh usando o controlador. Se você fizer alterações ou excluir o recurso usando o App Mesh, o controlador não alterará nem recriará o recurso alterado ou excluído do App Mesh por dez horas, por padrão. Você pode configurar essa duração para ser menor. Para obter mais informações, consulte Configuração ativada GitHub.

Como implantar recursos do App Mesh

1. Crie um namespace do Kubernetes no qual implantar recursos do App Mesh.

   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `namespace.yaml`:

   ```
   apiVersion: v1
   kind: Namespace
   metadata:
   ```

```
  name: my-apps
  labels:
    mesh: my-mesh
    appmesh.k8s.aws/sidecarInjectorWebhook: enabled
```

b. Crie o novo namespace.

```
kubectl apply -f namespace.yaml
```

2. Crie uma malha de serviços do App Mesh.

a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `mesh.yaml`: O arquivo é usado para criar um recurso de malha chamado *my-mesh*. Uma malha de serviços é um limite lógico para o tráfego de rede entre os serviços que residem nela.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: Mesh
metadata:
  name: my-mesh
spec:
  namespaceSelector:
    matchLabels:
      mesh: my-mesh
```

b. Crie a malha.

```
kubectl apply -f mesh.yaml
```

c. Veja os detalhes do recurso de malha do Kubernetes que foi criado.

```
kubectl describe mesh my-mesh
```

Saída

```
Name:         my-mesh
Namespace:
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"Mesh","metadata":{"annotations":{},"name":"my-mesh"},"spec":
{"namespaceSelector":{"matchLa...
API Version:  appmesh.k8s.aws/v1beta2
```

```
Kind:         Mesh
Metadata:
  Creation Timestamp:  2020-06-17T14:51:37Z
  Finalizers:
    finalizers.appmesh.k8s.aws/mesh-members
    finalizers.appmesh.k8s.aws/aws-appmesh-resources
  Generation:        1
  Resource Version:  6295
  Self Link:         /apis/appmesh.k8s.aws/v1beta2/meshes/my-mesh
  UID:               111a11b1-c11d-1e1f-gh1i-j11k1l111m711
Spec:
  Aws Name:  my-mesh
  Namespace Selector:
    Match Labels:
      Mesh:  my-mesh
Status:
  Conditions:
    Last Transition Time:  2020-06-17T14:51:37Z
    Status:                True
    Type:                  MeshActive
  Mesh ARN:                arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh
  Observed Generation:     1
Events:                    <none>
```

d. Veja os detalhes sobre a malha de serviços do App Mesh que o controlador criou.

```
aws appmesh describe-mesh --mesh-name my-mesh
```

Saída

```
{
    "mesh": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh",
            "createdAt": "2020-06-17T09:51:37.920000-05:00",
            "lastUpdatedAt": "2020-06-17T09:51:37.920000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {},
```

```
        "status": {
            "status": "ACTIVE"
        }
    }
}
```

3. Crie um nó virtual do App Mesh. Um nó virtual atua como um ponteiro lógico para uma implantação do Kubernetes.

   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `virtual-node.yaml`: O arquivo será usado para criar um nó virtual do App Mesh chamado *my-service-a* no namespace *my-apps*. O nó virtual representa um serviço do Kubernetes criado em uma etapa posterior. O valor de `hostname` é o nome de hospedagem DNS totalmente qualificado do serviço real que este nó virtual representa.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: my-service-a
  namespace: my-apps
spec:
  podSelector:
    matchLabels:
      app: my-app-1
  listeners:
    - portMapping:
        port: 80
        protocol: http
  serviceDiscovery:
    dns:
      hostname: my-service-a.my-apps.svc.cluster.local
```

   Os nós virtuais têm recursos, como end-to-end criptografia e verificações de integridade, que não são abordados neste tutorial. Para ter mais informações, consulte the section called “Nós virtuais”. Para ver todas as configurações disponíveis para um nó virtual que é possível definir na especificação anterior, execute o comando a seguir.

```
aws appmesh create-virtual-node --generate-cli-skeleton yaml-input
```

   b. Implante o nó virtual.

```
kubectl apply -f virtual-node.yaml
```

c. Veja os detalhes do recurso de nó virtual do Kubernetes que foi criado.

```
kubectl describe virtualnode my-service-a -n my-apps
```

Saída

```
Name:         my-service-a
Namespace:    my-apps
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualNode","metadata":{"annotations":{},"name":"my-service-
a","namespace":"my-apps"},"s...
API Version:  appmesh.k8s.aws/v1beta2
Kind:         VirtualNode
Metadata:
  Creation Timestamp:  2020-06-17T14:57:29Z
  Finalizers:
    finalizers.appmesh.k8s.aws/aws-appmesh-resources
  Generation:        2
  Resource Version:  22545
  Self Link:         /apis/appmesh.k8s.aws/v1beta2/namespaces/my-apps/
virtualnodes/my-service-a
  UID:               111a11b1-c11d-1e1f-gh1i-j11k1l111m711
Spec:
  Aws Name:  my-service-a_my-apps
  Listeners:
    Port Mapping:
      Port:      80
      Protocol:  http
  Mesh Ref:
    Name:  my-mesh
    UID:   111a11b1-c11d-1e1f-gh1i-j11k1l111m711
  Pod Selector:
    Match Labels:
      App:  nginx
  Service Discovery:
    Dns:
      Hostname:  my-service-a.my-apps.svc.cluster.local
Status:
```

```
  Conditions:
    Last Transition Time:  2020-06-17T14:57:29Z
    Status:                True
    Type:                  VirtualNodeActive
  Observed Generation:     2
  Virtual Node ARN:        arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps
Events:                    <none>
```

d. Veja os detalhes do nó virtual que o controlador criou no App Mesh.

> **Note**
>
> Embora o nome do nó virtual criado no Kubernetes seja *my-service-a*, o nome do nó virtual criado no App Mesh é *my-service-a_my-apps*. O controlador acrescenta o nome do namespace do Kubernetes ao nome do nó virtual do App Mesh ao criar o recurso do App Mesh. O nome do namespace é adicionado porque no Kubernetes é possível criar nós virtuais com o mesmo nome em namespaces diferentes, mas no App Mesh, um nome de nó virtual deve ser exclusivo em uma malha.

```
aws appmesh describe-virtual-node --mesh-name my-mesh --virtual-node-name my-
service-a_my-apps
```

Saída

```
{
    "virtualNode": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps",
            "createdAt": "2020-06-17T09:57:29.840000-05:00",
            "lastUpdatedAt": "2020-06-17T09:57:29.840000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {
```

```
            "backends": [],
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ],
            "serviceDiscovery": {
                "dns": {
                    "hostname": "my-service-a.my-apps.svc.cluster.local"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualNodeName": "my-service-a_my-apps"
    }
}
```

4. Crie um roteador virtual do App Mesh. Os roteadores virtuais manipulam o tráfego de um ou mais serviços virtuais dentro da malha.

   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `virtual-router.yaml`: O arquivo é usado para criar um tráfego de roteador a roteador virtual para o nó virtual chamado `my-service-a` que foi criado na etapa anterior. O controlador cria o roteador virtual e os recursos de roteamento do App Mesh. É possível especificar muitos mais recursos para suas rotas e usar protocolos diferentes de `http`. Para obter mais informações, consulte the section called “Rotas” e the section called “Roteadores virtuais”. Perceba que o nome do nó virtual referenciado é o nome do nó virtual do Kubernetes, não o nome do nó virtual do App Mesh que foi criado no App Mesh pelo controlador.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualRouter
metadata:
  namespace: my-apps
  name: my-service-a-virtual-router
spec:
  listeners:
    - portMapping:
```

```
        port: 80
        protocol: http
  routes:
    - name: my-service-a-route
      httpRoute:
        match:
          prefix: /
        action:
          weightedTargets:
            - virtualNodeRef:
                name: my-service-a
              weight: 1
```

(Opcional) Para ver todas as configurações disponíveis para roteador virtual que podem ser definidas na especificação anterior, execute o comando a seguir.

```
aws appmesh create-virtual-router --generate-cli-skeleton yaml-input
```

Para ver todas as configurações disponíveis para uma rota que podem ser definidas na especificação anterior, execute o comando a seguir.

```
aws appmesh create-route --generate-cli-skeleton yaml-input
```

b. Implante o roteador virtual.

```
kubectl apply -f virtual-router.yaml
```

c. Veja o recurso de roteador virtual do Kubernetes que foi criado.

```
kubectl describe virtualrouter my-service-a-virtual-router -n my-apps
```

Resultado abreviado

```
Name:         my-service-a-virtual-router
Namespace:    my-apps
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualRouter","metadata":{"annotations":{},"name":"my-
service-a-virtual-router","namespac...
API Version:  appmesh.k8s.aws/v1beta2
```

```
Kind:         VirtualRouter
...
Spec:
  Aws Name:  my-service-a-virtual-router_my-apps
  Listeners:
    Port Mapping:
      Port:      80
      Protocol:  http
  Mesh Ref:
    Name:  my-mesh
    UID:   111a11b1-c11d-1e1f-gh1i-j11k1l111m711
  Routes:
    Http Route:
      Action:
        Weighted Targets:
          Virtual Node Ref:
            Name:  my-service-a
          Weight:  1
      Match:
        Prefix:  /
    Name:        my-service-a-route
Status:
  Conditions:
    Last Transition Time:  2020-06-17T15:14:01Z
    Status:                True
    Type:                  VirtualRouterActive
  Observed Generation:     1
  Route AR Ns:
    My - Service - A - Route:  arn:aws:appmesh:us-west-2:111122223333:mesh/my-
mesh/virtualRouter/my-service-a-virtual-router_my-apps/route/my-service-a-route
  Virtual Router ARN:          arn:aws:appmesh:us-west-2:111122223333:mesh/my-
mesh/virtualRouter/my-service-a-virtual-router_my-apps
Events:                        <none>
```

d. Veja o recurso de roteador virtual que o controlador criou no App Mesh. Especifique `my-service-a-virtual-router_my-apps` para `name`, porque quando o controlador criou o roteador virtual no App Mesh, ele anexou o nome do namespace do Kubernetes ao nome do roteador virtual.

```
aws appmesh describe-virtual-router --virtual-router-name my-service-a-virtual-
router_my-apps --mesh-name my-mesh
```

Saída

```
{
    "virtualRouter": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualRouter/my-service-a-virtual-router_my-apps",
            "createdAt": "2020-06-17T10:14:01.547000-05:00",
            "lastUpdatedAt": "2020-06-17T10:14:01.547000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "my-service-a-virtual-router_my-apps"
    }
}
```

e. Veja o recurso de rota que o controlador criou no App Mesh. Um recurso de rota não foi criado no Kubernetes porque a rota faz parte da configuração do roteador virtual no Kubernetes. As informações da rota foram mostradas nos detalhes do recurso do Kubernetes na subetapa c. O controlador não acrescentou o nome do namespace do Kubernetes ao nome da rota do App Mesh ao criar a rota no App Mesh porque os nomes de rota são exclusivos de um roteador virtual.

```
aws appmesh describe-route \
    --route-name my-service-a-route \
```

```
    --virtual-router-name my-service-a-virtual-router_my-apps \
    --mesh-name my-mesh
```

Saída


```
{
    "route": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualRouter/my-service-a-virtual-router_my-apps/route/my-service-a-route",
            "createdAt": "2020-06-17T10:14:01.577000-05:00",
            "lastUpdatedAt": "2020-06-17T10:14:01.577000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "routeName": "my-service-a-route",
        "spec": {
            "httpRoute": {
                "action": {
                    "weightedTargets": [
                        {
                            "virtualNode": "my-service-a_my-apps",
                            "weight": 1
                        }
                    ]
                },
                "match": {
                    "prefix": "/"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "my-service-a-virtual-router_my-apps"
    }
}
```


5. Crie um serviço virtual do App Mesh. Um serviço virtual é uma abstração de um serviço real que é fornecido por um nó virtual direta ou indiretamente por meio de um roteador virtual. Os

serviços dependentes chamam o serviço virtual pelo nome. Embora o nome não seja importante para o App Mesh, recomendamos nomear o serviço virtual com o nome de domínio totalmente qualificado do serviço real que o serviço virtual representa. Ao nomear seus serviços virtuais dessa forma, você não precisa alterar o código do aplicativo para fazer referência a um nome diferente. As solicitações são roteadas para o nó virtual ou o roteador virtual especificado como o provedor para o serviço virtual.

a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `virtual-service.yaml`: O arquivo é usado para criar um serviço virtual que usa um provedor de roteador virtual para direcionar o tráfego para o nó virtual chamado `my-service-a` criado em uma etapa anterior. O valor para `awsName` em `spec` é o nome de domínio totalmente qualificado (FQDN) do serviço do Kubernetes real extraído por este serviço virtual. O serviço Kubernetes é criado no the section called “Etapa 3: Criar ou atualizar serviços”. Para ter mais informações, consulte the section called “Serviços virtuais”.

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualService
metadata:
  name: my-service-a
  namespace: my-apps
spec:
  awsName: my-service-a.my-apps.svc.cluster.local
  provider:
    virtualRouter:
      virtualRouterRef:
        name: my-service-a-virtual-router
```

Para ver todas as configurações disponíveis para um serviço virtual que é possível definir na especificação anterior, execute o comando a seguir.

```
aws appmesh create-virtual-service --generate-cli-skeleton yaml-input
```

b. Crie o serviço virtual.

```
kubectl apply -f virtual-service.yaml
```

c. Veja os detalhes do recurso de serviço virtual do Kubernetes que foi criado.

```
kubectl describe virtualservice my-service-a -n my-apps
```

Saída

```
Name:         my-service-a
Namespace:    my-apps
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"appmesh.k8s.aws/
v1beta2","kind":"VirtualService","metadata":{"annotations":{},"name":"my-
service-a","namespace":"my-apps"}...
API Version:  appmesh.k8s.aws/v1beta2
Kind:         VirtualService
Metadata:
  Creation Timestamp:  2020-06-17T15:48:40Z
  Finalizers:
    finalizers.appmesh.k8s.aws/aws-appmesh-resources
  Generation:        1
  Resource Version:  13598
  Self Link:         /apis/appmesh.k8s.aws/v1beta2/namespaces/my-apps/
virtualservices/my-service-a
  UID:               111a11b1-c11d-1e1f-gh1i-j11k1l111m711
Spec:
  Aws Name:  my-service-a.my-apps.svc.cluster.local
  Mesh Ref:
    Name:  my-mesh
    UID:   111a11b1-c11d-1e1f-gh1i-j11k1l111m711
  Provider:
    Virtual Router:
      Virtual Router Ref:
        Name:  my-service-a-virtual-router
Status:
  Conditions:
    Last Transition Time:  2020-06-17T15:48:40Z
    Status:                True
    Type:                  VirtualServiceActive
  Observed Generation:     1
  Virtual Service ARN:     arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualService/my-service-a.my-apps.svc.cluster.local
Events:                    <none>
```

d. Veja os detalhes do recurso de serviço virtual que o controlador criou no App Mesh. O controlador do Kubernetes não acrescentou o nome do namespace do Kubernetes ao nome

do serviço virtual do App Mesh ao criar o serviço virtual no App Mesh porque o nome do serviço virtual é um FQDN exclusivo.

```
aws appmesh describe-virtual-service --virtual-service-name my-service-a.my-apps.svc.cluster.local --mesh-name my-mesh
```

Saída

```
{
    "virtualService": {
        "meshName": "my-mesh",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:111122223333:mesh/my-mesh/
virtualService/my-service-a.my-apps.svc.cluster.local",
            "createdAt": "2020-06-17T10:48:40.182000-05:00",
            "lastUpdatedAt": "2020-06-17T10:48:40.182000-05:00",
            "meshOwner": "111122223333",
            "resourceOwner": "111122223333",
            "uid": "111a11b1-c11d-1e1f-gh1i-j11k1l111m711",
            "version": 1
        },
        "spec": {
            "provider": {
                "virtualRouter": {
                    "virtualRouterName": "my-service-a-virtual-router_my-apps"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualServiceName": "my-service-a.my-apps.svc.cluster.local"
    }
}
```

Embora não seja abordado neste tutorial, o controlador também pode implantar the section called “Gateways virtuais” e the section called “Rotas de gateway” do App Mesh. Para ver um passo a passo da implantação desses recursos com o controlador, consulte Configurando o gateway de entrada ou um exemplo de manifesto que inclui os recursos em. GitHub

Etapa 3: Criar ou atualizar serviços

Todos os pods a serem usados com o App Mesh devem ter os contêineres associados do App Mesh adicionados a eles. O injetor adiciona automaticamente os contêineres associados a qualquer pod implantado com uma etiqueta especificada.

1. Habilite a autorização de proxy. Recomendamos que você habilite cada implantação do Kubernetes para transmitir somente a configuração de seu próprio nó virtual do App Mesh.

   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `proxy-auth.json`: Certifique-se de substituir os *valores de cores alternadas* por seus próprios.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "appmesh:StreamAggregatedResources",
               "Resource": [
                   "arn:aws:appmesh:Region-code:111122223333:mesh/my-mesh/
   virtualNode/my-service-a_my-apps"
               ]
           }
       ]
   }
   ```

   b. Crie a política do

   ```
   aws iam create-policy --policy-name my-policy --policy-document file://proxy-
   auth.json
   ```

   c. Crie um perfil do IAM, associe a ela a política criada na etapa anterior, crie uma conta de serviço do Kubernetes e vincule a política à conta de serviço do Kubernetes. A função permite que o controlador adicione, remova e altere os recursos do App Mesh.

   ```
   eksctl create iamserviceaccount \
       --cluster $CLUSTER_NAME \
       --namespace my-apps \
       --name my-service-a \
       --attach-policy-arn  arn:aws:iam::111122223333:policy/my-policy \
   ```

```
    --override-existing-serviceaccounts \
    --approve
```

Se você preferir criar a conta de serviço usando o AWS Management Console ou AWS CLI, consulte Criação de uma função e política do IAM para sua conta de serviço no Guia do usuário do Amazon EKS. Se você usar AWS Management Console ou AWS CLI para criar a conta, também precisará mapear a função para uma conta de serviço do Kubernetes. Para obter mais informações, consulte Como especificar um perfil do IAM para sua conta de serviço no Guia do usuário do Amazon EKS.

2. (Opcional) Se você deseja fazer sua implantação em pods do Fargate, será necessário criar um perfil do Fargate. Se você não tiver o `eksctl` instalado, poderá instalá-lo com as instruções em Como instalar ou atualizar `eksctl` no Guia de usuário do Amazon EKS. Se preferir criar o perfil usando o console, consulte Como criar um perfil do Fargate no Guia do usuário do Amazon EKS.

```
eksctl create fargateprofile --cluster my-cluster --region Region-code --name my-service-a --namespace my-apps
```

3. Crie um serviço e uma implantação do Kubernetes. Se você tem uma implantação existente que deseja usar com o App Mesh, precisará implantar um nó virtual, como fez na subetapa 3 de the section called “Etapa 2: Como implantar recursos do App Mesh”. Atualize sua implantação para garantir que o rótulo corresponda àquele definido no nó virtual, para que os contêineres auxiliares sejam automaticamente adicionados aos pods e os pods sejam reimplantados.

   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `example-service.yaml`: Se você alterar o nome do namespace e estiver usando pods do Fargate, certifique-se de que nome do namespace corresponda ao àquele definido no seu perfil do Fargate.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service-a
  namespace: my-apps
  labels:
    app: my-app-1
spec:
  selector:
    app: my-app-1
```

```
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-service-a
  namespace: my-apps
  labels:
    app: my-app-1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app-1
  template:
    metadata:
      labels:
        app: my-app-1
    spec:
      serviceAccountName: my-service-a
      containers:
      - name: nginx
        image: nginx:1.19.0
        ports:
        - containerPort: 80
```

> **Important**
>
> O valor para `app` `matchLabels` `selector` na especificação deve corresponder àquele especificado na criação do nó virtual na subetapa 3 em the section called "Etapa 2: Como implantar recursos do App Mesh", ou os contêineres associados não serão injetados no pod. No exemplo anterior, o valor para o rótulo é `my-app-1`. Se você implantar um gateway virtual, em vez de um nó virtual, o `Deployment` manifesto deverá incluir somente o contêiner Envoy. Para mais informações sobre a imagem a ser usada, consulte Envoy. Para ver um exemplo de manifestação, consulte o exemplo de implantação em. GitHub

b. Implante o serviço.

```
kubectl apply -f example-service.yaml
```

c. Visualize o serviço e a implantação.

```
kubectl -n my-apps get pods
```

Saída

```
NAME                               READY   STATUS    RESTARTS   AGE
my-service-a-54776556f6-2cxd9   2/2     Running   0          10s
my-service-a-54776556f6-w26kf   2/2     Running   0          18s
my-service-a-54776556f6-zw5kt   2/2     Running   0          26s
```

d. Visualize os detalhes de um dos pods que foi implantado.

```
kubectl -n my-apps describe pod my-service-a-54776556f6-2cxd9
```

Resultado abreviado

```
Name:         my-service-a-54776556f6-2cxd9
Namespace:    my-app-1
Priority:     0
Node:         ip-192-168-44-157.us-west-2.compute.internal/192.168.44.157
Start Time:   Wed, 17 Jun 2020 11:08:59 -0500
Labels:       app=nginx
              pod-template-hash=54776556f6
Annotations:  kubernetes.io/psp: eks.privileged
Status:       Running
IP:           192.168.57.134
IPs:
  IP:           192.168.57.134
Controlled By:  ReplicaSet/my-service-a-54776556f6
Init Containers:
  proxyinit:
    Container ID:   docker://
e0c4810d584c21ae0cb6e40f6119d2508f029094d0e01c9411c6cf2a32d77a59
    Image:          111345817488.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
proxy-route-manager:v2
    Image ID:       docker-pullable://111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager
    Port:           <none>
```

```
    Host Port:      <none>
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Fri, 26 Jun 2020 08:36:22 -0500
      Finished:     Fri, 26 Jun 2020 08:36:22 -0500
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:     10m
      memory:  32Mi
    Environment:
      APPMESH_START_ENABLED:         1
      APPMESH_IGNORE_UID:            1337
      APPMESH_ENVOY_INGRESS_PORT:    15000
      APPMESH_ENVOY_EGRESS_PORT:     15001
      APPMESH_APP_PORTS:             80
      APPMESH_EGRESS_IGNORED_IP:     169.254.169.254
      APPMESH_EGRESS_IGNORED_PORTS:  22
      AWS_ROLE_ARN:                  arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
      AWS_WEB_IDENTITY_TOKEN_FILE:   /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
    ...
Containers:
  nginx:
    Container ID:   docker://
be6359dc6ecd3f18a1c87df7b57c2093e1f9db17d5b3a77f22585ce3bcab137a
    Image:          nginx:1.19.0
    Image ID:       docker-pullable://nginx
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 26 Jun 2020 08:36:28 -0500
    Ready:          True
    Restart Count:  0
    Environment:
      AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
    ...
  envoy:
```

```
    Container ID:
 docker://905b55cbf33ef3b3debc51cb448401d24e2e7c2dbfc6a9754a2c49dd55a216b6
    Image:          840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.12.4.0-prod
    Image ID:       docker-pullable://840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy
    Port:           9901/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 26 Jun 2020 08:36:36 -0500
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:     10m
      memory:  32Mi
    Environment:
      APPMESH_RESOURCE_ARN:         arn:aws:iam::111122223333:mesh/my-mesh/
virtualNode/my-service-a_my-apps
      APPMESH_PREVIEW:              0
      ENVOY_LOG_LEVEL:              info
      AWS_REGION:                   us-west-2
      AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/eksctl-app-
mesh-addon-iamserviceaccount-my-a-Role1-NMNCVWB6PL0N
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
...
Events:
  Type    Reason     Age   From
   Message
  ----    ------     ----  ----
   -------
  Normal  Pulling    30s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Pulling image "111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager:v2"
  Normal  Pulled     23s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Successfully pulled image "111345817488.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-proxy-route-manager:v2"
  Normal  Created    21s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Created container proxyinit
  Normal  Started    21s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Started container proxyinit
  Normal  Pulling    20s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Pulling image "nginx:1.19.0"
```

```
  Normal  Pulled     16s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Successfully pulled image "nginx:1.19.0"
  Normal  Created    15s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Created container nginx
  Normal  Started    15s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Started container nginx
  Normal  Pulling    15s   kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Pulling image "840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy:v1.12.4.0-prod"
  Normal  Pulled     8s    kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Successfully pulled image "840364872350.dkr.ecr.us-
west-2.amazonaws.com/aws-appmesh-envoy:v1.12.4.0-prod"
  Normal  Created    7s    kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Created container envoy
  Normal  Started    7s    kubelet, ip-192-168-44-157.us-
west-2.compute.internal  Started container envoy
```

No resultado anterior, é possível ver que os contêineres `proxyinit` e `envoy` foram adicionados ao pod pelo controlador. Se você implantou o serviço de exemplo no Fargate, o contêiner `envoy` foi adicionado ao pod pelo controlador, mas o contêiner `proxyinit` não foi.

4. (Opcional) Instale complementos como Prometheus, Grafana, Jaeger e AWS X-Ray Datadog. Para obter mais informações, consulte os complementos do App Mesh GitHub e a seção Observabilidade do Guia do usuário do App Mesh.

> **Note**
>
> Para ver mais exemplos e orientações sobre o App Mesh, consulte o repositório de exemplos do App Mesh.

Etapa 4: limpar

Remova todos os recursos de exemplo criados neste tutorial. O controlador também remove os recursos criados na malha de serviços `my-mesh` do App Mesh.

```
kubectl delete namespace my-apps
```

Se você criou um perfil do Fargate para o serviço de exemplo, remova-o.

```
eksctl delete fargateprofile --name my-service-a --cluster my-cluster --region Region-code
```

Exclua a malha.

```
kubectl delete mesh my-mesh
```

(Opcional) É possível remover os componentes de integração do Kubernetes.

```
helm delete appmesh-controller -n appmesh-system
```

(Opcional) Se você implantou os componentes de integração do Kubernetes no Fargate, exclua o perfil do Fargate.

```
eksctl delete fargateprofile --name appmesh-system --cluster my-cluster --
region Region-code
```

Começando a usar AWS App Mesh o Amazon EC2

Este tópico ajuda você a usar AWS App Mesh com um serviço real que está sendo executado no Amazon EC2. Esse tutorial aborda os atributos básicos de vários tipos de recursos do App Mesh.

Cenário

Para ilustrar como usar o App Mesh, suponha que você tenha um aplicativo com as seguintes características:

- Consiste em dois serviços chamados `serviceA` e `serviceB`.
- Ambos os serviços estão registrados em um namespace chamado `apps.local`.
- O `ServiceA` se comunica com o `serviceB` por HTTP/2, porta 80.
- Você já implantou a versão 2 do `serviceB` e a registrou com o nome de `serviceBv2` no namespace `apps.local`.

Você tem os seguintes requisitos:

- Você deseja enviar 75% do tráfego do `serviceA` para o `serviceB` e 25% do tráfego do `serviceBv2` para garantir que o `serviceBv2` esteja livre de bugs antes de enviar 100% do tráfego do `serviceA` para ele.
- Você quer poder ajustar facilmente a ponderação do tráfego para que 100% do tráfego vá para o `serviceBv2` quando for comprovado que ele é confiável. Depois que todo o tráfego estiver sendo enviado para o `serviceBv2`, você deseja descontinuar o `serviceB`.
- Você não quer ter que alterar nenhum código de aplicativo ou registro de descoberta de serviços existente para que seus serviços reais atendam aos requisitos anteriores.

Para atender às suas necessidades, você decide criar uma malha de serviços do App Mesh com serviços virtuais, nós virtuais, um roteador virtual e uma rota. Depois de implementar a malha, você atualiza os serviços que usam o proxy do Envoy. Assim que forem atualizados, os serviços se comunicarão entre si por meio do proxy Envoy em vez de diretamente entre si.

Pré-requisitos

O App Mesh oferece suporte a serviços Linux registrados com DNS ou ambos. AWS Cloud Map Para usar este guia de conceitos básicos, recomendamos que você tenha três serviços existentes registrados no DNS. É possível criar uma malha de serviço e seus recursos mesmo que os serviços não existam, mas não será possível usar a malha enquanto não tiver implantado serviços reais.

Se ainda não tiver serviços em execução, você poderá iniciar instâncias do Amazon EC2 e implantar aplicativos nelas. Para obter mais informações, consulte Tutorial: Conceitos básicos de instâncias do Amazon EC2 para Linux no Guia do usuário do Amazon EC2 para instâncias do Linux. Supõe-se nas etapas restantes que os serviços reais sejam nomeados `serviceA`, `serviceB` e `serviceBv2` e que todos os serviços sejam detectáveis por meio de um namespace chamado `apps.local`.

Etapa 1: Criar uma malha e um serviço virtual

Uma malha de serviços é um limite lógico para o tráfego de rede entre os serviços que residem nela. Para ter mais informações, consulte Malhas de serviço. Um serviço virtual é uma abstração de um serviço real. Para ter mais informações, consulte Serviços virtuais.

Crie os recursos a seguir :

- Uma malha chamada `apps`, uma vez que todos os serviços no cenário estão registrados no namespace `apps.local`.

- Um serviço virtual chamado `serviceb.apps.local`, uma vez que o serviço virtual representa um serviço que é detectável com esse nome e você não quer alterar o código para fazer referência a outro nome. Um serviço virtual chamado `servicea.apps.local` será adicionado em uma etapa posterior.

Você pode usar a AWS CLI versão 1.18.116 AWS Management Console ou superior ou 2.0.38 ou superior para concluir as etapas a seguir. Se estiver usando o AWS CLI, use o `aws --version` comando para verificar sua AWS CLI versão instalada. Se você não tiver a versão 1.18.116 ou superior ou a versão 2.0.38 ou superior instalada, será necessário instalar ou atualizar a AWS CLI. Selecione a guia da ferramenta que deseja usar.

AWS Management Console

1. Abra o assistente de primeira execução do console do App Mesh em https://console.aws.amazon.com/appmesh/get-started.
2. Em Mesh name (Nome da malha), insira **apps**.
3. Em Virtual service name (Nome do serviço virtual), insira **serviceb.apps.local**.
4. Para continuar, escolha Avançar.

AWS CLI

1. Crie uma malha com o comando `create-mesh`.

```
aws appmesh create-mesh --mesh-name apps
```

2. Crie um serviço virtual com o comando `create-virtual-service`.

```
aws appmesh create-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local --spec {}
```

Etapa 2: Criar um nó virtual

Um nó virtual funciona como um apontador lógico para um serviço real. Para ter mais informações, consulte Nós virtuais.

Crie um nó virtual chamado `serviceB`, uma vez que um dos nós virtuais representa o serviço real chamado `serviceB`. O serviço real que o nó virtual representa é detectável por meio do DNS com

um nome de host de `serviceb.apps.local`. Como alternativa, é possível descobrir serviços reais usando o AWS Cloud Map. O nó virtual recebe o tráfego usando o protocolo HTTP/2 na porta 80. Outros protocolos, assim como verificações de integridade, também são compatíveis. Você cria nós virtuais para `serviceA` e `serviceBv2` em uma etapa posterior.

AWS Management Console

1. Em Virtual node name (Nome do nó virtual), insira **`serviceB`**.
2. Em Service discovery method (Método de descoberta de serviços), escolha DNS e insira **`serviceb.apps.local`** para DNS hostname (Nome de host do DNS).
3. Em Listener configuration (Configuração do Listener), escolha http2 para Protocol (Protocolo) e digite **`80`** para Port (Porta).
4. Para continuar, escolha Avançar.

AWS CLI

1. Crie um arquivo denominado `create-virtual-node-serviceb.json` com o conteúdo a seguir:

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceB.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceB"
}
```

2. Crie o nó virtual com o create-virtual-nodecomando usando o arquivo JSON como entrada.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-
serviceb.json
```

Etapa 3: Criar um roteador virtual e uma rota

Os roteadores virtuais cuidam do tráfego de um ou mais serviços virtuais dentro da malha. Para obter mais informações, consulte Roteadores virtuais e Rotas.

Crie os recursos a seguir :

- Um roteador virtual denominado `serviceB`, uma vez que o serviço virtual do `serviceB.apps.local` não inicia a comunicação de saída com nenhum outro serviço. Lembre-se de que o serviço virtual criado anteriormente é uma abstração do serviço `serviceb.apps.local` real. O serviço virtual envia tráfego para o roteador virtual. O roteador virtual recebe o tráfego usando o protocolo HTTP/2 na porta 80. Outros protocolos também são compatíveis.
- Uma rota chamada `serviceB`. Ela roteia 100% de seu tráfego para o nó virtual do `serviceB`. O peso será alterado em uma etapa posterior, depois de adicionar o nó virtual do `serviceBv2`. Embora não seja abordado neste guia, é possível adicionar critérios de filtro adicionais para a rota e adicionar uma política de novas tentativas para fazer com que o proxy Envoy faça várias tentativas de enviar tráfego para um nó virtual quando ele tiver um problema de comunicação.

AWS Management Console

1. Em Virtual router name (Nome do roteador virtual), insira **`serviceB`**.
2. Em Listener configuration (Configuração do Listener), escolha http2 para Protocol (Protocolo) e especifique **`80`** para Port (Porta).
3. Em Route name (Nome da rota), insira **`serviceB`**.
4. Em Route type (Tipo de rota), escolha http2.
5. Para o nome do nó virtual em Configuração de destino, selecione `serviceB` e digite **`100`** para Peso.
6. Em Configuração de correspondência, escolha um Método.
7. Para continuar, escolha Avançar.

AWS CLI

1. Crie um roteador virtual.

   a. Crie um arquivo denominado `create-virtual-router.json` com o conteúdo a seguir:

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ]
    },
    "virtualRouterName": "serviceB"
}
```

   b. Crie o roteador virtual com o create-virtual-routercomando usando o arquivo JSON como entrada.

```
aws appmesh create-virtual-router --cli-input-json file://create-virtual-router.json
```

2. Crie uma rota.

   a. Crie um arquivo denominado `create-route.json` com o conteúdo a seguir:

```
{
    "meshName" : "apps",
    "routeName" : "serviceB",
    "spec" : {
        "httpRoute" : {
            "action" : {
                "weightedTargets" : [
                    {
                        "virtualNode" : "serviceB",
                        "weight" : 100
                    }
```

```
                ]
            },
            "match" : {
                "prefix" : "/"
            }
        }
    },
    "virtualRouterName" : "serviceB"
}
```

b. Crie a rota com o comando create-route usando o arquivo JSON como entrada.

```
aws appmesh create-route --cli-input-json file://create-route.json
```

Etapa 4: revisar e criar

Revise as configurações em relação às instruções anteriores.

AWS Management Console

Se precisar fazer alterações em qualquer seção, selecione Edit (Editar). Quando estiver satisfeito com as configurações, escolha Create mesh (Criar malha).

A tela Status mostra todos os recursos de malha que foram criados. Você pode ver no console os recursos criados selecionando View mesh (Exibir malha).

AWS CLI

Revise as configurações da malha criada com o comando describe-mesh.

```
aws appmesh describe-mesh --mesh-name apps
```

Revise as configurações do serviço virtual que você criou com o describe-virtual-servicecomando.

```
aws appmesh describe-virtual-service --mesh-name apps --virtual-service-name
 serviceb.apps.local
```

Revise as configurações do nó virtual que você criou com o describe-virtual-nodecomando.

```
aws appmesh describe-virtual-node --mesh-name apps --virtual-node-name serviceB
```

Revise as configurações do roteador virtual que você criou com o describe-virtual-routercomando.

```
aws appmesh describe-virtual-router --mesh-name apps --virtual-router-name serviceB
```

Revise as configurações da rota criada com o comando describe-route.

```
aws appmesh describe-route --mesh-name apps \
    --virtual-router-name serviceB  --route-name serviceB
```

Etapa 5: Criar recursos adicionais

Para concluir o cenário, é necessário:

- Criar um nó virtual chamado `serviceBv2` e outro chamado `serviceA`. Ambos os nós virtuais escutam solicitações por meio da porta 80 do HTTP/2. Para o nó virtual `serviceA`, configure um back-end do `serviceb.apps.local`. Todo o tráfego de saída do nó virtual `serviceA` é enviado para o serviço virtual chamado `serviceb.apps.local`. Embora não seja abordado neste guia, também é possível especificar um caminho de arquivo para gravar logs de acesso para um nó virtual.
- Crie um serviço virtual adicional chamado `servicea.apps.local`, que enviará todo o tráfego diretamente para o nó virtual do `serviceA`.
- Atualizar a rota do `serviceB` criada em uma etapa anterior para enviar 75% de seu tráfego para o nó virtual do `serviceB` e 25% de seu tráfego para o nó virtual do `serviceBv2`. Com o passar do tempo, você poderá continuar a modificar os pesos até que o `serviceBv2` receba 100% do tráfego. Depois que todo o tráfego for enviado para o `serviceBv2`, você poderá descontinuar o nó virtual do `serviceB` e o serviço real. Conforme você altera os pesos, o código não exigirá nenhuma modificação, porque os nomes de serviço `serviceb.apps.local` virtual e real não são alterados. Lembre-se de que o serviço virtual `serviceb.apps.local` envia tráfego para o roteador virtual, que roteia o tráfego para os nós virtuais. Os nomes de descoberta de serviço para os nós virtuais podem ser alterados a qualquer momento.

AWS Management Console

1. No painel de navegação à esquerda, selecione Meshes (Malhas).
2. Selecione a malha `apps` criada em uma etapa anterior.
3. No painel de navegação esquerdo, selecione Virtual nodes (Nós virtuais).

4. Selecione Create nó virtual (Criar nó virtual).
5. Em Virtual node name (Nome do nó virtual), insira **`serviceBv2`**, em Service discovery method (Método de descoberta de serviço), escolha DNS e, em DNS hostname (Nome de host do DNS), insira **`servicebv2.apps.local`**.
6. Em Listener configuration (Configuração do Listener), selecione http2 para Protocol (Protocolo) e digite **`80`** para Port (Porta).
7. Selecione Create nó virtual (Criar nó virtual).
8. Selecione Create nó virtual (Criar nó virtual) novamente. Digite **`serviceA`** para o o Virtual node name (Nome do nó virtual). Em Service discovery method (Método de descoberta de serviços), escolha DNS e, para DNS hostname (Nome de host do DNS), insira **`servicea.apps.local`**.
9. Para Enter a virtual service name (Digite um nome de serviço virtual) em New backend (Novo back-end), digite **`serviceb.apps.local`**.
10. Em Listener configuration (Configuração do Listener), escolha http2 para Protocol (Protocolo), digite **`80`** para Port (Porta) e escolha Create virtual node (Criar nó virtual).
11. No painel de navegação esquerdo, selecione Virtual routers (Roteadores virtuais) e, depois, selecione o roteador virtual `serviceB` na lista.
12. Em Routes (Rotas), selecione a rota chamada `ServiceB` criada em uma etapa anterior e escolha Edit (Editar).
13. Em Targets (Destinos), Virtual node name (Nome do nó virtual), altere o valor de Weight (Peso) de `serviceB` para **`75`**.
14. Escolha Adicionar destino, depois escolha `serviceBv2` na lista suspensa e defina o valor de Peso como **`25`**.
15. Escolha Salvar.
16. No painel de navegação esquerdo, selecione Virtual services (Serviços virtuais) e escolha Create virtual service (Criar serviço virtual).
17. Insira **`servicea.apps.local`** para Virtual service name (Nome do serviço virtual), selecione Virtual node para Provider (Provedor), selecione `serviceA` para Virtual node (Nó virtual) e escolha Create virtual service (Criar serviço virtual).

AWS CLI

1. Crie o nó virtual `serviceBv2`.

a. Crie um arquivo denominado `create-virtual-node-servicebv2.json` com o conteúdo a seguir:

```
{
    "meshName": "apps",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http2"
                }
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceBv2.apps.local"
            }
        }
    },
    "virtualNodeName": "serviceBv2"
}
```

b. Crie o nó virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicebv2.json
```

2. Crie o nó virtual `serviceA`.

a. Crie um arquivo denominado `create-virtual-node-servicea.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "spec" : {
      "backends" : [
         {
            "virtualService" : {
               "virtualServiceName" : "serviceb.apps.local"
            }
         }
```

```
      ],
      "listeners" : [
         {
            "portMapping" : {
               "port" : 80,
               "protocol" : "http2"
            }
         }
      ],
      "serviceDiscovery" : {
         "dns" : {
            "hostname" : "servicea.apps.local"
         }
      }
   },
   "virtualNodeName" : "serviceA"
}
```

b. Crie o nó virtual.

```
aws appmesh create-virtual-node --cli-input-json file://create-virtual-node-servicea.json
```

3. Atualize o serviço virtual `serviceb.apps.local` criado em uma etapa anterior para enviar seu tráfego para o roteador virtual `serviceB`. Quando o serviço virtual foi criado originalmente, ele não enviava tráfego para nenhum lugar, já que o roteador virtual `serviceB` ainda não tinha sido criado.

a. Crie um arquivo denominado `update-virtual-service.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualRouter" : {
            "virtualRouterName" : "serviceB"
         }
      }
   },
   "virtualServiceName" : "serviceb.apps.local"
}
```

b. Atualize o serviço virtual com o update-virtual-servicecomando.

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

4. Atualize a rota `serviceB` criada em uma etapa anterior.

a. Crie um arquivo denominado `update-route.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "routeName" : "serviceB",
   "spec" : {
      "http2Route" : {
         "action" : {
            "weightedTargets" : [
               {
                  "virtualNode" : "serviceB",
                  "weight" : 75
               },
               {
                  "virtualNode" : "serviceBv2",
                  "weight" : 25
               }
            ]
         },
         "match" : {
            "prefix" : "/"
         }
      }
   },
   "virtualRouterName" : "serviceB"
}
```

b. Atualize a rota com o comando update-route.

```
aws appmesh update-route --cli-input-json file://update-route.json
```

5. Crie o serviço virtual `serviceA`.

a. Crie um arquivo denominado `create-virtual-servicea.json` com o conteúdo a seguir:

```
{
   "meshName" : "apps",
   "spec" : {
      "provider" : {
         "virtualNode" : {
            "virtualNodeName" : "serviceA"
         }
      }
   },
   "virtualServiceName" : "servicea.apps.local"
}
```

b. Crie o serviço virtual.

```
aws appmesh create-virtual-service --cli-input-json file://create-virtual-servicea.json
```

Resumo da malha

Antes de criar a malha de serviço, você tinha três serviços reais chamados `servicea.apps.local`, `serviceb.apps.local` e `servicebv2.apps.local`. Além dos serviços reais, agora você tem uma malha de serviços que contém os seguintes recursos que representam os serviços reais:

- Dois serviços virtuais. O proxy envia todo o tráfego do serviço virtual `servicea.apps.local` para o serviço virtual `serviceb.apps.local` por meio de um roteador virtual.
- Três nós virtuais chamados `serviceA`, `serviceB` e `serviceBv2`. O proxy Envoy usa as informações de descoberta de serviço configuradas para os nós virtuais para pesquisar os endereços IP dos serviços reais.
- Um roteador virtual com uma rota que instrui o proxy Envoy a rotear 75% do tráfego de entrada para o nó virtual `serviceB` e 25% do tráfego para o nó virtual `serviceBv2`.

Etapa 6: Atualizar os serviços

Depois de criar a malha, é necessário concluir as seguintes tarefas:

- Autorize o proxy Envoy implantado com cada serviço para ler a configuração de um ou mais nós virtuais. Para mais informações sobre como autorizar o proxy, consulte Autorização do Envoy Proxy.
- Para atualizar o serviço existente, conclua as etapas a seguir.

Como configurar uma instância do Amazon EC2 como um membro de nó virtual

1. Criar um perfil do IAM.

   a. Crie um arquivo denominado `ec2-trust-relationship.json` com o seguinte conteúdo:

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "ec2.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Crie um perfil do IAM com o comando a seguir.

   ```
   aws iam create-role --role-name mesh-virtual-node-service-b --assume-role-
   policy-document file://ec2-trust-relationship.json
   ```

2. Anexe políticas do IAM ao perfil que lhe permitam ler do Amazon ECR e apenas a configuração de um nó virtual do App Mesh específico.

   a. Crie um arquivo denominado `virtual-node-policy.json` com o conteúdo a seguir. `apps` é o nome da malha que você criou em the section called “Etapa 1: Criar uma malha e um serviço virtual” e `serviceB` é o nome do nó virtual que você criou em the section called “Etapa 2: Criar um nó virtual”. Substitua *111122223333* pela ID da sua conta e *us-west-2* pela Região na qual você criou sua malha.

   ```
   {
   ```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:StreamAggregatedResources",
            "Resource": [
                "arn:aws:appmesh:us-west-2:111122223333:mesh/apps/
virtualNode/serviceB"
            ]
        }
    ]
}
```

b. Crie a política com o comando a seguir.

```
aws iam create-policy --policy-name virtual-node-policy --policy-document
 file://virtual-node-policy.json
```

c. Anexe a política que você criou na etapa anterior ao perfil para que ele possa ler a configuração somente para o nó virtual de `serviceB` no App Mesh.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::111122223333:policy/
virtual-node-policy --role-name mesh-virtual-node-service-b
```

d. Anexe a política gerenciada `AmazonEC2ContainerRegistryReadOnly` ao perfil para que ele possa extrair a imagem do contêiner Envoy do Amazon ECR.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name mesh-virtual-node-service-b
```

3. Inicie uma instância do Amazon EC2 com o perfil do IAM que você criou.
4. Conecte-se à instância via SSH.
5. Instale o Docker e o AWS CLI na sua instância de acordo com a documentação do sistema operacional.
6. Autentique-se no repositório do Amazon ECR do Envoy na região da qual você deseja que seu cliente Docker obtenha a imagem.

   - Todas as regiões, exceto `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`. É possível substituir *us-west-2* por qualquer região compatível,

exceto `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`.

```
$aws ecr get-login-password \
    --region us-west-2 \
| docker login \
    --username AWS \
    --password-stdin 840364872350.dkr.ecr.us-west-2.amazonaws.com
```

- Região `me-south-1`

```
$aws ecr get-login-password \
    --region me-south-1 \
| docker login \
    --username AWS \
    --password-stdin 772975370895.dkr.ecr.me-south-1.amazonaws.com
```

- Região da `ap-east-1`

```
$aws ecr get-login-password \
    --region ap-east-1 \
| docker login \
    --username AWS \
    --password-stdin 856666278305.dkr.ecr.ap-east-1.amazonaws.com
```

7. Execute um dos seguintes comandos para iniciar o contêiner do App Mesh Envoy na sua instância, dependendo de qual Região você queira obter a imagem. Os valores de *apps* e de *serviceB* são os nomes da malha e do nó virtual definidos no cenário. Esses dados informam o proxy sobre a configuração de nó virtual a ser lida no App Mesh. Para concluir o cenário, também é necessário concluir essas etapas para as instâncias do EC2 que hospedam os serviços representados pelos nós virtuais `serviceBv2` e `serviceA`. Para os aplicativos, substitua esses valores pelos seus próprios.

   - Todas as regiões, exceto `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`. Você pode substituir o *Código de região* por qualquer região compatível, exceto as regiões `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`. Você pode substituir *1337* por qualquer valor entre `0` e `2147483647`.

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB  \
```

```
-u 1337 --network host 840364872350.dkr.ecr.region-code.amazonaws.com/aws-
appmesh-envoy:v1.27.3.0-prod
```

- Região `me-south-1`. Você pode substituir *1337* por qualquer valor entre `0` e `2147483647`.

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB  \
-u 1337 --network host 772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

- Região `ap-east-1`. Você pode substituir *1337* por qualquer valor entre `0` e `2147483647`.

```
sudo docker run --detach --env APPMESH_RESOURCE_ARN=mesh/apps/
virtualNode/serviceB  \
-u 1337 --network host 856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-
envoy:v1.27.3.0-prod
```

> **Note**
>
> A propriedade `APPMESH_RESOURCE_ARN` requer uma versão `1.15.0` ou posterior da imagem do Envoy. Para ter mais informações, consulte Envoy.

> **Important**
>
> Somente a versão v1.9.0.0-prod ou posterior é compatível para uso com o App Mesh.

8. Selecione `Show more` abaixo. Crie em sua instância um arquivo denominado `envoy-networking.sh` com os conteúdos a seguir. Substitua *8000* pela porta que o código do aplicativo usa para tráfego de entrada. Você pode alterar o valor para `APPMESH_IGNORE_UID`, mas o valor deve ser o mesmo que o valor especificado na etapa anterior, por exemplo, `1337`. Se necessário, você pode incluir endereços adicionais ao `APPMESH_EGRESS_IGNORED_IP`. Não modifique nenhuma outra linha.

```
#!/bin/bash -e

#
# Start of configurable options
#
```

```
#APPMESH_START_ENABLED="0"
APPMESH_IGNORE_UID="1337"
APPMESH_APP_PORTS="8000"
APPMESH_ENVOY_EGRESS_PORT="15001"
APPMESH_ENVOY_INGRESS_PORT="15000"
APPMESH_EGRESS_IGNORED_IP="169.254.169.254,169.254.170.2"

# Enable routing on the application start.
[ -z "$APPMESH_START_ENABLED" ] && APPMESH_START_ENABLED="0"

# Enable IPv6.
[ -z "$APPMESH_ENABLE_IPV6" ] && APPMESH_ENABLE_IPV6="0"

# Egress traffic from the processess owned by the following UID/GID will be
 ignored.
if [ -z "$APPMESH_IGNORE_UID" ] && [ -z "$APPMESH_IGNORE_GID" ]; then
    echo "Variables APPMESH_IGNORE_UID and/or APPMESH_IGNORE_GID must be set."
    echo "Envoy must run under those IDs to be able to properly route it's egress
 traffic."
    exit 1
fi

# Port numbers Application and Envoy are listening on.
if [ -z "$APPMESH_ENVOY_EGRESS_PORT" ]; then
    echo "APPMESH_ENVOY_EGRESS_PORT must be defined to forward traffic from the
 application to the proxy."
    exit 1
fi

# If an app port was specified, then we also need to enforce the proxies ingress
 port so we know where to forward traffic.
if [ ! -z "$APPMESH_APP_PORTS" ] && [ -z "$APPMESH_ENVOY_INGRESS_PORT" ]; then
    echo "APPMESH_ENVOY_INGRESS_PORT must be defined to forward traffic from the
 APPMESH_APP_PORTS to the proxy."
    exit 1
fi

# Comma separated list of ports for which egress traffic will be ignored, we always
 refuse to route SSH traffic.
if [ -z "$APPMESH_EGRESS_IGNORED_PORTS" ]; then
    APPMESH_EGRESS_IGNORED_PORTS="22"
else
```

```
    APPMESH_EGRESS_IGNORED_PORTS="$APPMESH_EGRESS_IGNORED_PORTS,22"
fi

#
# End of configurable options
#

function initialize() {
    echo "=== Initializing ==="
    if [ ! -z "$APPMESH_APP_PORTS" ]; then
        iptables -t nat -N APPMESH_INGRESS
        if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
            ip6tables -t nat -N APPMESH_INGRESS
        fi
    fi
    iptables -t nat -N APPMESH_EGRESS
    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        ip6tables -t nat -N APPMESH_EGRESS
    fi
}

function enable_egress_routing() {
    # Stuff to ignore
    [ ! -z "$APPMESH_IGNORE_UID" ] && \
        iptables -t nat -A APPMESH_EGRESS \
        -m owner --uid-owner $APPMESH_IGNORE_UID \
        -j RETURN

    [ ! -z "$APPMESH_IGNORE_GID" ] && \
        iptables -t nat -A APPMESH_EGRESS \
        -m owner --gid-owner $APPMESH_IGNORE_GID \
        -j RETURN

    [ ! -z "$APPMESH_EGRESS_IGNORED_PORTS" ] && \
        for IGNORED_PORT in $(echo "$APPMESH_EGRESS_IGNORED_PORTS" | tr "," "\n");
 do
          iptables -t nat -A APPMESH_EGRESS \
          -p tcp \
          -m multiport --dports "$IGNORED_PORT" \
          -j RETURN
        done

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
      # Stuff to ignore ipv6
```

```
    [ ! -z "$APPMESH_IGNORE_UID" ] && \
        ip6tables -t nat -A APPMESH_EGRESS \
        -m owner --uid-owner $APPMESH_IGNORE_UID \
        -j RETURN

    [ ! -z "$APPMESH_IGNORE_GID" ] && \
        ip6tables -t nat -A APPMESH_EGRESS \
        -m owner --gid-owner $APPMESH_IGNORE_GID \
        -j RETURN

    [ ! -z "$APPMESH_EGRESS_IGNORED_PORTS" ] && \
      for IGNORED_PORT in $(echo "$APPMESH_EGRESS_IGNORED_PORTS" | tr "," "\n");
do
        ip6tables -t nat -A APPMESH_EGRESS \
        -p tcp \
        -m multiport --dports "$IGNORED_PORT" \
        -j RETURN
      done
  fi

  # The list can contain both IPv4 and IPv6 addresses. We will loop over this
list
  # to add every IPv4 address into `iptables` and every IPv6 address into
`ip6tables`.
  [ ! -z "$APPMESH_EGRESS_IGNORED_IP" ] && \
      for IP_ADDR in $(echo "$APPMESH_EGRESS_IGNORED_IP" | tr "," "\n"); do
          if [[ $IP_ADDR =~ .*:.* ]]
          then
              [ "$APPMESH_ENABLE_IPV6" == "1" ] && \
                  ip6tables -t nat -A APPMESH_EGRESS \
                      -p tcp \
                      -d "$IP_ADDR" \
                      -j RETURN
          else
              iptables -t nat -A APPMESH_EGRESS \
                  -p tcp \
                  -d "$IP_ADDR" \
                  -j RETURN
          fi
      done

  # Redirect everything that is not ignored
  iptables -t nat -A APPMESH_EGRESS \
      -p tcp \
```

```
        -j REDIRECT --to $APPMESH_ENVOY_EGRESS_PORT

    # Apply APPMESH_EGRESS chain to non local traffic
    iptables -t nat -A OUTPUT \
        -p tcp \
        -m addrtype ! --dst-type LOCAL \
        -j APPMESH_EGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        # Redirect everything that is not ignored ipv6
        ip6tables -t nat -A APPMESH_EGRESS \
            -p tcp \
            -j REDIRECT --to $APPMESH_ENVOY_EGRESS_PORT
        # Apply APPMESH_EGRESS chain to non local traffic ipv6
        ip6tables -t nat -A OUTPUT \
            -p tcp \
            -m addrtype ! --dst-type LOCAL \
            -j APPMESH_EGRESS
    fi

}

function enable_ingress_redirect_routing() {
    # Route everything arriving at the application port to Envoy
    iptables -t nat -A APPMESH_INGRESS \
        -p tcp \
        -m multiport --dports "$APPMESH_APP_PORTS" \
        -j REDIRECT --to-port "$APPMESH_ENVOY_INGRESS_PORT"

    # Apply AppMesh ingress chain to everything non-local
    iptables -t nat -A PREROUTING \
        -p tcp \
        -m addrtype ! --src-type LOCAL \
        -j APPMESH_INGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        # Route everything arriving at the application port to Envoy ipv6
        ip6tables -t nat -A APPMESH_INGRESS \
            -p tcp \
            -m multiport --dports "$APPMESH_APP_PORTS" \
            -j REDIRECT --to-port "$APPMESH_ENVOY_INGRESS_PORT"

        # Apply AppMesh ingress chain to everything non-local ipv6
        ip6tables -t nat -A PREROUTING \
```

```
            -p tcp \
            -m addrtype ! --src-type LOCAL \
            -j APPMESH_INGRESS
    fi
}

function enable_routing() {
    echo "=== Enabling routing ==="
    enable_egress_routing
    if [ ! -z "$APPMESH_APP_PORTS" ]; then
        enable_ingress_redirect_routing
    fi
}

function disable_routing() {
    echo "=== Disabling routing ==="
    iptables -t nat -F APPMESH_INGRESS
    iptables -t nat -F APPMESH_EGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        ip6tables -t nat -F APPMESH_INGRESS
        ip6tables -t nat -F APPMESH_EGRESS
    fi
}

function dump_status() {
    echo "=== iptables FORWARD table ==="
    iptables -L -v -n
    echo "=== iptables NAT table ==="
    iptables -t nat -L -v -n

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        echo "=== ip6tables FORWARD table ==="
        ip6tables -L -v -n
        echo "=== ip6tables NAT table ==="
        ip6tables -t nat -L -v -n
    fi
}

function clean_up() {
    disable_routing
    ruleNum=$(iptables -L PREROUTING -t nat --line-numbers | grep APPMESH_INGRESS |
 cut -d " " -f 1)
    iptables -t nat -D PREROUTING $ruleNum
```

```
    ruleNum=$(iptables -L OUTPUT -t nat --line-numbers | grep APPMESH_EGRESS | cut
 -d " " -f 1)
    iptables -t nat -D OUTPUT $ruleNum

    iptables -t nat -X APPMESH_INGRESS
    iptables -t nat -X APPMESH_EGRESS

    if [ "$APPMESH_ENABLE_IPV6" == "1" ]; then
        ruleNum=$(ip6tables -L PREROUTING -t nat --line-numbers | grep
 APPMESH_INGRESS | cut -d " " -f 1)
        ip6tables -t nat -D PREROUTING $ruleNum

        ruleNum=$(ip6tables -L OUTPUT -t nat --line-numbers | grep APPMESH_EGRESS |
 cut -d " " -f 1)
        ip6tables -t nat -D OUTPUT $ruleNum

        ip6tables -t nat -X APPMESH_INGRESS
        ip6tables -t nat -X APPMESH_EGRESS
    fi
}

function main_loop() {
    echo "=== Entering main loop ==="
    while read -p '> ' cmd; do
        case "$cmd" in
            "quit")
                clean_up
                break
                ;;
            "status")
                dump_status
                ;;
            "enable")
                enable_routing
                ;;
            "disable")
                disable_routing
                ;;
            *)
                echo "Available commands: quit, status, enable, disable"
                ;;
        esac
    done
```

```
}

function print_config() {
    echo "=== Input configuration ==="
    env | grep APPMESH_ || true
}

print_config

initialize

if [ "$APPMESH_START_ENABLED" == "1" ]; then
    enable_routing
fi

main_loop
```

9. Para configurar regras de `iptables` para rotear o tráfego do aplicativo para o proxy Envoy, execute o script que você criou na etapa anterior.

```
sudo ./envoy-networking.sh
```

10. Inicie o código do aplicativo do nó virtual.

> **Note**
>
> Para ver mais exemplos e orientações sobre o App Mesh, consulte o repositório de exemplos do App Mesh.

Roteiro do App Mesh

Este é o roteiro público experimental para o AWS App Mesh. O roteiro permite que os clientes saibam sobre nossos próximos produtos e prioridades, o que os ajuda a planejar como usar o App Mesh no futuro. Este repositório contém informações sobre no que estamos trabalhando e permite que todos os clientes da AWS forneçam feedback direto.

Roteiro do App Mesh

Exemplos do App Mesh

Você pode encontrar tutoriais completos mostrando exemplos práticos e de código do AWS App Mesh para integração com vários serviços AWS no seguinte repositório:

Exemplos do App Mesh

Conceitos do App Mesh

App Mesh é composto pelos seguintes conceitos.

- Malhas de serviço
- Serviços virtuais
- Gateways virtuais
- Nós virtuais
- Roteadores virtuais

Malhas de serviço

Uma malha de serviços é um limite lógico para o tráfego de rede entre os serviços que residem nela. Depois de criar sua malha de serviços, você poderá criar serviços virtuais, nós virtuais, roteadores virtuais e rotas para distribuir o tráfego entre os aplicativos na sua malha.

Criação de uma malha de serviços

> **Note**
>
> Ao criar uma malha, você deve adicionar um seletor de namespace. Se o seletor de namespace estiver vazio, ele selecionará todos os namespaces. Para restringir os namespaces, use um rótulo para associar os recursos do App Mesh à malha criada.

AWS Management Console

Criação de uma nova malha de serviços usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Selecione Create mesh (Criar malha).
3. Em Mesh name (Nome da malha), especifique um nome para sua malha de serviços.
4. (Opcional) Escolha Permitir tráfego externo. Por padrão, os proxies na malha somente encaminham o tráfego entre si. Se permitir tráfego externo, os proxies na malha também encaminharão o tráfego TCP diretamente para serviços que não estão implantados com um proxy definido na malha.

> **Note**
>
> Se você especificar qualquer back-end em um nó virtual usando `ALLOW_ALL`, deverá especificar todas as saídas desse nó virtual como back-ends. Caso contrário, `ALLOW_ALL` não funcionará mais para esse nó virtual.

5. Preferência de versão do IP

Controle qual versão de IP deve ser usada para tráfego dentro da malha ativando Substituir comportamento da versão de IP padrão. Por padrão, o App Mesh usa uma variedade de versões de IP.

> **Note**
>
> A malha aplica a preferência de IP a todos os nós virtuais e gateways virtuais dentro de uma malha. Esse comportamento pode ser substituído em um nó virtual individual configurando a preferência de IP ao criar ou editar o nó. A preferência de IP não pode ser substituída em um gateway virtual porque a configuração dos gateways virtuais que permite a escuta do tráfego IPv4 e IPv6 é a mesma, independentemente da preferência definida na malha.

- Padrão
  - O resolvedor de DNS do Envoy prefere o `IPv6` e volta para o `IPv4`.
  - Usamos o endereço `IPv4` retornado pelo AWS Cloud Map, se disponível, e voltamos a usar o endereço `IPv6`.
  - O endpoint criado para o aplicativo local usa um endereço `IPv4`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4`.
- IPv6 preferencial
  - O resolvedor de DNS do Envoy prefere o `IPv6` e volta para o `IPv4`.
  - O endereço `IPv6` retornado pelo AWS Cloud Map é usado, se disponível, e volta a usar o endereço `IPv4`
  - O endpoint criado para o aplicativo local usa um endereço `IPv6`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.
- IPv4 preferencial

- O resolvedor de DNS do Envoy prefere o `IPv4` e volta para o `IPv6`.
- Usamos o endereço `IPv4` retornado pelo AWS Cloud Map, se disponível, e voltamos a usar o endereço `IPv6`.
- O endpoint criado para o aplicativo local usa um endereço `IPv4`.
- Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.

- Somente IPv6
  - O resolvedor de DNS do Envoy usa apenas `IPv6`.
  - Somente o endereço `IPv6` retornado pelo AWS Cloud Map é usado. Se o AWS Cloud Map retornar um endereço `IPv4`, nenhum endereço IP será usado e os resultados vazios serão retornados ao Envoy.
  - O endpoint criado para o aplicativo local usa um endereço `IPv6`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.
- Somente IPv4
  - O resolvedor de DNS do Envoy usa apenas `IPv4`.
  - Somente o endereço `IPv4` retornado pelo AWS Cloud Map é usado. Se o AWS Cloud Map retornar um endereço `IPv6`, nenhum endereço IP será usado e os resultados vazios serão retornados ao Envoy.
  - O endpoint criado para o aplicativo local usa um endereço `IPv4`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.

6. Selecione Create mesh (Criar malha) para concluir.
7. (Opcional) Compartilhe a malha com outras contas. Uma malha compartilhada permite que recursos criados por contas diferentes se comuniquem entre si na mesma malha. Para obter mais informações, consulte Como trabalhar com malhas compartilhadas.

AWS CLI

Criação de uma malha usando a AWS CLI.

Crie uma malha de serviços usando o seguinte comando (substitua os valores *vermelhos* pelos seus):

1.
```
aws appmesh create-mesh --mesh-name meshName
```
2. Resultado do exemplo:

```
{
    "mesh":{
        "meshName":"meshName",
        "metadata":{
            "arn":"arn:aws:appmesh:us-west-2:123456789012:mesh/meshName",
            "createdAt":"2022-04-06T08:45:50.072000-05:00",
            "lastUpdatedAt":"2022-04-06T08:45:50.072000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version":1
        },
        "spec":{},
        "status":{
            "status":"ACTIVE"
        }
    }
}
```

Para obter mais informações sobre como criar uma malha com a AWS CLI para o App Mesh, consulte o comando create-mesh na referência da AWS CLI.

Exclusão de uma malha

AWS Management Console

Para excluir um gateway virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha que deseja excluir. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Na caixa de confirmação, digite **`delete`** e clique em Excluir.

AWS CLI

Exclusão de uma malha usando a AWS CLI

1. Use o comando a seguir para excluir sua malha (substitua os valores *vermelhos* pelos seus):

```
aws appmesh delete-mesh \
     --mesh-name meshName
```

2. Resultado do exemplo:

```
{
    "mesh": {
        "meshName": "meshName",
        "metadata": {
            "arn":"arn:aws:appmesh:us-west-2:123456789012:mesh/meshName",
            "createdAt": "2022-04-06T08:45:50.072000-05:00",
            "lastUpdatedAt": "2022-04-07T11:06:32.795000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {},
        "status": {
            "status": "DELETED"
        }
    }
}
```

Para obter mais informações sobre como excluir uma malha com a AWS CLI para o App Mesh, consulte o comando delete-mesh na referência da AWS CLI.

Serviços virtuais

Um serviço virtual é uma abstração de um serviço real que é fornecido por um nó virtual direta ou indiretamente por meio de um roteador virtual. Os serviços dependentes chamam o serviço virtual pelo seu `virtualServiceName` e essas solicitações são roteadas para o nó virtual ou para o roteador virtual que é especificado como o provedor do serviço virtual.

Criar um serviço virtual

AWS Management Console

Para criar um serviço virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha na qual você deseja criar o serviço virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual services (Serviços virtuais) no painel de navegação à esquerda.
4. Selecione Create virtual service (Criar serviço virtual).
5. Em Virtual service name (Nome do serviço virtual), escolha um nome para o serviço virtual. Você pode escolher qualquer nome, mas o nome da descoberta de serviços real que você está almejando, como `my-service.default.svc.cluster.local`, é recomendado para facilitar a correlação de seus serviços virtuais com serviços reais. Dessa forma, não é necessário alterar o código para referenciar um nome diferente do referenciado atualmente no código. O nome que você especificar deve ser resolvido como um endereço IP não loopback, pois o contêiner do aplicativo deve ser capaz de resolver o nome com êxito antes que a solicitação seja enviada ao proxy do Envoy. Você pode usar qualquer endereço IP não loopback porque nem o aplicativo nem os contêineres de proxy se comunicam com esse endereço IP. O proxy se comunica com outros serviços virtuais por meio dos nomes que você configurou para eles no App Mesh, não por meio de endereços IP para os quais os nomes são resolvidos.
6. Em Provider (Provedor), escolha o tipo de provedor para o serviço virtual:
   - Se desejar que o serviço virtual distribua o tráfego entre vários nós virtuais, selecione Virtual router (Roteador virtual) e escolha o roteador virtual a ser usado no menu suspenso.
   - Se desejar que o serviço virtual acesse um nó virtual diretamente, sem um roteador virtual, selecione Nó virtual e escolha o nó virtual a ser usado no menu suspenso.

     > **Note**
     >
     > O App Mesh pode criar automaticamente uma política de novas tentativas de rota padrão do Envoy para cada provedor de nó virtual que você tenha definido em ou após 29 de julho de 2020, mesmo que você não possa definir essa política por

meio da API do App Mesh. Para obter mais informações, consulte Política padrão de tentativas de rotas.

- Se não desejar que o serviço virtual faça o roteamento do tráfego no momento (por exemplo, se os seus nós virtuais ou o roteador virtual ainda não existirem), selecione None (Nenhum). Você poderá atualizar o provedor desse serviço virtual mais tarde.

7. Selecione Create virtual service (Criar serviço virtual) para concluir.

AWS CLI

Para criar um serviço virtual usando a AWS CLI.

Crie um serviço virtual com um provedor de nó virtual usando o comando a seguir e um arquivo JSON de entrada (substitua os valores em *vermelho* pelos seus):

1.
```
aws appmesh create-virtual-service \
--cli-input-json file://create-virtual-service-virtual-node.json
```

2. Conteúdo do exemplo create-virtual-service-virtual-node.json:

```
{
    "meshName": "meshName",
    "spec": {
        "provider": {
            "virtualNode": {
                "virtualNodeName": "nodeName"
            }
        }
    },
    "virtualServiceName": "serviceA.svc.cluster.local"
}
```

3. Resultado do exemplo:

```
{
    "virtualService": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualService/serviceA.svc.cluster.local",
            "createdAt": "2022-04-06T09:45:35.890000-05:00",
```

```
            "lastUpdatedAt": "2022-04-06T09:45:35.890000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "provider": {
                "virtualNode": {
                    "virtualNodeName": "nodeName"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualServiceName": "serviceA.svc.cluster.local"
    }
}
```

Para obter mais informações sobre como criar um serviço virtual com a AWS CLI para App Mesh, consulte o comando create-virtual-service na referência da AWS CLI.

Excluir um serviço virtual

> **Note**
>
> Você não pode excluir um serviço virtual referenciado por uma rota do gateway. Primeiro, você precisa excluir a rota do gateway.

AWS Management Console

Para excluir um serviço virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha da qual você deseja excluir um serviço virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual services (Serviços virtuais) no painel de navegação à esquerda.

4. Escolha o serviço virtual que você deseja excluir e clique em Excluir no canto superior direito. Você só pode excluir um gateway virtual em que sua conta esteja listada como Proprietário do recurso.
5. Na caixa de confirmação, digite **`delete`** e clique em Excluir.

AWS CLI

Para excluir um serviço virtual usando a AWS CLI

1. Use o comando a seguir para excluir o serviço virtual (substitua os valores em *vermelho* pelos seus):

```
aws appmesh delete-virtual-service \
     --mesh-name meshName \
     --virtual-service-name serviceA.svc.cluster.local
```

2. Resultado do exemplo:

```
{
    "virtualService": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualService/serviceA.svc.cluster.local",
            "createdAt": "2022-04-06T09:45:35.890000-05:00",
            "lastUpdatedAt": "2022-04-07T10:39:42.772000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "provider": {
                "virtualNode": {
                    "virtualNodeName": "nodeName"
                }
            }
        },
        "status": {
            "status": "DELETED"
        },
```

```
        "virtualServiceName": "serviceA.svc.cluster.local"
    }
}
```

Para obter mais informações sobre como excluir um serviço virtual com a AWS CLI para App Mesh, consulte o comando delete-virtual-service na referência da AWS CLI.

Gateways virtuais

Um gateway virtual permite que recursos fora da sua malha se comuniquem com recursos que estão dentro da sua malha. O gateway virtual representa um proxy Envoy em execução em uma tarefa do serviço Amazon ECS, em um serviço do Kubernetes ou em uma instância do Amazon EC2. Ao contrário de um nó virtual, que representa um Envoy em execução com um aplicativo, um gateway virtual representa o Envoy implantado por si mesmo.

Os recursos externos devem ser capazes de resolver um nome do DNS para um endereço IP atribuído ao serviço ou instância que executa o Envoy. O Envoy pode então acessar toda a configuração do App Mesh para os recursos que estão dentro da malha. A configuração para lidar com as solicitações de entrada no gateway virtual é especificada usando Rotas do gateway.

> **Important**
>
> Um gateway virtual com um receptor HTTP ou HTTP2 regrava o nome do host da solicitação de entrada no Nome do serviço virtual de destino da Rota do gateway, e o prefixo correspondente da Rota do gateway é regravado para `/`, por padrão. Por exemplo, se você configurou o prefixo de correspondência da Rota do gateway como `/chapter`, se a solicitação de entrada for `/chapter/1`, a solicitação será regravada para `/1`. Para configurar regravações, consulte a seção Criar uma rota de gateway em Rotas do gateway. Ao criar um gateway virtual, `proxyConfiguration` e `user` não devem ser configurados.

Para obter uma explicação completa, consulte Configuração do gateway de entrada.

Criar um gateway virtual

> **Note**
>
> Ao criar um gateway virtual, você deve adicionar um Seletor de namespace com um rótulo para identificar a lista de namespaces aos quais associar Rotas do gateway ao Gateway virtual criado.

AWS Management Console

Para criar um gateway virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha na qual você deseja criar o gateway virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Escolha Gateways virtuais no painel de navegação à esquerda.
4. Escolha Criar gateway virtual.
5. Em Nome do gateway virtual, insira um nome para o seu gateway virtual.
6. (Opcional, mas recomendado) Configurar os Padrões de política de cliente.
   a. (Opcional) Selecione Impor o TLS se quiser exigir que o gateway só se comunique com os serviços virtuais usando Transport Layer Security (TLS).
   b. (Opcional) Para Portas, especifique uma ou mais portas nas quais você deseja impor o TLS na comunicação com serviços virtuais.
   c. Para Método de validação, selecione uma das opções a seguir. O certificado especificado já deve existir e atender aos requisitos específicos. Para obter mais informações, consulte Requisitos de certificado.
      - Hospedagem da AWS Private Certificate Authority: selecione um ou mais Certificados existentes.
      - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo para o Envoy buscar usando o Secret Discovery Service.
      - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados no sistema de arquivos em que o Envoy está implantado.

d. (Opcional) Insira um Nome alternativo do assunto. Para adicionar mais SANs, selecione Adicionar SAN. Os SANs devem estar formatados como FQDN ou URI.

e. (Opcional) Selecione Fornecer certificado de cliente e uma das opções abaixo para fornecer um certificado de cliente quando um servidor o solicitar e habilitar a autenticação de TLS mútuo. Para saber mais sobre o TLS mútuo, consulte os documentos de Autenticação de TLS mútuo do App Mesh.

- Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo para o Envoy buscar usando o Secret Discovery Service.
- Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados como a Chave privada, no sistema de arquivos em que o Envoy está implantado. Para uma explicação completa da implantação de uma malha com um aplicativo de exemplo usando criptografia com arquivos locais, consulte Configurar o TLS com Certificados TLS fornecidos por arquivo no GitHub.

7. (Opcional) Para configurar o registro em log, selecione Registro. Insira o caminho dos logs de acesso HTTP que você deseja que o Envoy use. Recomendamos o caminho `/dev/stdout`, para que você possa usar drivers de log do Docker para exportar seus logs do Envoy para um serviço, como o Amazon CloudWatch Logs.

> **Note**
>
> Os logs ainda devem ser ingeridos por um agente no seu aplicativo e enviados para um destino. Esse caminho de arquivo apenas informa ao Envoy para onde os logs devem ser enviados.

8. Configure o Receptor.

a. Selecione um Protocolo e especifique a Porta na qual o Envoy recebe o tráfego. O receptor http permite a transição da conexão para websockets. Você pode clicar em Adicionar receptor para adicionar vários receptores. O botão Remover removerá esse receptor.

b. (Opcional) Habilitar grupo de conexões

O pooling de conexões limita o número de conexões que o Envoy do gateway virtual pode estabelecer simultaneamente. O objetivo é proteger sua instância do Envoy de ficar sobrecarregada com conexões e permitir que você ajuste a modelagem de tráfego de acordo com as necessidades dos seus aplicativos.

Você pode definir as configurações do grupo de conexões do lado do destino para um receptor de gateway virtual. O App Mesh define as configurações do grupo de conexões do lado do cliente como infinitas por padrão, simplificando a configuração da malha.

> **Note**
>
> Os protocolos portMapping `connectionPool` e `connectionPool` devem ser os mesmos. Se o protocolo do seu receptor for `grpc` ou `http2`, especifique `maxRequests` somente. Se o protocolo do seu receptor for `http`, você poderá especificar ambos, `maxConnections` e `maxPendingRequests`.

- Em Máximo de conexões, especifique o número máximo de conexões de saída.
- Em Máximo de solicitações, especifique o número máximo de solicitações paralelas que podem ser estabelecidas com o Envoy do gateway virtual.
- (Opcional) Para Máximo de solicitações pendentes, especifique o número de solicitações excedentes após o Máximo de conexões que um Envoy coloca na fila. O valor padrão é `2147483647`.

c. (Opcional) Se você quiser configurar uma verificação de integridade para seu receptor, selecione Habilitar a verificação de integridade.

Uma política de verificação de integridade é opcional, mas se você especificar qualquer valor para uma política de integridade, deverá especificar valores para Limite de integridade, Intervalo de verificação de integridade, Protocolo de verificação de integridade, Tempo limite e Limite de integridade inadequada.

- Em Protocolo de verificação de integridade, escolha um protocolo. Se você selecionar grpc, o serviço deverá estar em conformidade com o Protocolo de verificação de integridade GRPC.
- Em Health check port (Porta de verificação de integridade), especifique a porta em que a verificação de integridade deve ser executada.
- Em Healthy threshold (Limite de integridade), especifique o número de verificações de integridade consecutivas bem-sucedidas que deve ocorrer antes de o listener ser declarado íntegro.
- Em Health check interval (Intervalo de verificação de integridade), especifique o período em milissegundos entre cada execução de verificação de integridade.

- Em Path (Caminho), especifique o caminho de destino para a solicitação de verificação de integridade. Esse valor é usado somente se o Protocolo de verificação de integridade for `http` ou `http2`. O valor é ignorado para outros protocolos.
- Em Período de tempo limite, especifique tempo de espera ao receber uma resposta da verificação de integridade, em milissegundos.
- Em Unhealthy threshold (Limite de não integridade), especifique o número de verificações de integridade consecutivas com falha que deve ocorrer antes de o listener ser declarado não íntegro.

d. (Opcional) Se você quiser especificar se os clientes se comunicam com esse gateway virtual usando TLS, selecione Habilitar encerramento do TLS.

- Em Modo, selecione o modo no qual você deseja que o TLS seja configurado no receptor.
- Em Método de certificado, selecione uma das seguintes opções. O certificado deve atender aos requisitos específicos. Para obter mais informações, consulte Requisitos de certificado.
  - Hospedagem do AWS Certificate Manager: selecione um Certificado existente.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo para o Envoy buscar usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para os arquivos da Cadeia de certificados e da Chave privada no sistema de arquivos em que o Envoy está implantado.
- (Opcional) Selecione Exigir certificado de cliente e uma das opções abaixo para habilitar a autenticação de TLS mútuo quando um cliente fornecer um certificado. Para saber mais sobre o TLS mútuo, consulte os documentos de Autenticação de TLS mútuo do App Mesh.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo para o Envoy buscar usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados no sistema de arquivos em que o Envoy está implantado.
- (Opcional) Insira um Nome alternativo do assunto. Para adicionar mais SANs, selecione Adicionar SAN. Os SANs devem estar formatados como FQDN ou URI.

9. Escolha Criar gateway virtual para concluir.

AWS CLI

Para criar um gateway virtual usando a AWS CLI.

Crie um gateway virtual usando o comando a seguir e um arquivo JSON de entrada (substitua os valores em *vermelho* pelos seus):

1.
```
aws appmesh create-virtual-gateway \
--mesh-name meshName \
--virtual-gateway-name virtualGatewayName \
--cli-input-json file://create-virtual-gateway.json
```

2. Conteúdo do exemplo create-virtual-gateway.json:

```
{
    "spec": {
      "listeners": [
        {
          "portMapping": {
            "port": 9080,
            "protocol": "http"
          }
        }
      ]
    }
}
```

3. Resultado do exemplo:

```
{
    "virtualGateway": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/
virtualGateway/virtualGatewayName",
            "createdAt": "2022-04-06T10:42:42.015000-05:00",
            "lastUpdatedAt": "2022-04-06T10:42:42.015000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
```

```
            "listeners": [
                {
                    "portMapping": {
                        "port": 9080,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualGatewayName": "virtualGatewayName"
    }
}
```

Para obter mais informações sobre como criar um gateway virtual com a AWS CLI para App Mesh, consulte o comando create-virtual-gateway na referência da AWS CLI.

Implementar um gateway virtual

Implante um serviço Amazon ECS ou Kubernetes que contenha somente o contêiner do Envoy. Você também pode implantar o contêiner Envoy em uma instância do Amazon EC2. Para obter mais informações, consulte Conceitos básicos do App Mesh e do Amazon EC2. Para obter mais informações sobre como implantar no Amazon ECS, consulte Conceitos básicos do App Mesh e do Amazon ECS ou Conceitos básicos do AWS App Mesh e do Kubernetes para implantar no Kubernetes. Você precisa definir a variável de ambiente `APPMESH_RESOURCE_ARN` como `mesh/`*`mesh-name`*`/virtualGateway/`*`virtual-gateway-name`* e não especificar a configuração do proxy para que o tráfego do proxy não seja redirecionado para ele mesmo. Por padrão, o App Mesh usa o nome do recurso especificado em `APPMESH_RESOURCE_ARN` quando o Envoy está se referindo a si mesmo em métricas e rastreamentos. É possível substituir esse comportamento definindo a variável de ambiente `APPMESH_RESOURCE_CLUSTER` com seu próprio nome.

Recomendamos implantar várias instâncias do contêiner e configurar um Network Load Balancer para balancear a carga do tráfego para as instâncias. O nome da descoberta de serviços do balanceador de carga é o nome que você deseja que os serviços externos usem para acessar os recursos que estão na malha, como *`myapp.example.com`*. Para obter mais informações, consulte Criação de um Network Load Balancer (Amazon ECS), Criação de um Load Balancer

externo (Kubernetes) ou Tutorial: Aumente a disponibilidade do seu aplicativo no Amazon EC2. Você também pode encontrar mais exemplos e orientações em nossos Exemplos do App Mesh.

Habilite a autorização de proxy para o Envoy. Para obter mais informações, consulte Autorização do Envoy Proxy.

Excluir um gateway virtual

AWS Management Console

Para excluir um gateway virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha da qual você deseja excluir um gateway virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Escolha Gateways virtuais no painel de navegação à esquerda.
4. Escolha o gateway virtual que deseja excluir e selecione Excluir. Você não pode excluir um gateway virtual se ele alguma rota do gateway associada. Você deve primeiro excluir todas as rotas do gateway associadas. Você só pode excluir um gateway virtual em que sua conta esteja listada como Proprietário do recurso.
5. Na caixa de confirmação, digite **`delete`** e selecione Excluir.

AWS CLI

Para excluir um gateway virtual usando a AWS CLI

1. Use o comando a seguir para excluir o gateway virtual (substitua os valores em *vermelho* pelos seus):

```
aws appmesh delete-virtual-gateway \
     --mesh-name meshName \
     --virtual-gateway-name virtualGatewayName
```

2. Resultado do exemplo:

```
{
    "virtualGateway": {
        "meshName": "meshName",
        "metadata": {
```

```
            "arn": "arn:aws:appmesh:us-west-2:123456789012:mesh/meshName/
virtualGateway/virtualGatewayName",
            "createdAt": "2022-04-06T10:42:42.015000-05:00",
            "lastUpdatedAt": "2022-04-07T10:57:22.638000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "123456789012",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 9080,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "DELETED"
        },
        "virtualGatewayName": "virtualGatewayName"
    }
}
```

Para obter mais informações sobre como excluir um gateway virtual com a AWS CLI para App Mesh, consulte o comando delete-virtual-gateway na referência da AWS CLI.

Rotas de gateway

Uma rota de gateway é anexada a um gateway virtual e roteia o tráfego para um serviço virtual existente. Se uma rota corresponder a uma solicitação, ela poderá distribuir o tráfego para um serviço virtual de destino. Este tópico ajuda você a trabalhar com rotas de gateway em uma malha de serviços.

Como criar uma rota de gateway

AWS Management Console

Para criar uma rota de gateway usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Selecione a malha na qual você deseja criar a rota de gateway. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Escolha Gateways virtuais no painel de navegação à esquerda.
4. Selecione o gateway virtual ao qual você deseja associar uma nova rota de gateway. Se não houver nenhum, você precisará criar um gateway virtual primeiro. Você só pode criar uma rota de gateway para um gateway virtual cuja conta esteja listada como Proprietária do recurso.
5. Na tabela de Rotas de gateway, selecione Criar rota de gateway.
6. Em Nome da rota do gateway, especifique o nome a ser usado para a rota.
7. Para o Tipo de rota de gateway, selecione http, http2 ou grpc.
8. Selecione um nome de serviço virtual existente. Se não houver nenhum, você precisará criar um serviço virtual primeiro.
9. Selecione a porta que corresponde ao destino em Porta do provedor de serviços virtuais. A porta do provedor de serviços virtuais é necessária quando o provedor (roteador ou nó) do serviço virtual selecionado tem vários receptores.
10. (Opcional) Em Prioridade, especifique a prioridade dessa rota de gateway.
11. Para a configuração da Correspondência, especifique:
    - Se http/http2 for o tipo selecionado:
      - (Opcional) Método: especifica o cabeçalho do método a ser correspondido nas solicitações de entrada http/http2.
      - (Opcional) Correspondência de porta: corresponde a porta para o tráfego de entrada. A correspondência de porta é obrigatória se esse gateway virtual tiver vários receptores.
      - (Opcional) Nome de host exato/com sufixo: especifica o nome do host que deve ser correspondente na solicitação recebida para rotear para o serviço virtual de destino.
      - (Opcional) Caminho Prefixo/Exato/Regex: o método de correspondência do caminho do URL.

- Correspondência de prefixo: uma solicitação correspondente por uma rota de gateway é regravada no nome do serviço virtual de destino e o prefixo correspondente é regravado, por padrão como `/`. Dependendo de como você configura seu serviço virtual, ele pode usar um roteador virtual para rotear a solicitação para diferentes nós virtuais, com base em prefixos ou cabeçalhos específicos.

  > **Important**
  >
  > - Você não pode especificar nenhum `/aws-appmesh*` ou `/aws-app-mesh*` para a Correspondência de prefixo. Esses prefixos são reservados para uso interno futuro do App Mesh.
  > - Se várias rotas de gateway forem definidas, uma solicitação corresponderá à rota com o prefixo mais longo. Por exemplo, se existissem duas rotas de gateway, uma com prefixo de `/chapter` e outra com prefixo de `/`, uma solicitação de `www.example.com/chapter/` corresponderia à rota de gateway com o prefixo `/chapter`.

  > **Note**
  >
  > Se você habilitar a correspondência baseada no Caminho/Prefixo, o App Mesh habilita a normalização do caminho (normalize_path e merge_slashes) para minimizar a probabilidade de vulnerabilidades de confusão de caminhos. As vulnerabilidades de confusão de caminhos ocorrem quando as partes que participam da solicitação usam representações de caminho diferentes.

- Correspondência exata: o parâmetro exato desativa a correspondência parcial de uma rota e garante que ela só retorne a rota se o caminho for uma correspondência EXATA com a URL atual.
- Correspondência Regex: usado para descrever padrões em que vários URLs podem realmente identificar uma única página no site.

- (Opcional) Parâmetros de consulta: esse campo permite que você faça a correspondência com os parâmetros da consulta.
- (Opcional) Cabeçalhos: especifica os cabeçalhos para http e http2. Ele deve corresponder à solicitação de entrada para rotear para o serviço virtual de destino.

- Se grpc for o tipo selecionado:

- Tipo de correspondência de nome de host e (opcional) Correspondência exata/de sufixo: especifica o nome do host que deve ser correspondente na solicitação de entrada para rotear para o serviço virtual de destino.
- Nome do serviço grpc: o serviço grpc atua como uma API para seu aplicativo e é definido com ProtoBuf.

> **Important**
>
> Você não pode especificar `/aws.app-mesh*` ou `aws.appmesh` para o Nome do serviço. Esses nomes de serviço são reservados para uso interno futuro do App Mesh.

- (Opcional) Metadados: especifica os metadados para grpc. Deve corresponder à solicitação recebida para rotear para o serviço virtual de destino.

12. (Opcional) Para regravar a configuração:

- Se http/http2 for o tipo selecionado:
  - Se Prefixo for o tipo de correspondência selecionado:
    - Substituir regravação automática do nome do host: por padrão, o nome do host é regravado no nome do serviço virtual de destino.
    - Substituir a regravação automática do prefixo: quando ativada, a reescrita do prefixo especifica o valor do prefixo regravado.
  - Se Caminho exato for o tipo de correspondência selecionado:
    - Substituir regravação automática do nome do host: por padrão, o nome do host é regravado no nome do serviço virtual de destino.
    - Regravação do caminho: especifica o valor do caminho regravado. Sem caminho padrão.
  - Se Caminho Regex for o tipo de correspondência selecionado:
    - Substituir regravação automática do nome do host: por padrão, o nome do host é regravado no nome do serviço virtual de destino.
    - Regravação do caminho: especifica o valor do caminho regravado. Sem caminho padrão.
- Se grpc for o tipo selecionado:

- Substituir regravação automática do nome do host: por padrão, o nome do host é regravado no nome do serviço virtual de destino.

13. Selecione Criar rota de gateway para concluir.

AWS CLI

Para criar uma rota de gateway usando a AWS CLI.

Crie uma rota de gateway usando o seguinte comando e insira JSON (substitua os valores em *vermelho* pelos seus):

1.
```
aws appmesh create-virtual-gateway \
--mesh-name meshName \
--virtual-gateway-name virtualGatewayName \
--gateway-route-name gatewayRouteName \
--cli-input-json file://create-gateway-route.json
```

2. Conteúdo do exemplo create-gateway-route.json:

```
{
    "spec": {
        "httpRoute" : {
            "match" : {
                "prefix" : "/"
            },
            "action" : {
                "target" : {
                    "virtualService": {
                        "virtualServiceName": "serviceA.svc.cluster.local"
                    }
                }
            }
        }
    }
}
```

3. Resultado do exemplo:

```
{
    "gatewayRoute": {
        "gatewayRouteName": "gatewayRouteName",
```

```
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualGateway/virtualGatewayName/gatewayRoute/gatewayRouteName",
            "createdAt": "2022-04-06T11:05:32.100000-05:00",
            "lastUpdatedAt": "2022-04-06T11:05:32.100000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "httpRoute": {
                "action": {
                    "target": {
                        "virtualService": {
                            "virtualServiceName": "serviceA.svc.cluster.local"
                        }
                    }
                },
                "match": {
                    "prefix": "/"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualGatewayName": "gatewayName"
    }
}
```

Para mais informações sobre como criar uma rota de gateway com o AWS CLI para App Mesh, consulte o comando create-gateway-route na referência AWS CLI.

Como excluir uma rota de gateway

AWS Management Console

Para excluir uma rota do gateway usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Selecione a malha da qual você deseja excluir uma rota de gateway. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Escolha Gateways virtuais no painel de navegação à esquerda.
4. Selecione o gateway virtual do qual você deseja excluir uma rota de gateway.
5. Na tabela de rotas do gateway, selecione a rota do gateway que você deseja excluir e selecione Excluir. Você só pode excluir uma rota de gateway se sua conta estiver listada como Proprietária do recurso.
6. Na caixa de confirmação, digite **`delete`** e clique em Excluir.

AWS CLI

Para excluir uma rota do gateway usando a AWS CLI

1. Use o comando a seguir para excluir sua rota de gateway (substitua os valores em *`vermelho`* pelos seus próprios):

```
aws appmesh delete-gateway-route \
     --mesh-name meshName \
     --virtual-gateway-name virtualGatewayName \
     --gateway-route-name gatewayRouteName
```

2. Resultado do exemplo:

```
{
    "gatewayRoute": {
        "gatewayRouteName": "gatewayRouteName",
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualGateway/virtualGatewayName/gatewayRoute/gatewayRouteName",
            "createdAt": "2022-04-06T11:05:32.100000-05:00",
            "lastUpdatedAt": "2022-04-07T10:36:33.191000-05:00",
```

```
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "httpRoute": {
                "action": {
                    "target": {
                        "virtualService": {
                            "virtualServiceName": "serviceA.svc.cluster.local"
                        }
                    }
                },
                "match": {
                    "prefix": "/"
                }
            }
        },
        "status": {
            "status": "DELETED"
        },
        "virtualGatewayName": "virtualGatewayName"
    }
}
```

Para mais informações sobre como criar uma rota de gateway com a AWS CLI para App Mesh, consulte o comando create-gateway-route na referência da AWS CLI.

Nós virtuais

Um nó virtual atua como um ponteiro lógico para um grupo de tarefas específico, como um serviço do Amazon ECS ou uma implantação Kubernetes. Ao criar um nó virtual, você deve especificar um método de descoberta de serviços para seu grupo de tarefas. Qualquer tráfego de entrada que seu nó virtual esperar deverá ser especificado como um receptor. Qualquer serviço virtual para o qual um nó virtual envia tráfego de saída é especificado como back-end.

Os metadados de resposta para o novo nó virtual contêm o nome do recurso da Amazon (ARN) associado ao nó virtual. Defina esse valor como a variável de ambiente `APPMESH_RESOURCE_ARN` para o contêiner do proxy Envoy do seu grupo de tarefas em

sua definição de tarefa do Amazon ECS ou na especificação de pods do Kubernetes. Por exemplo, o valor pode ser `arn:aws:appmesh:us-west-2:111122223333:mesh/myMesh/virtualNode/myVirtualNode`. Depois, isso é mapeado para os parâmetros do Envoy `node.id` e `node.cluster`. É necessário usar `1.15.0` ou posterior da imagem do Envoy quando definir essa variável. Para obter mais informações sobre variáveis Envoy do App Mesh, consulte Envoy.

> **Note**
>
> Por padrão, o App Mesh usa o nome do recurso especificado em `APPMESH_RESOURCE_ARN` quando o Envoy está se referindo a si mesmo em métricas e rastreamentos.
> É possível substituir esse comportamento definindo a variável de ambiente `APPMESH_RESOURCE_CLUSTER` com seu próprio nome.

Criar um nó virtual

AWS Management Console

Para criar um nó virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha em que deseja criar o nó virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual nodes (Nós virtuais) no painel de navegação à esquerda.
4. Escolha Criar nó virtual e especifique as configurações para o seu nó virtual.
5. Em Nome do nó virtual, escolha um nome para seu nó virtual.
6. Em Método de descoberta de serviços, escolha uma das seguintes opções:

   - DNS: especifique o Nome de host DNS do serviço real que o nó virtual representa. O proxy Envoy é implantado em uma Amazon VPC. O proxy envia solicitações de resolução de nomes para o servidor DNS configurado para a VPC. Se o nome do host for resolvido, o servidor DNS retornará um ou mais endereços IP. Para obter mais informações sobre as configurações de DNS para uma VPC, consulte Usando DNS com sua VPC. Para o tipo de resposta DNS (opcional), especifique os tipos de endpoints retornados pelo resolvedor de DNS. Balanceador de carga significa que o resolvedor de DNS retorna um conjunto de endpoints balanceados. Endpoints significa que o resolvedor de DNS está retornando

todos os endpoints. Por padrão, presume-se que o tipo de resposta seja Balanceador de carga.

> **Note**
>
> Se você usar o Route53, precisará usar o Balanceador de carga.

- AWS Cloud Map: especifique um Nome do serviço e um Namespace HTTP existentes. Opcionalmente, você também pode especificar os atributos que o App Mesh pode consultar no AWS Cloud Map selecionando Adicionar linha e especificando uma Chave e um Valor. Somente as instâncias que correspondam a todos os pares de chave/valor serão retornadas. Para usar o AWS Cloud Map, sua conta deve ter a função vinculada ao serviço `AWSServiceRoleForAppMesh`. Para ter mais informações sobre o AWS Cloud Map, consulte o Guia do desenvolvedor do AWS Cloud Map.
- Nenhum: selecione se o nó virtual não esperar nenhum tráfego de entrada.

7. Preferência de versão do IP

Controle qual versão de IP deve ser usada para tráfego dentro da malha ativando Substituir comportamento da versão de IP padrão. Por padrão, o App Mesh usa uma variedade de versões de IP.

> **Note**
>
> A definição da preferência de IP em um nó virtual substitui apenas a preferência de IP definida para a malha nesse nó específico.

- Padrão
  - O resolvedor de DNS do Envoy prefere o `IPv6` e volta para o `IPv4`.
  - Usamos o endereço `IPv4` retornado pelo AWS Cloud Map, se disponível, e voltamos a usar o endereço `IPv6`.
  - O endpoint criado para o aplicativo local usa um endereço `IPv4`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4`.
- IPv6 preferencial
  - O resolvedor de DNS do Envoy prefere o `IPv6` e volta para o `IPv4`.

- O endereço `IPv6` retornado pelo AWS Cloud Map é usado, se disponível, e volta a usar o endereço `IPv4`
- O endpoint criado para o aplicativo local usa um endereço `IPv6`.
- Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.

- IPv4 preferencial
  - O resolvedor de DNS do Envoy prefere o `IPv4` e volta para o `IPv6`.
  - Usamos o endereço `IPv4` retornado pelo AWS Cloud Map, se disponível, e voltamos a usar o endereço `IPv6`.
  - O endpoint criado para o aplicativo local usa um endereço `IPv4`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.
- Somente IPv6
  - O resolvedor de DNS do Envoy usa apenas `IPv6`.
  - Somente o endereço `IPv6` retornado pelo AWS Cloud Map é usado. Se o AWS Cloud Map retornar um endereço `IPv4`, nenhum endereço IP será usado e os resultados vazios serão retornados ao Envoy.
  - O endpoint criado para o aplicativo local usa um endereço `IPv6`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.
- Somente IPv4
  - O resolvedor de DNS do Envoy usa apenas `IPv4`.
  - Somente o endereço `IPv4` retornado pelo AWS Cloud Map é usado. Se o AWS Cloud Map retornar um endereço `IPv6`, nenhum endereço IP será usado e os resultados vazios serão retornados ao Envoy.
  - O endpoint criado para o aplicativo local usa um endereço `IPv4`.
  - Os receptores do Envoy se vinculam a todos os endereços `IPv4` e `IPv6`.

8. (Opcional) Padrões da política do cliente: configure os requisitos padrão ao se comunicar com serviços virtuais de back-end.

> **Note**
>
> - Se você quiser habilitar o Transport Layer Security (TLS) para um nó virtual existente, recomendamos criar um novo nó virtual, que represente o mesmo serviço do nó virtual existente, para habilitar nele o TLS. Em seguida, transfira

gradualmente o tráfego para o novo nó virtual usando um roteador virtual e uma rota. Para obter mais informações sobre como criar uma rota e ajustar os pesos para a transição, consulte Rotas. Se você atualizar um nó virtual existente que serve tráfego com TLS, há uma chance de que os proxies Envoy do cliente downstream recebam o contexto de validação de TLS antes que o proxy Envoy do nó virtual que você atualizou receba o certificado. Isso pode causar erros de negociação de TLS nos proxies Envoy downstream.

- A autorização de proxy deve ser habilitada para o proxy Envoy implantado com o aplicativo representado pelos nós virtuais do serviço de back-end. Recomendamos que, ao habilitar a autorização de proxy, restrinja o acesso somente aos nós virtuais com os quais esse nó virtual está se comunicando.

- (Opcional) Selecione Impor o TLS se quiser exigir que o nó virtual se comunique com todos os back-ends usando o Transport Layer Security (TLS).
- (Opcional) Se você quiser exigir apenas o uso de TLS para uma ou mais portas específicas, insira um número em Portas. Para adicionar mais portas, selecione Adicionar porta. Se você não especificar nenhuma porta, o TLS será aplicado a todas as portas.
- Para Método de validação, selecione uma das opções a seguir. O certificado especificado já deve existir e atender aos requisitos específicos. Para obter mais informações, consulte Requisitos de certificado.
  - Hospedagem da AWS Private Certificate Authority: selecione um ou mais Certificados existentes. Para uma explicação completa da implantação de uma malha com um aplicativo de exemplo usando criptografia com um Certificado do ACM, consulte Configurar o TLS com o AWS Certificate Manager no GitHub.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo que o Envoy buscará usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados no sistema de arquivos em que o Envoy está implantado. Para uma explicação completa da implantação de uma malha com um aplicativo de exemplo usando criptografia com arquivos locais, consulte Configurar o TLS com Certificados TLS fornecidos por arquivo no GitHub.
- (Opcional) Insira um Nome alternativo do assunto. Para adicionar mais SANs, selecione Adicionar SAN. Os SANs devem estar formatados como FQDN ou URI.

- (Opcional) Selecione Fornecer certificado de cliente e uma das opções abaixo para fornecer um certificado de cliente quando um servidor o solicitar e habilitar a autenticação de TLS mútuo. Para saber mais sobre o TLS mútuo, consulte os documentos de Autenticação de TLS mútuo do App Mesh.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo que o Envoy buscará usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados como a Chave privada, no sistema de arquivos em que o Envoy está implantado.

9. (Opcional) Backends de serviço: especifique o serviço virtual do App Mesh com o qual o nó virtual se comunicará.

- Insira um nome do serviço virtual do App Mesh ou um nome do recurso da Amazon (ARN) completo para o serviço virtual com o qual o seu nó virtual se comunica.
- (Opcional) Se você quiser definir configurações de TLS exclusivas para um back-end, selecione Configurações de TLS e, em seguida, selecione Substituir padrões.
  - (Opcional) Selecione Impor o TLS se quiser exigir que o nó virtual se comunique com todos os back-ends usando TLS.
  - (Opcional) Se você quiser exigir apenas o uso de TLS para uma ou mais portas específicas, insira um número em Portas. Para adicionar mais portas, selecione Adicionar porta. Se você não especificar nenhuma porta, o TLS será aplicado a todas as portas.
  - Para Método de validação, selecione uma das opções a seguir. O certificado especificado já deve existir e atender aos requisitos específicos. Para obter mais informações, consulte Requisitos de certificado.
    - Hospedagem da AWS Private Certificate Authority: selecione um ou mais Certificados existentes.
    - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo que o Envoy buscará usando o Secret Discovery Service.
    - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados no sistema de arquivos em que o Envoy está implantado.
  - (Opcional) Insira um Nome alternativo do assunto. Para adicionar mais SANs, selecione Adicionar SAN. Os SANs devem estar formatados como FQDN ou URI.

- (Opcional) Selecione Fornecer certificado de cliente e uma das opções abaixo para fornecer um certificado de cliente quando um servidor o solicitar e habilitar a autenticação de TLS mútuo. Para saber mais sobre o TLS mútuo, consulte os documentos de Autenticação de TLS mútuo do App Mesh.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo que o Envoy buscará usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados como a Chave privada, no sistema de arquivos em que o Envoy está implantado.
- Para adicionar mais back-ends, selecione Adicionar back-end.

10. (Opcional) Registro

    Para configurar o registro em log, insira o caminho de logs de acesso HTTP que o Envoy deve usar. Recomendamos o caminho `/dev/stdout`, para que você possa usar drivers de log do Docker para exportar seus logs do Envoy para um serviço, como o Amazon CloudWatch Logs.

    > **Note**
    >
    > Os logs ainda devem ser ingeridos por um agente no seu aplicativo e enviados para um destino. Esse caminho de arquivo apenas informa ao Envoy para onde os logs devem ser enviados.

11. Configuração do receptor

    Os receptores são compatíveis com os protocolos `HTTP`, `HTTP/2`, `GRPC` e `TCP`. `HTTPS` não é suportado.

    a. Se o seu nó virtual esperar tráfego de entrada, especifique uma Porta e um Protocolo para o Receptor. O receptor http permite a transição da conexão para websockets. Você pode clicar em Adicionar receptor para adicionar vários receptores. O botão Remover removerá esse receptor.

    b. (Opcional) Habilitar grupo de conexões

       O pooling de conexões limita o número de conexões que um Envoy pode estabelecer simultaneamente com o cluster de aplicativos local. O objetivo é proteger seu aplicativo

local de ficar sobrecarregado com conexões e permitir que você ajuste a modelagem de tráfego de acordo com as necessidades dos seus aplicativos.

Você pode definir as configurações do grupo de conexões do lado do destino para um receptor de nó virtual. O App Mesh define as configurações do grupo de conexões do lado do cliente como infinitas por padrão, simplificando a configuração da malha.

> **Note**
>
> Os protocolos connectionPool e portMapping devem ser os mesmos. Se o protocolo do seu receptor for tcp, especifique somente maxConnections. Se o protocolo do seu receptor for grpc ou http2, especifique somente maxRequests. Se o protocolo do seu receptor for http, você poderá especificar maxConnections e maxPendingRequests.

- Em Máximo de conexões, especifique o número máximo de conexões de saída.
- (Opcional) Para Máximo de solicitações pendentes, especifique o número de solicitações excedentes após o Máximo de conexões que um Envoy colocará na fila. O valor padrão é `2147483647`.

c. (Opcional) Habilitar detecção de discrepância

A detecção de discrepâncias aplicada no Envoy do cliente permite que os clientes tomem medidas quase imediatas em conexões com falhas conhecidas e observadas. É uma forma de implementação de disjuntor que rastreia o estado de integridade de hosts individuais no serviço upstream.

A detecção de discrepâncias determina dinamicamente se os endpoints em um cluster upstream têm um desempenho diferente dos outros e os remove do conjunto de balanceamento de carga íntegro.

> **Note**
>
> Para configurar com eficácia a detecção discrepâncias para um nó virtual do servidor, o método de descoberta de serviços desse nó virtual pode ser o AWS Cloud Map ou o DNS com o campo do tipo de resposta definido como `ENDPOINTS`. Se você usar o método de descoberta de serviços por DNS com o tipo de resposta como `LOADBALANCER`, o proxy Envoy elegerá apenas um único

endereço IP para rotear para o serviço upstream. Isso anula o comportamento de detecção de discrepâncias de ejetar um host não íntegro de um conjunto de hosts. Consulte a seção Método de descoberta de serviços para obter mais detalhes sobre o comportamento do proxy Envoy em relação ao tipo de descoberta de serviços.

- Para Erros do servidor, especifique o número de erros 5xx consecutivos necessários para a ejeção.
- Para o Intervalo da detecção de discrepância, especifique o intervalo e a unidade de tempo entre a análise de varredura e a ejeção.
- Para Duração da ejeção básica, especificar o valor básico e a unidade de tempo no qual um host é ejetado.
- Para Porcentagem de ejeção, especificar a porcentagem máxima de hosts no grupo de balanceamento de carga que pode ser ejetado.

d. (Opcional) Ativar verificação de integridade: defina as configurações de uma política de verificação de integridade.

   Uma política de verificação de integridade é opcional, mas se você especificar qualquer valor para uma política de integridade, deverá especificar valores para Limite de integridade, Intervalo de verificação de integridade, Protocolo de verificação de integridade, Tempo limite e Limite de integridade inadequada.

   - Em Protocolo de verificação de integridade, escolha um protocolo. Se você selecionar grpc, o serviço deverá estar em conformidade com o Protocolo de verificação de integridade GRPC.
   - Em Health check port (Porta de verificação de integridade), especifique a porta em que a verificação de integridade deve ser executada.
   - Em Healthy threshold (Limite de integridade), especifique o número de verificações de integridade consecutivas bem-sucedidas que deve ocorrer antes de o listener ser declarado íntegro.
   - Em Health check interval (Intervalo de verificação de integridade), especifique o período em milissegundos entre cada execução de verificação de integridade.

- Em Path (Caminho), especifique o caminho de destino para a solicitação de verificação de integridade. Esse valor é usado somente se o Protocolo de verificação de integridade for `http` ou `http2`. O valor é ignorado para outros protocolos.
- Em Timeout period (Período de tempo limite), especifique tempo de espera ao receber uma resposta da verificação de integridade, em milissegundos.
- Em Unhealthy threshold (Limite de não integridade), especifique o número de verificações de integridade consecutivas com falha que deve ocorrer antes de o listener ser declarado não íntegro.

e. (Opcional) Habilitar encerramento do TLS: configure como outros nós virtuais se comunicam com esse nó virtual usando TLS.

- Em Modo, selecione o modo para o qual você deseja que o TLS seja configurado no receptor.
- Em Método de certificado, selecione uma das seguintes opções. O certificado deve atender aos requisitos específicos. Para obter mais informações, consulte Requisitos de certificado.
  - Hospedagem do AWS Certificate Manager: selecione um Certificado existente.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo que o Envoy buscará usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados como a Chave privada, no sistema de arquivos em que o proxy Envoy está implantado.
- (Opcional) Selecione Exigir certificados de cliente e uma das opções abaixo para habilitar a autenticação de TLS mútuo quando um cliente fornecer um certificado. Para saber mais sobre o TLS mútuo, consulte os documentos de Autenticação de TLS mútuo do App Mesh.
  - Envoy Secret Discovery Service (SDS) hosting: digite o nome do segredo que o Envoy buscará usando o Secret Discovery Service.
  - Hospedagem de arquivos local: especifique o caminho para o arquivo da Cadeia de certificados no sistema de arquivos em que o Envoy está implantado.
- (Opcional) Insira um Nome alternativo do assunto. Para adicionar mais SANs, selecione Adicionar SAN. Os SANs devem estar formatados como FQDN ou URI.

f. (Opcional) Tempos limite

> **Note**
>
> Se você especificar um tempo limite maior que o padrão, certifique-se de configurar um roteador virtual e uma rota com um tempo limite maior que o padrão. No entanto, se você diminuir o tempo limite para um valor menor que o padrão, é opcional atualizar os tempos limite em Rota. Para obter mais informações, consulte Rotas.

- Tempo limite da solicitação: você pode especificar um tempo limite da solicitação se tiver selecionado grpc, http ou http2 para o Protocolo do receptor. O padrão é 15 segundos. Um valor de `0` desabilita o tempo limite.
- Duração do tempo ocioso: você pode especificar uma duração do tempo ocioso para qualquer protocolo do receptor. O padrão é 300 segundos.

12. Escolha Criar nó virtual para concluir.

AWS CLI

Para criar um nó virtual usando a AWS CLI.

Crie um nó virtual que use DNS para descoberta de serviços usando o comando a seguir e um arquivo JSON de entrada (substitua os valores em *vermelho* pelos seus):

1.
```
aws appmesh create-virtual-node \
--cli-input-json file://create-virtual-node-dns.json
```

2. Conteúdo do exemplo create-virtual-node-dns.json:

```
{
    "meshName": "meshName",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http"
                }
```

```
            }
        ],
        "serviceDiscovery": {
            "dns": {
                "hostname": "serviceBv1.svc.cluster.local"
            }
        }
    },
    "virtualNodeName": "nodeName"
}
```

3. Resultado do exemplo:

```
{
    "virtualNode": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
            "createdAt": "2022-04-06T09:12:24.348000-05:00",
            "lastUpdatedAt": "2022-04-06T09:12:24.348000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ],
            "serviceDiscovery": {
                "dns": {
                    "hostname": "serviceBv1.svc.cluster.local"
                }
            }
        },
        "status": {
            "status": "ACTIVE"
        },
```

```
        "virtualNodeName": "nodeName"
    }
}
```

Para obter mais informações sobre como criar um nó virtual com a AWS CLI para App Mesh, consulte o comando create-virtual-node na referência da AWS CLI.

Excluir um nó virtual

> **Note**
>
> Você não pode excluir um nó virtual se ele for especificado como destino em qualquer rota ou como provedor em qualquer serviço virtual.

AWS Management Console

Para excluir um nó virtual usando a AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha da qual você deseja excluir um nó virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual nodes (Nós virtuais) no painel de navegação à esquerda.
4. Na tabela Nós virtuais, escolha o nó virtual que deseja excluir e selecione Excluir. Para excluir um nó virtual, o ID da sua conta deve estar listado nas colunas Proprietário da malha ou Proprietário do recurso do nó virtual.
5. Na caixa de confirmação, digite **`delete`** e selecione Excluir.

AWS CLI

Para excluir um nó virtual usando a AWS CLI

1. Use o comando a seguir para excluir o nó virtual (substitua os valores em *vermelho* pelos seus):

```
aws appmesh delete-virtual-node \
     --mesh-name meshName \
```

```
    --virtual-node-name nodeName
```

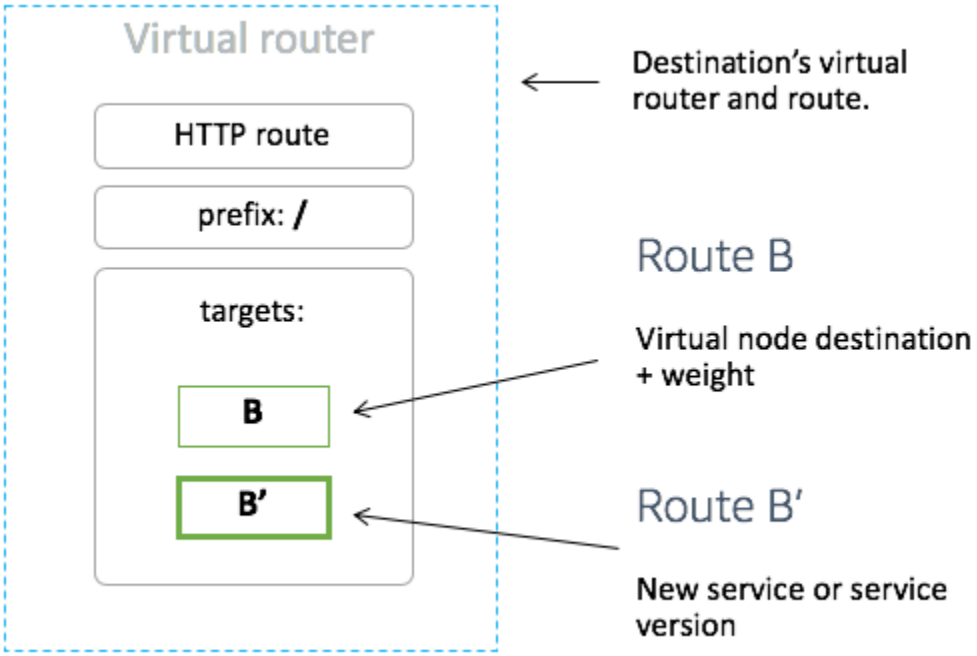
2. Resultado do exemplo:

```
{
    "virtualNode": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
            "createdAt": "2022-04-06T09:12:24.348000-05:00",
            "lastUpdatedAt": "2022-04-07T11:03:48.120000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
            "backends": [],
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ],
            "serviceDiscovery": {
                "dns": {
                    "hostname": "serviceBv1.svc.cluster.local"
                }
            }
        },
        "status": {
            "status": "DELETED"
        },
        "virtualNodeName": "nodeName"
    }
}
```

Para obter mais informações sobre como excluir um nó virtual com a AWS CLI para App Mesh, consulte o comando delete-virtual-node na referência da AWS CLI.

Roteadores virtuais

Os roteadores virtuais manipulam o tráfego de um ou mais serviços virtuais dentro da malha. Depois de criar um roteador virtual, você pode criar e associar rotas para seu roteador virtual que direcionem as solicitações de entrada para diferentes nós virtuais.



Qualquer tráfego de entrada que o roteador virtual esperar deverá ser especificado como um receptor.

Criar um roteador virtual

AWS Management Console

Para criar um roteador virtual usando o AWS Management Console

> **Note**
>
> Ao criar um roteador virtual, você deve adicionar um seletor de namespace com um rótulo para identificar a lista de namespaces para associar Rotas ao Roteador virtual criado.

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.

2. Escolha a malha em que deseja criar o roteador virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual routers (Roteadores virtuais) no painel de navegação à esquerda.
4. Selecione Create virtual router (Criar roteador virtual).
5. Em Virtual router name (Nome do roteador virtual), especifique um nome para seu roteador virtual. São permitidos até 255 letras, números, hifens e sublinhados.
6. (Opcional) Na configuração do Receptor, especifique uma Porta e um Protocolo para seu roteador virtual. O receptor `http` permite a transição da conexão para websockets. Você pode clicar em Adicionar receptor para adicionar vários receptores. O botão Remover removerá esse receptor.
7. Selecione Create virtual router (Criar roteador virtual) para concluir.

AWS CLI

Para criar um roteador virtual usando a AWS CLI.

Crie um roteador virtual usando o comando a seguir e um arquivo JSON de entrada (substitua os valores em *vermelho* pelos seus):

1.
```
aws appmesh create-virtual-router \
     --cli-input-json file://create-virtual-router.json
```
2. Conteúdo do exemplo create-virtual-router.json
3.
```
{
    "meshName": "meshName",
    "spec": {
        "listeners": [
            {
                "portMapping": {
                    "port": 80,
                    "protocol": "http"
                }
            }
        ]
    },
    "virtualRouterName": "routerName"
}
```
4. Resultado do exemplo:

```
{
    "virtualRouter": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName",
            "createdAt": "2022-04-06T11:49:47.216000-05:00",
            "lastUpdatedAt": "2022-04-06T11:49:47.216000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "spec": {
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "routerName"
    }
}
```

Para obter mais informações sobre como criar um roteador virtual com a AWS CLI para App Mesh, consulte o comando create-virtual-router na referência da AWS CLI.

Excluir um roteador virtual

> **Note**
>
> Você não pode excluir um roteador virtual se ele tiver alguma rota ou se estiver especificado como provedor de qualquer serviço virtual.

AWS Management Console

Para excluir um roteador virtual usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha da qual você deseja excluir um roteador virtual. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual routers (Roteadores virtuais) no painel de navegação à esquerda.
4. Na tabela Roteadores virtuais, escolha o roteador virtual que você deseja excluir e selecione Excluir no canto superior direito. Para excluir um roteador virtual, o ID da conta deve estar listado nas colunas Proprietário da malha ou Proprietário do recurso do roteador virtual.
5. Na caixa de confirmação, digite **`delete`** e clique em Excluir.

AWS CLI

Para excluir um roteador virtual usando a AWS CLI

1. Use o comando a seguir para excluir o roteador virtual (substitua os valores em *vermelho* pelos seus):

```
aws appmesh delete-virtual-router \
     --mesh-name meshName \
     --virtual-router-name routerName
```

2. Resultado do exemplo:

```
{
    "virtualRouter": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName",
            "createdAt": "2022-04-06T11:49:47.216000-05:00",
            "lastUpdatedAt": "2022-04-07T10:49:53.402000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
        },
        "spec": {
```

```
            "listeners": [
                {
                    "portMapping": {
                        "port": 80,
                        "protocol": "http"
                    }
                }
            ]
        },
        "status": {
            "status": "DELETED"
        },
        "virtualRouterName": "routerName"
    }
}
```

Para obter mais informações sobre como excluir um roteador virtual com a AWS CLI para App Mesh, consulte o comando delete-virtual-router na referência da AWS CLI.

Rotas

A rota está associada a um roteador virtual. A rota é usada para atender às solicitações do roteador virtual e distribuir o tráfego para os nós virtuais associados. Se uma rota corresponder a uma solicitação, ela poderá distribuir o tráfego para um ou mais nós virtuais de destino. Você pode especificar o peso relativo para cada nó virtual. Este tópico ajuda você a trabalhar com rotas em uma malha de serviços.

Criar uma rota

AWS Management Console

Para criar uma rota usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha na qual deseja criar a rota. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual routers (Roteadores virtuais) no painel de navegação à esquerda.
4. Escolha o roteador ao qual deseja associar uma nova rota. Se nenhum estiver listado, você precisará criar um roteador virtual primeiro.

5. Na tabela Routes (Rotas), selecione Create route (Criar rota). Para criar uma rota, o ID da conta deve estar listado como o Proprietário do recurso da rota.
6. Em Route name (Nome da rota), especifique o nome a ser usado para a rota.
7. Em Tipo de rota, escolha o protocolo que deseja para a rota. O protocolo que selecionar deve corresponder ao protocolo do receptor selecionado para o roteador virtual e o nó virtual para o qual você está roteando o tráfego.
8. (Opcional) Em Prioridade da rota, especifique uma prioridade de 0 a 1000 para usar em sua rota. A correspondência das rotas é feita com base em um valor especificado, e 0 é a prioridade mais alta.
9. (Opcional) Escolha Configuração adicional. Nos protocolos abaixo, escolha o protocolo que você selecionou para Tipo de rota e especifique as configurações no console conforme desejado.
10. Para Configuração de destino, selecione o nó virtual existente do App Mesh para direcionar o tráfego e especificar um Peso. Você pode escolher Adicionar destino para adicionar mais destinos. A porcentagem de todos os destinos deve somar 100. Se nenhum nó virtual estiver listado, você deverá criar um primeiro. Se o nó virtual selecionado tiver vários receptores, a Porta de destino será necessária.
11. Para a configuração da Correspondência, especifique:

    A configuração de correspondência não está disponível para *tcp*

    - Se http/http2 for o tipo selecionado:
      - (Opcional) Método: especifica o cabeçalho do método a ser correspondido nas solicitações de entrada http/http2.
      - (Opcional) Correspondência de porta: corresponde a porta para o tráfego de entrada. A correspondência de portas é necessária se esse roteador virtual tiver vários receptores.
      - (Opcional) Caminho Prefixo/Exato/Regex: método de correspondência do caminho do URL.
        - Correspondência de prefixo: uma solicitação correspondente por uma rota de gateway é regravada no nome do serviço virtual de destino e o prefixo correspondente é regravado, por padrão como `/`. Dependendo de como você configura seu serviço virtual, ele pode usar um roteador virtual para rotear a solicitação para diferentes nós virtuais, com base em prefixos ou cabeçalhos específicos.

> **Note**
>
> Se você habilitar a correspondência baseada no Caminho/Prefixo, o App Mesh habilita a normalização do caminho (normalize_path e merge_slashes) para minimizar a probabilidade de vulnerabilidades de confusão de caminhos. As vulnerabilidades de confusão de caminhos ocorrem quando as partes que participam da solicitação usam representações de caminho diferentes.

- Correspondência exata: o parâmetro exato desabilita a correspondência parcial de uma rota e garante que ela retorne a rota somente se o caminho for uma correspondência EXATA com o URL atual.
- Correspondência Regex: usado para descrever padrões em que vários URLs podem realmente identificar uma única página no site.

- (Opcional) Parâmetros de consulta: esse campo permite que você faça a correspondência com os parâmetros da consulta.
- (Opcional) Cabeçalhos: especifica os cabeçalhos para http e http2. Ele deve corresponder à solicitação de entrada para rotear para o serviço virtual de destino.

- Se grpc for o tipo selecionado:
  - Nome do serviço: o serviço de destino ao qual corresponder à solicitação.
  - Nome do método: o método de destino ao qual corresponder à solicitação.
  - (Opcional) Metadados: especifica a `Match` com base na presença de metadados. Todos devem corresponder para que a solicitação seja processada.

12. Selecione Criar rota.

AWS CLI

Para criar uma rota usando a AWS CLI.

Crie uma rota gRPC usando o comando a seguir e JSON de entrada (substitua os valores em *vermelho* pelos seus):

1.
```
aws appmesh create-route \
     --cli-input-json file://create-route-grpc.json
```

2. Conteúdo do exemplo create-route-grpc.json

```
{
    "meshName" : "meshName",
    "routeName" : "routeName",
    "spec" : {
       "grpcRoute" : {
          "action" : {
             "weightedTargets" : [
                {
                   "virtualNode" : "nodeName",
                   "weight" : 100
                }
             ]
          },
          "match" : {
             "metadata" : [
                {
                   "invert" : false,
                   "match" : {
                      "prefix" : "123"
                   },
                   "name" : "myMetadata"
                }
             ],
             "methodName" : "nameOfmethod",
             "serviceName" : "serviceA.svc.cluster.local"
          },
          "retryPolicy" : {
             "grpcRetryEvents" : [ "deadline-exceeded" ],
             "httpRetryEvents" : [ "server-error", "gateway-error" ],
             "maxRetries" : 3,
             "perRetryTimeout" : {
                "unit" : "s",
                "value" : 15
             },
             "tcpRetryEvents" : [ "connection-error" ]
          }
       },
       "priority" : 100
    },
    "virtualRouterName" : "routerName"
}
```

3. Resultado do exemplo:

```
{
    "route": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName/route/routeName",
            "createdAt": "2022-04-06T13:48:20.749000-05:00",
            "lastUpdatedAt": "2022-04-06T13:48:20.749000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 1
        },
        "routeName": "routeName",
        "spec": {
            "grpcRoute": {
                "action": {
                    "weightedTargets": [
                        {
                            "virtualNode": "nodeName",
                            "weight": 100
                        }
                    ]
                },
                "match": {
                    "metadata": [
                        {
                            "invert": false,
                            "match": {
                                "prefix": "123"
                            },
                            "name": "myMetadata"
                        }
                    ],
                    "methodName": "nameOfMehod",
                    "serviceName": "serviceA.svc.cluster.local"
                },
                "retryPolicy": {
"grpcRetryEvents": [
                        "deadline-exceeded"
                    ],
                    "httpRetryEvents": [
                        "server-error",
```

```
                        "gateway-error"
                    ],
                    "maxRetries": 3,
                    "perRetryTimeout": {
                        "unit": "s",
                        "value": 15
                    },
                    "tcpRetryEvents": [
                        "connection-error"
                    ]
                }
            },
            "priority": 100
        },
        "status": {
            "status": "ACTIVE"
        },
        "virtualRouterName": "routerName"
    }
}
```

Para obter mais informações sobre como criar uma rota com a AWS CLI para App Mesh, consulte o comando create-route na referência da AWS CLI.

gRPC

(Opcional) Correspondência

- (Opcional) Insira o Nome do serviço do serviço de destino ao qual corresponder a solicitação. Se você não especificar um nome, as solicitações para qualquer serviço serão correspondidas.
- (Opcional) Insira o Nome do método do método de destino ao qual corresponder a solicitação. Se você não especificar um nome, as solicitações para qualquer método serão correspondidas. Se você especificar um nome do método, deverá especificar um nome do serviço.

(Opcional) Metadados

Escolha Add Metadata (Adicionar metadados).

- (Opcional) Insira o Nome dos metadados nos quais você deseja rotear, selecione um Tipo de correspondência e insira um Valor de correspondência. Selecionar Inverter corresponderá ao

oposto. Por exemplo, se você especificar um Nome dos metadados de `myMetadata`, um Tipo de correspondência de Exato, um Valor de correspondência de `123` e selecionar Inverter, a rota será correspondida para qualquer solicitação que tenha um nome dos metadados que comece com algo diferente de `123`.

- (Opcional) Selecione Adicionar metadados para adicionar até dez itens de metadados.

(Opcional) Política de novas tentativas

Uma política de repetição permite que os clientes se protejam contra falhas de rede intermitentes ou falhas intermitentes no lado do servidor. Uma política de novas tentativas é opcional, mas recomendada. Os valores de tempo limite de novas tentativas definem o tempo limite por nova tentativa (incluindo a tentativa inicial). Se você não definir uma política de novas tentativas, o App Mesh poderá criar automaticamente uma política padrão para cada uma de suas rotas. Para obter mais informações, consulte Política padrão de tentativas de rotas.

- Em Tempo limite de novas tentativas, insira o número de unidades para a duração do tempo limite. Um valor é necessário se você selecionar qualquer evento de tentativa de protocolo.
- Em Unidade do tempo limite de novas tentativas, selecione uma unidade. Um valor é necessário se você selecionar qualquer evento de tentativa de protocolo.
- Em Máximo de novas tentativas, insira o número máximo de novas tentativas quando a solicitação falhar. Um valor é necessário se você selecionar qualquer evento de tentativa de protocolo. Recomendamos um valor de pelo menos dois.
- Selecione um ou mais Eventos de novas tentativas HTTP. Recomendamos selecionar pelo menos stream-error e gateway-error.
- Selecione um Evento de novas tentativas TCP.
- Selecione um ou mais Eventos de novas tentativas gRPC. Recomendamos selecionar pelo menos cancelado e indisponível.

(Opcional) Tempos limite

- O padrão é 15 segundos. Se você especificou uma Política de novas tentativas, a duração especificada aqui sempre deverá ser maior ou igual à duração da nova tentativa multiplicada pelo Máximo de novas tentativas definido na Política de novas tentativas para que sua política de novas tentativas possa ser concluída. Se você especificar uma duração maior que 15 segundos, certifique-se de que o tempo limite especificado para o receptor de qualquer nó virtual de Destino também seja maior que 15 segundos. Para obter mais informações, consulte Nós virtuais.

- Um valor de `0` desabilita o tempo limite.
- O valor de tempo máximo em que a rota pode ficar ociosa.

HTTP e HTTP/2

(Opcional) Correspondência

- Especifique o Prefixo ao qual a rota deve corresponder. Por exemplo, se o nome do serviço virtual for `service-b.local` e você desejar que a rota corresponda as solicitações a `service-b.local/metrics`, seu prefixo deverá ser `/metrics`. Especificar rotas `/` para todo o tráfego.
- (Opcional) Selecione um Método.
- (Opcional) Selecione um Esquema. Aplicável apenas para rotas HTTP2.

(Opcional) Cabeçalhos

- (Opcional) Selecione Adicionar cabeçalho. Insira o Nome do cabeçalho no qual você deseja rotear, selecione um Tipo de correspondência e insira um Valor de correspondência. Selecionar Inverter corresponderá ao oposto. Por exemplo, se você especificar um cabeçalho chamado `clientRequestId` com um Prefixo de `123` e selecionar Inverter, a rota corresponderá a qualquer solicitação que tenha um cabeçalho que comece com algo diferente de `123`.
- (Opcional) Selecione Adicionar cabeçalho. Você pode adicionar até dez cabeçalhos.

(Opcional) Política de novas tentativas

Uma política de repetição permite que os clientes se protejam contra falhas de rede intermitentes ou falhas intermitentes no lado do servidor. Uma política de novas tentativas é opcional, mas recomendada. Os valores de tempo limite de novas tentativas definem o tempo limite por nova tentativa (incluindo a tentativa inicial). Se você não definir uma política de novas tentativas, o App Mesh poderá criar automaticamente uma política padrão para cada uma de suas rotas. Para obter mais informações, consulte Política padrão de tentativas de rotas.

- Em Tempo limite de novas tentativas, insira o número de unidades para a duração do tempo limite. Um valor é necessário se você selecionar qualquer evento de tentativa de protocolo.
- Em Unidade do tempo limite de novas tentativas, selecione uma unidade. Um valor é necessário se você selecionar qualquer evento de tentativa de protocolo.

- Em Máximo de novas tentativas, insira o número máximo de novas tentativas quando a solicitação falhar. Um valor é necessário se você selecionar qualquer evento de tentativa de protocolo. Recomendamos um valor de pelo menos dois.
- Selecione um ou mais Eventos de novas tentativas HTTP. Recomendamos selecionar pelo menos stream-error e gateway-error.
- Selecione um Evento de novas tentativas TCP.

(Opcional) Tempos limite

- Tempo limite da solicitação: o padrão é 15 segundos. Se você especificou uma Política de novas tentativas, a duração especificada aqui sempre deverá ser maior ou igual à duração da nova tentativa multiplicada pelo Máximo de novas tentativas definido na Política de novas tentativas para que sua política de novas tentativas possa ser concluída.
- Duração do tempo ocioso: o padrão é 300 segundos.
- Um valor de `0` desabilita o tempo limite.

> **Note**
>
> Se você especificar um tempo limite maior que o padrão, certifique-se de que o tempo limite especificado para o receptor para todos os participantes do nó virtual também seja maior que o padrão. No entanto, se você diminuir o tempo limite para um valor menor que o padrão, é opcional atualizar os tempos limite nos nós virtuais. Para obter mais informações, consulte Nós virtuais.

TCP

(Opcional) Tempos limite

- Duração do tempo ocioso: o padrão é 300 segundos.
- Um valor de `0` desabilita o tempo limite.

Excluir uma rota

AWS Management Console

Para excluir uma rota usando o AWS Management Console

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. Escolha a malha da qual você deseja excluir uma rota. Todas as malhas que você possui e que foram compartilhadas com você estão listadas.
3. Selecione Virtual routers (Roteadores virtuais) no painel de navegação à esquerda.
4. Escolha o roteador do qual você deseja excluir uma rota.
5. Na tabela Rotas, escolha a rota que você deseja excluir e selecione Excluir no canto superior direito.
6. Na caixa de confirmação, digite **`delete`** e clique em Excluir.

AWS CLI

Para excluir uma rota usando a AWS CLI

1. Use o comando a seguir para excluir a rota (substitua os valores em *`vermelho`* pelos seus):

```
aws appmesh delete-route \
     --mesh-name meshName \
     --virtual-router-name routerName \
     --route-name routeName
```

2. Resultado do exemplo:

```
{
    "route": {
        "meshName": "meshName",
        "metadata": {
            "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualRouter/routerName/route/routeName",
            "createdAt": "2022-04-06T13:46:54.750000-05:00",
            "lastUpdatedAt": "2022-04-07T10:43:57.152000-05:00",
            "meshOwner": "123456789012",
            "resourceOwner": "210987654321",
            "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
            "version": 2
```

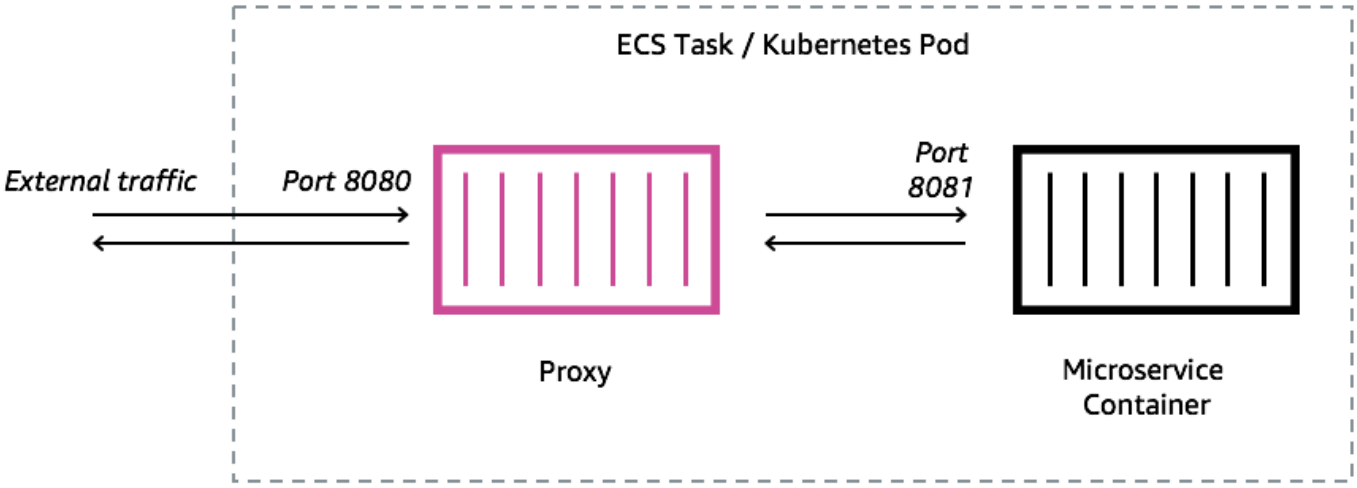
```
        },
        "routeName": "routeName",
        "spec": {
            "grpcRoute": {
                "action": {
                    "weightedTargets": [
                        {
                            "virtualNode": "nodeName",
                            "weight": 100
                        }
                    ]
                },
                "match": {
                    "metadata": [
                        {
                            "invert": false,
                            "match": {
                                "prefix": "123"
                            },
                            "name": "myMetadata"
                        }
                    ],
                    "methodName": "methodName",
                    "serviceName": "serviceA.svc.cluster.local"
                },
                "retryPolicy": {
                    "grpcRetryEvents": [
                        "deadline-exceeded"
                    ],
                    "httpRetryEvents": [
                        "server-error",
                        "gateway-error"
                    ],
                    "maxRetries": 3,
                    "perRetryTimeout": {
                        "unit": "s",
                        "value": 15
                    },
                    "tcpRetryEvents": [
                        "connection-error"
                    ]
                }
            },
            "priority": 100
```

```
        },
        "status": {
            "status": "DELETED"
        },
        "virtualRouterName": "routerName"
    }
}
```

Para obter mais informações sobre como excluir uma rota com a AWS CLI para App Mesh, consulte o comando delete-route na referência da AWS CLI.

Imagem do Envoy

AWS App Mesh é uma malha de serviços baseada no proxy Envoy.



Você deve adicionar um proxy do Envoy à tarefa do Amazon ECS, ao pod do Kubernetes ou à instância do Amazon EC2 representada pelo seu endpoint do App Mesh, como um nó virtual ou gateway virtual. O App Mesh vende uma imagem de contêiner proxy Envoy que é corrigida com as atualizações mais recentes de vulnerabilidade e desempenho. O App Mesh testa cada nova versão do proxy Envoy em relação ao conjunto de recursos do App Mesh antes de disponibilizar uma nova imagem para você.

Variantes de imagem do Envoy

O App Mesh fornece duas variantes da imagem do contêiner proxy Envoy. As diferenças entre os dois são como o proxy Envoy se comunica com o plano de dados do App Mesh e como os proxies Envoy se comunicam entre si. Uma delas é uma imagem padrão, que se comunica com os endpoints padrão do serviço App Mesh. A outra variante é compatível com FIPS, que se comunica com os terminais do serviço FIPS do App Mesh e aplica a criptografia FIPS na comunicação TLS entre os serviços do App Mesh.

Você pode escolher uma imagem regional da lista abaixo ou uma imagem do nosso repositório público chamada `aws-appmesh-envoy`.

Important

- A partir de 30 de junho de 2023, somente a imagem Envoy `v1.17.2.0-prod` ou posterior é compatível para uso com o App Mesh. Para clientes atuais que usavam uma imagem do Envoy anteriormente`v1.17.2.0`, embora os envios existentes continuem sendo compatíveis, é altamente recomendável migrar para a versão mais recente.
- Como prática recomendada, é altamente recomendável atualizar com frequência a versão do Envoy para a versão mais recente. Somente a versão mais recente do Envoy é validada com os patches de segurança, lançamentos de recursos e melhorias de desempenho mais recentes.
- A versão `1.17` foi uma atualização significativa para o Envoy. Consulte Atualização/migração para o Envoy 1.17 para mais detalhes.
- A versão `1.20.0.1` ou posterior é compatível com `ARM64`.
- Para suporte ao `IPv6`, é necessária a versão `1.20` do Envoy ou posterior.

Todas as regiões suportadas, exceto `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`. Você pode substituir o *`Código de região`* por qualquer Região que não seja `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1` e `af-south-1`.

Padrão

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

`me-south-1`

Padrão

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

ap-east-1

Padrão

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

ap-southeast-3

Padrão

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-
fips
```

eu-south-1

Padrão

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

il-central-1

Padrão

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

`af-south-1`

Padrão

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

`Public repository`

Padrão

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod
```

Compatível com FIPS

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.27.3.0-prod-fips
```

> **Note**
>
> Recomendamos a alocação de 512 unidades de CPU e pelo menos 64 MiB de memória para o contêiner do Envoy. No Fargate, a menor quantidade de memória que você pode definir é 1024 MiB. A alocação de recursos para o contêiner do Envoy pode ser aumentada se os insights do contêiner ou outras métricas indicarem recursos insuficientes devido à maior carga.

> **Note**
>
> Todas as versões de lançamento de imagens do `aws-appmesh-envoy` a partir de `v1.22.0.0` são criadas como uma imagem do Docker sem distribuição. Fizemos essa

alteração para que pudéssemos reduzir o tamanho da imagem e reduzir nossa exposição à vulnerabilidade em pacotes não utilizados presentes na imagem. Se você está construindo com base na aws-appmesh-envoy imagem e está confiando em alguns dos pacotes básicos do AL2 (por exemplo, yum) e funcionalidades, sugerimos que você copie os binários de dentro de uma `aws-appmesh-envoy` imagem para criar uma nova imagem do Docker com base AL2.
Execute esse script para gerar uma imagem do docker personalizada com a tag `aws-appmesh-envoy:v1.22.0.0-prod-al2:`

```
cat << EOF > Dockerfile
FROM public.ecr.aws/appmesh/aws-appmesh-envoy:v1.22.0.0-prod as envoy

FROM public.ecr.aws/amazonlinux/amazonlinux:2
RUN yum -y update && \
    yum clean all && \
    rm -rf /var/cache/yum

COPY --from=envoy /usr/bin/envoy /usr/bin/envoy
COPY --from=envoy /usr/bin/agent /usr/bin/agent
COPY --from=envoy /aws_appmesh_aggregate_stats.wasm /
aws_appmesh_aggregate_stats.wasm

CMD [ "/usr/bin/agent" ]
EOF

docker build -f Dockerfile -t aws-appmesh-envoy:v1.22.0.0-prod-al2 .
```

O acesso a essa imagem de contêiner no Amazon ECR é controlado pelo AWS Identity and Access Management (IAM). Como resultado, você deve usar o IAM para verificar se você tem acesso de leitura ao Amazon ECR. Por exemplo, ao usar o Amazon ECS, você pode atribuir uma função apropriada de execução de tarefas a uma tarefa do Amazon ECS. Se você usa políticas do IAM que limitam o acesso a recursos específicos do Amazon ECR, certifique-se de verificar se você permite acesso ao nome do recurso da Amazon (ARN) específico da região que identifica o repositório `aws-appmesh-envoy`. Por exemplo, na região `us-west-2`, você permite acesso ao seguinte recurso: `arn:aws:ecr:us-west-2:840364872350:repository/aws-appmesh-envoy`. Para obter mais informações, consulte Políticas gerenciadas do Amazon ECR. Se você estiver usando o Docker em uma instância do Amazon EC2, autentique o Docker no repositório. Para obter mais informações, consulte Autenticação de registro.

Ocasionalmente, lançamos novos atributos do App Mesh que dependem das alterações do Envoy que ainda não foram mescladas às imagens upstream do Envoy. Para usar esses novos atributos do App Mesh antes que as alterações do Envoy sejam mescladas no upstream, você deve usar a imagem de contêiner do Envoy fornecida pelo App Mesh. Para ver uma lista de mudanças, consulte os problemas do GitHub roteiro do App Mesh com o `Envoy Upstream` rótulo. Recomendamos o uso da imagem do contêiner do App Mesh como a melhor opção compatível.

Variáveis de configuração do Envoy

Use as seguintes variáveis de ambiente para configurar os contêineres do Envoy para seus grupos de tarefas do nó virtual do App Mesh.

> **Note**
>
> O App Mesh Envoy 1.17 não é compatível com a API v2 xDS do Envoy. Se você estiver usando variáveis de configuração do Envoy que aceitam arquivos de configuração do Envoy, elas devem ser atualizadas para a API v3 xDS mais recente.

Variáveis obrigatórias

A variável de ambiente a seguir é necessária para todos os contêineres do App Mesh Envoy. Essa variável só pode ser usada com a versão `1.15.0` ou posterior da imagem do Envoy. Se você estiver usando uma versão anterior da imagem, deverá definir a variável `APPMESH_VIRTUAL_NODE_NAME` em vez disso.

`APPMESH_RESOURCE_ARN`

Ao adicionar o contêiner do Envoy a um grupo de tarefas, defina essa variável de ambiente para o ARN do nó virtual ou do gateway virtual que o grupo de tarefas representa. A lista a seguir contém exemplos de ARNs:

- Nó virtual – *arn:aws:appmesh: Código da região: 111122223333:mesh/ MeshName /VirtualNode/ virtualNodeName*
- Gateway virtual – *arn:aws:appmesh: Código da região: 111122223333:mesh/ MeshName /VirtualGateway/ virtualGatewayName*

Ao usar o Canal de demonstração do App Mesh, os ARNs devem usar a região *us-west-2* e usar `appmesh-preview`, em vez de `appmesh`. Por exemplo, o ARN de um nó

virtual no Canal de demonstração do App Mesh é `arn:aws:appmesh-preview:us-west-2:111122223333:mesh/meshName/virtualNode/virtualNodeName`.

Variáveis opcionais

A variável de ambiente a seguir é opcional para todos os contêineres do App Mesh Envoy.

`ENVOY_LOG_LEVEL`

Especifica o nível de log do contêiner do Envoy.

Valores válidos: `trace`, `debug`, `info`, `warn`, `error`, `critical`, `off`

Padrão: `info`

`ENVOY_INITIAL_FETCH_TIMEOUT`

Especifica a quantidade de tempo que o Envoy espera pela primeira resposta de configuração do servidor de gerenciamento durante o processo de inicialização.

Para obter mais informações, consulte Configuração de fontes, na documentação do Envoy. Quando definido como `0`, não há tempo limite.

Padrão: `0`

`ENVOY_CONCURRENCY`

Define a opção de linha de comando `--concurrency` ao iniciar o Envoy. Isso não é definido por padrão. Essa opção está disponível na versão `v1.24.0.0-prod` ou superior do Envoy.

Para obter mais informações, consulte Opções da linha de comando na documentação do Envoy.

Variáveis de administração

Usam essas variáveis de ambiente para configurar a interface administrativa do Envoy.

`ENVOY_ADMIN_ACCESS_PORT`

Especificam uma porta de administração personalizada para o Envoy receber. Padrão: `9901`.

> **Note**
>
> A porta de administração do Envoy deve ser diferente de qualquer porta de ouvinte no gateway virtual ou no nó virtual.

`ENVOY_ADMIN_ACCESS_LOG_FILE`

Especificam um caminho personalizado no qual os logs de acesso do Envoy serão gravados. Padrão: `/tmp/envoy_admin_access.log`.

`ENVOY_ADMIN_ACCESS_ENABLE_IPV6`

Alterna a interface de administração do Envoy para aceitar tráfego `IPv6`, o que permite que essa interface aceite tanto o tráfego `IPv4` quanto o tráfego `IPv6`. Por padrão, esse sinalizador é definido como falso e o Envoy só recebe o tráfego `IPv4`. Essa variável só pode ser usada com a versão 1.22.0 ou posterior da imagem do Envoy.

Variáveis de agente

Use essas variáveis de ambiente para configurar o AWS App Mesh Agent for Envoy. Para obter mais informações, consulte o Agente para Envoy no App Mesh.

`APPNET_ENVOY_RESTART_COUNT`

Especifica o número de vezes que o Agente reinicia o processo de proxy do Envoy em uma tarefa ou pod em execução, caso ele saia. O Agente também registra o status de saída toda vez que o Envoy sai para facilitar a solução de problemas. O valor padrão da variável é `0`. Quando o valor padrão é definido, o Agente não tenta reiniciar o processo.

Padrão: `0`

Máximo: `10`

`PID_POLL_INTERVAL_MS`

Especifica o intervalo em milissegundos no qual o estado do processo do proxy do Envoy é verificado pelo Agente. O valor padrão é `100`.

Padrão: `100`

Mínimo: `100`

Máximo: `1000`

`LISTENER_DRAIN_WAIT_TIME_S`

Especifica a quantidade de tempo em segundos que o proxy do Envoy espera que as conexões ativas sejam fechadas antes que o processo seja encerrado.

Padrão: `20`

Mínimo: `5`

Máximo: `110`

`APPNET_AGENT_ADMIN_MODE`

Inicia o servidor da interface de gerenciamento do Agente e o vincula a um endereço tcp ou a um soquete unix.

Valores válidos: `tcp`, `uds`

`APPNET_AGENT_HTTP_PORT`

Especifique uma porta a ser usada para vincular a interface de gerenciamento do Agente no modo `tcp`. Certifique-se de que o valor da porta seja > `1024` se `uid` ! = `0`. Certifique-se de que a porta seja menor que `65535`.

Padrão: `9902`

`APPNET_AGENT_ADMIN_UDS_PATH`

Especifique o caminho do soquete de domínio unix para a interface de gerenciamento do Agente no modo `uds`.

Padrão: `/var/run/ecs/appnet_admin.sock`

Variáveis de rastreamento

É possível configurar um ou nenhum dos seguintes drivers de rastreamento.

AWS X-Ray variáveis

Use as seguintes variáveis de ambiente para configurar o App Mesh com AWS X-Ray. Para obter mais informações, consulte o Guia do desenvolvedor do AWS X-Ray.

`ENABLE_ENVOY_XRAY_TRACING`

Ativa o rastreamento do X-Ray usando `127.0.0.1:2000` como o endpoint padrão do daemon. Para habilitar, defina o valor como `1`. O valor padrão é `0`.

`XRAY_DAEMON_PORT`

Especifique um valor de porta para substituir a porta padrão do daemon X-Ray: `2000`.

`XRAY_SAMPLING_RATE`

Especifique uma taxa de amostragem para substituir a taxa de amostragem padrão do rastreador do X-Ray de `0.05` (5%). Especifique o valor como decimal entre `0` e `1.00` (100%). Esse valor é substituído se `XRAY_SAMPLING_RULE_MANIFEST` for especificado. Essa variável é compatível com imagens do Envoy da versão `v1.19.1.1-prod` e posterior.

`XRAY_SAMPLING_RULE_MANIFEST`

Especifique um caminho de arquivo no sistema de arquivos do contêiner Envoy para configurar as regras de amostragem personalizadas localizadas para o rastreador do X-Ray. Para obter mais informações, consulte Regras de amostragem no Guia do desenvolvedor do AWS X-Ray . Essa variável é compatível com imagens do Envoy da versão `v1.19.1.0-prod` e posterior.

`XRAY_SEGMENT_NAME`

Especifique um nome de segmento para traços para substituir o nome padrão do segmento do X-Ray. Por padrão, esse valor é definido como `mesh/resourceName`. Essa variável é compatível com imagens do Envoy da versão `v1.23.1.0-prod` ou posterior.

Variáveis de rastreamento do Datadog

As variáveis de ambiente a seguir ajudam você a configurar o App Mesh com o rastreador do agente Datadog. Para obter mais informações, consulte Agente de configuração, na documentação do Datadog.

`ENABLE_ENVOY_DATADOG_TRACING`

Permite a coleta de rastreamento do Datadog usando `127.0.0.1:8126` como endpoint padrão do agente Datadog. Para ativar, defina o valor como `1` (o valor padrão é `0`).

`DATADOG_TRACER_PORT`

Especifique um valor de porta para substituir a porta padrão do Datadog: `8126`.

`DATADOG_TRACER_ADDRESS`

Especifique um endereço IP para substituir o endereço padrão do agente do Datadog: `127.0.0.1`.

`DD_SERVICE`

Especifique um nome de serviço para traços para substituir o nome padrão do serviço do Datadog: `envoy-meshName/virtualNodeName`. Essa variável é compatível com imagens do Envoy da versão `v1.18.3.0-prod` e posterior.

Variáveis de rastreamento de Jaeger

Use as seguintes variáveis de ambiente para configurar o App Mesh com rastreamento de Jaeger. Para obter mais informações, consulte Conceitos básicos na Documentação do Jaeger. Essas variáveis são compatíveis com imagens do Envoy da versão `1.16.1.0-prod` e posterior.

`ENABLE_ENVOY_JAEGER_TRACING`

Ativa a coleta de traços do Jaeger usando `127.0.0.1:9411` como endpoint padrão do Jaeger. Para ativar, defina o valor como 1 (o valor padrão é `0`).

`JAEGER_TRACER_PORT`

Especifique um valor de porta para substituir a porta padrão do Jaeger: `9411`.

`JAEGER_TRACER_ADDRESS`

Especifique um endereço IP para substituir o endereço padrão do agente do Jaeger: `127.0.0.1`.

`JAEGER_TRACER_VERSION`

Especifique se o coletor precisa de traços em `JSON` ou `PROTO` em formato codificado. Por padrão, esse valor é definido como `PROTO`. Essa variável é compatível com imagens do Envoy da versão `v1.23.1.0-prod` ou posterior.

Variável de rastreamento do Envoy

Defina a variável de ambiente a seguir para usar sua própria configuração de rastreamento.

`ENVOY_TRACING_CFG_FILE`

Especifique um caminho de arquivo no sistema de arquivos do contêiner Envoy. Para obter mais informações, consulte `config.trace.v3.Tracing` na documentação do Envoy.

> **Note**
>
> Se a configuração de rastreamento exigir a especificação de um cluster de rastreamento, certifique-se de configurar a configuração do cluster associado abaixo `static_resources` no mesmo arquivo de configuração de rastreamento. Por exemplo, o Zipkin tem um campo `collector_cluster` para o nome do cluster que hospeda os coletores de rastreamento, e esse cluster precisa ser definido estaticamente.

DogStatsVariáveis D

Use as seguintes variáveis de ambiente para configurar o App Mesh com DogStats D. Para obter mais informações, consulte a documentação DogStatsD.

`ENABLE_ENVOY_DOG_STATSD`

Ativa o uso de estatísticas DogStats D `127.0.0.1:8125` como endpoint padrão do daemon. Para habilitar, defina o valor como 1.

`STATSD_PORT`

Especifique um valor de porta para substituir a porta padrão do daemon DogStats D.

`STATSD_ADDRESS`

Especifique um valor de endereço IP para substituir o endereço IP padrão do daemon DogStats D. Padrão: `127.0.0.1`. Essa variável só pode ser usada com a versão `1.15.0` ou posterior da imagem do Envoy.

`STATSD_SOCKET_PATH`

Especifique um soquete de domínio unix para o daemon DogStats D. Se essa variável não for especificada e DogStats D estiver habilitada, esse valor será padronizado para a porta de endereço IP do daemon DogStats D de. `127.0.0.1:8125` Se a `ENVOY_STATS_SINKS_CFG_FILE` variável for especificada contendo uma configuração de coletores de estatísticas, ela substituirá todas as DogStats variáveis D. Essa variável é compatível com imagens do Envoy da versão `v1.19.1.0-prod` ou posterior.

Variáveis do App Mesh

As variáveis a seguir ajudam você a configurar o App Mesh.

`APPMESH_PREVIEW`

Defina o valor como 1 para se conectar ao endpoint do Canal de demonstração do App Mesh. Para mais informações sobre o uso do Canal de demonstração do App Mesh, consulte Canal de demonstração do App Mesh.

`APPMESH_RESOURCE_CLUSTER`

Por padrão, o App Mesh usa o nome do recurso especificado em `APPMESH_RESOURCE_ARN` quando o Envoy está se referindo a si mesmo em métricas e rastreamentos. É possível substituir esse comportamento definindo a variável de ambiente `APPMESH_RESOURCE_CLUSTER` com seu próprio nome. Essa variável só pode ser usada com a versão `1.15.0` ou posterior da imagem do Envoy.

`APPMESH_METRIC_EXTENSION_VERSION`

Defina o valor como 1 para ativar a extensão de métricas do App Mesh. Para mais informações sobre o uso da extensão de métricas do App Mesh, consulte Extensão de métricas para o App Mesh.

`APPMESH_DUALSTACK_ENDPOINT`

Defina o valor como 1 para se conectar ao endpoint do App Mesh Dual Stack. Quando esse sinalizador é definido, o Envoy usa nosso domínio com capacidade de pilha dupla. Por padrão, esse sinalizador é definido como falso e se conecta apenas ao nosso domínio `IPv4`. Essa variável só pode ser usada com a versão 1.22.0 ou posterior da imagem do Envoy.

Variáveis de estatísticas do Envoy

Use as seguintes variáveis de ambiente para configurar o App Mesh com Envoy. Para obter mais informações, consulte a documentação Estatísticas do Envoy.

`ENABLE_ENVOY_STATS_TAGS`

Permite o uso de tags definidas pelo App Mesh `appmesh.mesh` e `appmesh.virtual_node`. Para obter mais informações, consulte config.metrics.v3. TagSpecifierna documentação do Envoy. Para habilitar, defina o valor como 1.

`ENVOY_STATS_CONFIG_FILE`

Especifique um caminho de arquivo no sistema de arquivos do contêiner Envoy para substituir o arquivo de configuração padrão das tags Stats pelo seu próprio. Para obter mais informações, consulte config.metrics.v3. StatsConfig.

> **Note**
>
> Definir uma configuração de estatísticas personalizada que inclua filtros de estatísticas pode levar o Envoy a entrar em um estado em que não será mais sincronizado adequadamente com o estado mundial do App Mesh. Isso é um bug do Envoy. Nossa recomendação é não realizar nenhuma filtragem de estatísticas no Envoy. Se a filtragem for absolutamente necessária, listamos algumas soluções alternativas para esse problema em nosso roteiro.

`ENVOY_STATS_SINKS_CFG_FILE`

Especifique um caminho de arquivo no sistema de arquivos do contêiner Envoy para substituir a configuração padrão pela sua própria. Para obter mais informações, consulte config.metrics.v3.StatsSinkna documentação do Envoy.

Variáveis descontinuadas

As variáveis `APPMESH_VIRTUAL_NODE_NAME` e `APPMESH_RESOURCE_NAME` de ambiente não são mais compatíveis com a versão `1.15.0` ou posterior do Envoy. No entanto, elas ainda são compatíveis com malhas existentes. Em vez de usar essas variáveis com a versão Envoy `1.15.0` ou posterior, use `APPMESH_RESOURCE_ARN` para todos os endpoints do App Mesh.

Padrões do Envoy definidos pelo App Mesh

As seções a seguir fornecem informações sobre os padrões do Envoy para a política de repetição de rota e o disjuntor definidos pelo App Mesh.

Política padrão de tentativas de rotas

Se você não tinha malhas em sua conta antes de 29 de julho de 2020, o App Mesh cria automaticamente uma política de repetição de rota padrão do Envoy para todas as solicitações HTTP, HTTP/2 e gRPC em qualquer malha em sua conta em ou após 29 de julho de 2020. Se você tinha alguma malha em sua conta antes de 29 de julho de 2020, nenhuma política padrão foi criada para nenhuma rota do Envoy que existisse antes, em ou depois de 29 de julho de 2020. Isso ocorre a menos que você abra um ticket com o AWS suporte. Depois que o suporte processa o ticket, a política padrão é criada para todas as rotas futuras do Envoy que o App Mesh criar na

data em que o ticket foi processado ou após a data em que o ticket foi processado. Para obter mais informações sobre as políticas de repetição de rotas do Envoy, consulte config.route.v3.RetryPolicyna documentação do Envoy.

O App Mesh cria uma rota do Envoy quando você cria uma rota do App Mesh ou define um provedor de nós virtuais para um serviço virtual do App Mesh. Embora você possa criar uma política de repetição de rotas do App Mesh, não é possível criar uma política de repetição do App Mesh para um provedor de nós virtuais.

A política padrão não é visível por meio da API App Mesh. A política padrão só é visível por meio do Envoy. Para visualizar a configuração, habilite a interface de administração e envie uma solicitação ao Envoy para obter um `config_dump`. Essa política inclui as seguintes configurações:

- Máximo de novas tentativas: 2
- Eventos de nova tentativa do gRPC: `UNAVAILABLE`
- Eventos de nova tentativa do HTTP: `503`

> **Note**
>
> Não é possível criar uma política de repetição de rota do App Mesh que procure um código de erro HTTP específico. No entanto, uma política de nova tentativa de rota do App Mesh pode procurar `server-error` ou `gateway-error`. Ambos incluem erros `503`. Para ter mais informações, consulte Rotas.

- Evento de nova tentativa de TCP: `connect-failure` e `refused-stream`

> **Note**
>
> Não é possível criar uma política de repetição de rota do App Mesh que procure qualquer desses eventos. No entanto, uma política de nova tentativa de rota do App Mesh pode procurar `connection-error`, que é equivalente a `connect-failure`. Para ter mais informações, consulte Rotas.

- Redefinir: O Envoy tentará novamente se o servidor upstream não responder de forma alguma (tempo limite de desconexão/redefinição/leitura).

Disjuntor padrão

Quando você implanta um Envoy no App Mesh, os valores padrão do Envoy são definidos para algumas das configurações do disjuntor. Para obter mais informações, consulte cluster.CircuitBreakers.Limites na documentação do Envoy. Essas configurações não são visíveis por meio da API do App Mesh. As configurações só são visíveis por meio do Envoy. Para visualizar a configuração, habilite a interface de administração e envie uma solicitação ao Envoy para obter um `config_dump`.

Se você não tinha malhas em sua conta antes de 29 de julho de 2020, para cada Envoy que você implanta em uma malha criada em ou após 29 de julho de 2020, o App Mesh desativa efetivamente os disjuntores alterando os valores padrão do Envoy para as configurações a seguir. Se você tinha alguma malha em sua conta antes de 29 de julho de 2020, os valores padrão do Envoy são definidos para qualquer Envoy que você implanta no App Mesh em ou após 29 de julho de 2020, a menos que você abra um ticket com o suporte. AWS Depois que o suporte processa o ticket, os valores padrão do App Mesh para as seguintes configurações do Envoy são definidas pelo App Mesh em todos os Envoys que você implanta após a data em que o ticket é processado:

- **`max_requests`** – 2147483647
- **`max_pending_requests`** – 2147483647
- **`max_connections`** – 2147483647
- **`max_retries`** – 2147483647

> **Note**
>
> Não importa se seus Envoys têm os valores de disjuntor padrão do Envoy ou do App Mesh, você não pode modificar os valores.

Como atualizar/migrar para o Envoy 1.17

Serviço Secreto de Descoberta com SPIRE

Se você estiver usando o SPIRE (SPIFFE Runtime Environment) com o App Mesh para distribuir certificados de confiança aos seus serviços, verifique se está usando pelo menos a versão `0.12.0` do agente SPIRE (lançada em dezembro de 2020). Esta é a primeira versão compatível com as versões do Envoy posteriores à `1.16`.

Alterações de expressão regular

A partir do Envoy `1.17`, o App Mesh configura o Envoy para usar o mecanismo de expressão regular RE2 por padrão. Essa mudança é evidente para a maioria dos usuários, mas as correspondências em Rotas ou Rotas de Gateway não permitem mais referências antecipadas ou retroativas em expressões regulares.

Visão prospectiva positiva e negativa

Positivo: Uma previsão positiva é uma expressão entre parênteses que começa com `?=`:

```
(?=example)
```

Elas são mais úteis ao substituir uma string porque permitem combinar uma string sem consumir os caracteres como parte da correspondência. Como o App Mesh não suporta a substituição de strings Regex, recomendamos que você as substitua por correspondências regulares.

```
(example)
```

Negativo: uma previsão negativa é uma expressão entre parênteses que começa com `?!`.

```
ex(?!amp)le
```

As expressões entre parênteses são usadas para afirmar que parte da expressão não corresponde a uma determinada entrada. Na maioria dos casos, é possível substituí-los por um quantificador zero.

```
ex(amp){0}le
```

Se a expressão em si for uma classe de caracteres, você poderá negar a classe inteira e marcá-la como opcional usando `?`.

```
prefix(?![0-9])suffix => prefix[^0-9]?suffix
```

Dependendo do seu caso de uso, você também poderá alterar suas rotas para lidar com isso.

```
{
    "routeSpec": {
```

```
        "priority": 0,
        "httpRoute": {
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix(?!suffix)"
                        }
                    }
                ]
            }
        }
    }
}

{
    "routeSpec": {
        "priority": 1,
        "httpRoute": {
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix"
                        }
                    }
                ]
            }
        }
    }
}
```

A primeira correspondência de rota procura um cabeçalho que começa com “prefixo”, mas não é seguido por “sufixo”. A segunda rota age de acordo com todos os outros cabeçalhos que começam com “prefixo”, incluindo aqueles que terminam em “sufixo”. Em vez disso, eles também podem ser revertidos como forma de remover a previsão negativa.

```
{
    "routeSpec": {
        "priority": 0,
        "httpRoute": {
```

```
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix.*?suffix"
                        }
                    }
                ]
            }
        }
    }
}

{
    "routeSpec": {
        "priority": 1,
        "httpRoute": {
            "match": {
                "headers": [
                    {
                        "name": "x-my-example-header",
                        "match": {
                            "regex": "^prefix"
                        }
                    }
                ]
            }
        }
    }
}
```

Este exemplo inverte as rotas para fornecer maior prioridade aos cabeçalhos que terminam em “sufixo”, e todos os outros cabeçalhos que começam com “prefixo” são correspondidos na rota de menor prioridade.

Referências inversas

Uma referência inversa é uma forma de escrever expressões mais curtas repetindo para um grupo anterior entre parênteses. Elas têm esse formulário.

```
(group1)(group2)\1
```

Uma barra invertida \ seguida por um número atua como um espaço reservado para o n-ésimo grupo entre parênteses na expressão. Neste exemplo, `\1` é usado como uma forma alternativa de escrever `(group1)` uma segunda vez.

```
(group1)(group2)(group1)
```

Eles podem ser removidos simplesmente substituindo a referência anterior pelo grupo que está sendo referenciado, como no exemplo.

Agente do Envoy

O Agente é um gerenciador de processos dentro da imagem do Envoy que é fornecida para o App Mesh. O Agente garante que o Envoy continue funcionando, permaneça íntegro e reduza o tempo de inatividade. Ele filtra as estatísticas do Envoy e os dados auxiliares para fornecer uma visão resumida da operação do proxy do Envoy no App Mesh. Isso pode ajudá-lo a solucionar erros relacionados com mais rapidez.

Você pode usar o Agente para configurar o número de vezes que deseja reiniciar o proxy Envoy caso o proxy fique inoperante. Se ocorrer uma falha, o Agente registrará o status de saída conclusivo quando o Envoy sair. Você pode usar isso ao solucionar a falha. O Agente também facilita a drenagem da conexão do Envoy, o que ajuda a tornar seus aplicativos mais resilientes a falhas.

Configure o Agente do Envoy usando estas variáveis:

- `APPNET_ENVOY_RESTART_COUNT`: quando essa variável é definida com um valor diferente de zero, o Agente tenta reiniciar o processo de proxy do Envoy até o número que você definiu quando considera que o status do processo de proxy não está íntegro na sondagem. Isso ajuda a reduzir o tempo de inatividade, fornecendo uma reinicialização mais rápida se comparado com a substituição de uma tarefa ou pod pelo orquestrador de contêineres no caso de falhas na verificação de integridade do proxy.
- `PID_POLL_INTERVAL_MS`: ao configurar essa variável, o padrão é mantido como `100`. Quando definido com esse valor, você permite uma detecção e reinicialização mais rápidas do processo Envoy quando ele sai, em comparação com a substituição de tarefas ou pods por meio de verificações de integridade do orquestrador de contêineres.
- `LISTENER_DRAIN_WAIT_TIME_S`: ao configurar essa variável, considere o tempo limite do orquestrador de contêineres definido para interromper a tarefa ou o pod. Por exemplo, se esse valor for maior que o tempo limite do orquestrador, o proxy Envoy só poderá ser drenado durante o período até que o orquestrador forçosamente interrompa a tarefa ou o pod.

- `APPNET_AGENT_ADMIN_MODE`: quando essa variável é definida como `tcp` ou `uds`, o Agente fornece uma interface de gerenciamento local. Essa interface de gerenciamento serve como um endpoint seguro para interagir com o proxy Envoy e fornece as seguintes APIs para verificações de integridade, dados de telemetria e resume a condição operacional do proxy.
  - `GET /status`: consulta as estatísticas do Envoy e retorna as informações do servidor.
  - `POST /drain_listeners`: drena todos os receptores entrantes.
  - `POST /enableLogging?level=<desired_level>`: altere o nível de registro do Envoy em todos os registradores.
  - `GET /stats/prometheus`: exibe estatísticas do Envoy no formato Prometheus.
  - `GET /stats/prometheus?usedonly`: exibe apenas as estatísticas que o Envoy atualizou.

Para mais informações sobre as variáveis de configuração do Agente, consulte Variáveis de configuração do Envoy.

O novo Agente do AWS App Mesh está incluído nas imagens do Envoy otimizadas para o App Mesh a partir da versão `1.21.0.0` e não requer alocação adicional de recursos nas tarefas ou pods do cliente.

observabilidade do App Mesh

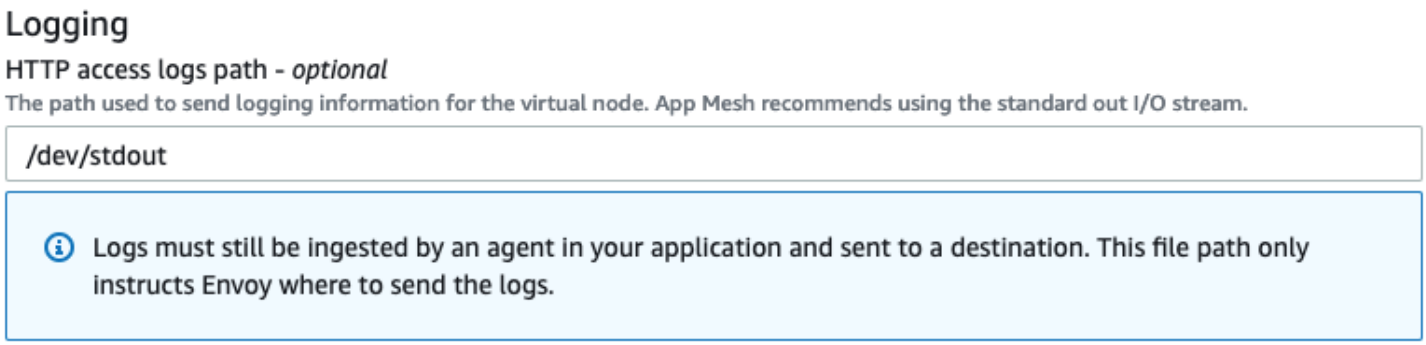
Um dos benefícios de trabalhar com o App Mesh é maior visibilidade de seus aplicativos de microsserviços. O App Mesh é capaz de trabalhar com várias soluções diferentes de log, métricas e rastreamento.

O proxy Envoy e o App Mesh oferecem os seguintes tipos de ferramentas para ajudar você a ter uma visão mais clara de seus aplicativos e proxies:

- Registro em log
- Métricas
- Rastreamento

Registro em log

Ao criar seus nós virtuais e gateways virtuais, você tem a opção de configurar os logs de acesso do Envoy. No console, isso está na seção Registro em log do nó virtual e do gateway virtual para criar ou editar fluxos de trabalho.



A imagem anterior mostra um caminho de log do `/dev/stdout` para logs de acesso do Envoy.

Para `format`, especifique um dos dois formatos possíveis, `json` ou `text`, e o padrão. `json` pega pares de chaves e os transforma em estrutura JSON antes de passá-los para o Envoy.

O bloco de código a seguir mostra a representação JSON que você pode usar no AWS CLI.

```
    "logging": {
       "accessLog": {
```

```
            "file": {
               "path": "/dev/stdout",
                "format" : {
                    // Exactly one of json or text should be specified
                    "json": [ // json will be implemented with key pairs
                        {
                            "key": "string",
                            "value": "string"
                        }
                    ]
                    "text": "string" //e.g. "%LOCAL_REPLY_BODY%:%RESPONSE_CODE%:path=
%REQ(:path)%\n"
                }
            }
         }
      }
```

> **Important**
>
> Certifique-se de verificar se seu padrão de entrada é válido para o Envoy, ou o Envoy rejeitará a atualização e armazenará as alterações mais recentes no `error state`.

Quando você envia os logs de acesso do Envoy para `/dev/stdout`, eles são misturados aos logs do contêiner do Envoy. Você pode exportá-los para um serviço de armazenamento e processamento de logs, como o CloudWatch Logs, usando drivers de log padrão do Docker, como `awslogs`. Para obter mais informações, consulte a seção Como usar o driver de log do awslogs no Guia do desenvolvedor do Amazon ECS. Para exportar somente os logs de acesso do Envoy (e ignorar os outros logs do contêiner do Envoy), você pode definir o `ENVOY_LOG_LEVEL` como `off`. Você pode registrar a solicitação sem a string de consulta incluindo a string de formato `%REQ_WITHOUT_QUERY(X?Y):Z%`. Para ver exemplos, consulte Formatador `ReqWithoutQuery`. Para obter mais informações, consulte Logs de acesso na documentação do Envoy.

Habilitar logs de acesso no Kubernetes

Ao usar o Controlador do App Mesh para Kubernetes, você pode configurar nós virtuais com log de acesso adicionando a configuração de log à especificação do nó virtual, conforme mostrado no exemplo a seguir.

```
---
```

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: virtual-node-name
  namespace: namespace
spec:
  listeners:
    - portMapping:
        port: 9080
        protocol: http
  serviceDiscovery:
    dns:
      hostName: hostname
  logging:
    accessLog:
      file:
        path: "/dev/stdout"
```

Seu cluster deve ter um encaminhador de logs para coletar esses logs, como o Fluentd. Para mais informações, consulte Configuração do Fluentd como um DaemonSet para enviar logs ao CloudWatch Logs.

O Envoy também grava vários logs de depuração de seus filtros para o `stdout`. Esses logs são úteis para obter informações sobre a comunicação do Envoy com o App Mesh e o tráfego de serviço a serviço. Seu nível de log específico pode ser configurado usando a variável de ambiente `ENVOY_LOG_LEVEL`. Por exemplo, o texto a seguir é de um exemplo de log de depuração que mostra o cluster ao qual o Envoy correspondeu a uma solicitação HTTP específica.

```
[debug][router] [source/common/router/router.cc:434] [C4][S17419808847192030829]
 cluster 'cds_ingress_howto-http2-mesh_color_client_http_8080' match for URL '/ping'
```

Firelens e Cloudwatch

O Firelens é um roteador de log de contêiner que você pode usar para coletar logs do Amazon ECS e AWS Fargate. Você pode encontrar um exemplo de uso do Firelens em nosso repositório de amostras da AWS.

Você pode usar o CloudWatch para coletar informações de log e métricas. Você pode encontrar mais informações sobre o CloudWatch em nossa seção de métricas de exportação na documentação do App Mesh.

Como monitorar seu aplicativo usando métricas do Envoy

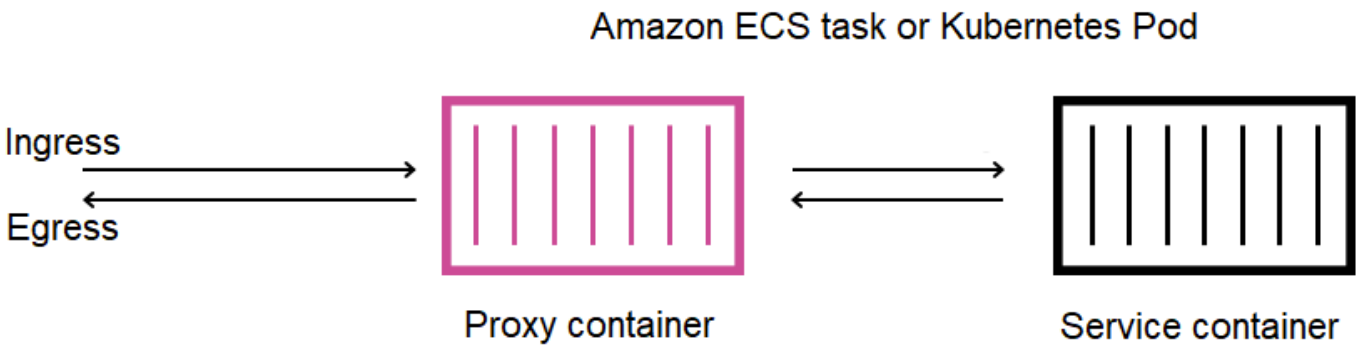
O Envoy classifica suas métricas nas seguintes categorias principais:

- Downstream: métricas relacionadas às conexões e solicitações que chegam ao proxy.
- Upstream: métricas relacionadas às conexões de saída e às solicitações feitas pelo proxy.
- Servidor: métricas que descrevem o estado interno do Envoy. Isso inclui métricas como tempo de atividade ou memória alocada.

No App Mesh, o proxy intercepta o tráfego ascendente e descendente. Por exemplo, as solicitações recebidas de seus clientes, bem como as solicitações feitas pelo seu contêiner de serviço, são classificadas como tráfego downstream pelo Envoy. Para distinguir entre esses diferentes tipos de tráfego upstream e downstream, o App Mesh categoriza ainda mais as métricas do Envoy, dependendo da direção do tráfego em relação ao seu serviço:

- Entrada: métricas e recursos relacionados às conexões e solicitações que fluem para seu contêiner de serviços.
- Saída: métricas e recursos relacionados às conexões e solicitações que fluem do seu contêiner de serviços e, por fim, da sua tarefa do Amazon ECS ou pod do Kubernetes.

A figura a seguir mostra a comunicação entre o proxy e os contêineres de serviço.



Convenções de nomenclatura de recursos

É útil entender como o Envoy vê sua malha e como seus recursos são mapeados de volta aos recursos que você define no App Mesh. Esses são os principais recursos do Envoy que o App Mesh configura:

- Receptores: os endereços e portas em que o proxy recebe as conexões downstream. Na imagem anterior, o App Mesh cria um receptor de entrada para o tráfego que chega à sua tarefa do Amazon ECS ou pod do Kubernetes e um receptor de saída para o tráfego que sai do seu contêiner de serviço.
- Clusters: um grupo nomeado de endpoints upstream aos quais o proxy se conecta e encaminha o tráfego. No App Mesh, seu contêiner de serviço é representado como um cluster, assim como todos os outros nós virtuais aos quais seu serviço pode se conectar.
- Rotas: correspondem às rotas que você define em sua malha. Eles contêm as condições pelas quais o proxy corresponde a uma solicitação, bem como ao cluster de destino para o qual a solicitação é enviada.
- Atribuições de endpoints e carga do cluster: os endereços IP dos clusters upstream. Quando o AWS Cloud Map é usado como mecanismo de descoberta de serviços para nós virtuais, o App Mesh envia instâncias de serviço descobertas como recursos de endpoint para seu proxy.
- Segredos: eles incluem, mas não estão limitados a, suas chaves de criptografia e certificados TLS. Quando AWS Certificate Manager é usado como fonte para certificados de cliente e servidor, o App Mesh envia certificados públicos e privados para seu proxy como recursos secretos.

O App Mesh usa um esquema consistente para nomear recursos do Envoy que você pode usar para relacioná-los à sua malha.

Compreender o esquema de nomenclatura para receptores e clusters é importante para entender as métricas do Envoy no App Mesh.

Nomes de receptores

Os receptores são nomeados usando o seguinte formato:

```
lds_<traffic direction>_<listener IP address>_<listening port>
```

Normalmente, você verá os seguintes receptores configurados no Envoy:

- `lds_ingress_0.0.0.0_15000`
- `lds_egress_0.0.0.0_15001`

Usando um plug-in CNI do Kubernetes ou regras de tabelas IP, o tráfego em sua tarefa do Amazon ECS ou pod do Kubernetes é direcionado para as portas `15000` e `15001`. O App Mesh configura

o Envoy com esses dois receptores para aceitar tráfego de entrada e saída. Se você não tiver um receptor configurado em seu nó virtual, não verá um receptor de entrada.

Nomes do cluster

A maioria dos clusters usa o seguinte formato:

```
cds_<traffic direction>_<mesh name>_<virtual node name>_<protocol>_<port>
```

Cada um dos nós virtuais com os quais seus serviços se comunicam tem seu próprio cluster. Conforme mencionado anteriormente, o App Mesh cria um cluster para o serviço executado ao lado do Envoy para que o proxy possa enviar tráfego de entrada para ele.

Por exemplo, se você tem um nó virtual chamado `my-virtual-node` que recebe o tráfego http na porta `8080` e esse nó virtual está em uma malha chamada `my-mesh`, o App Mesh cria um cluster chamado `cds_ingress_my-mesh_my-virtual-node_http_8080`. Esse cluster serve como destino para o tráfego no contêiner de serviço do `my-virtual-node`.

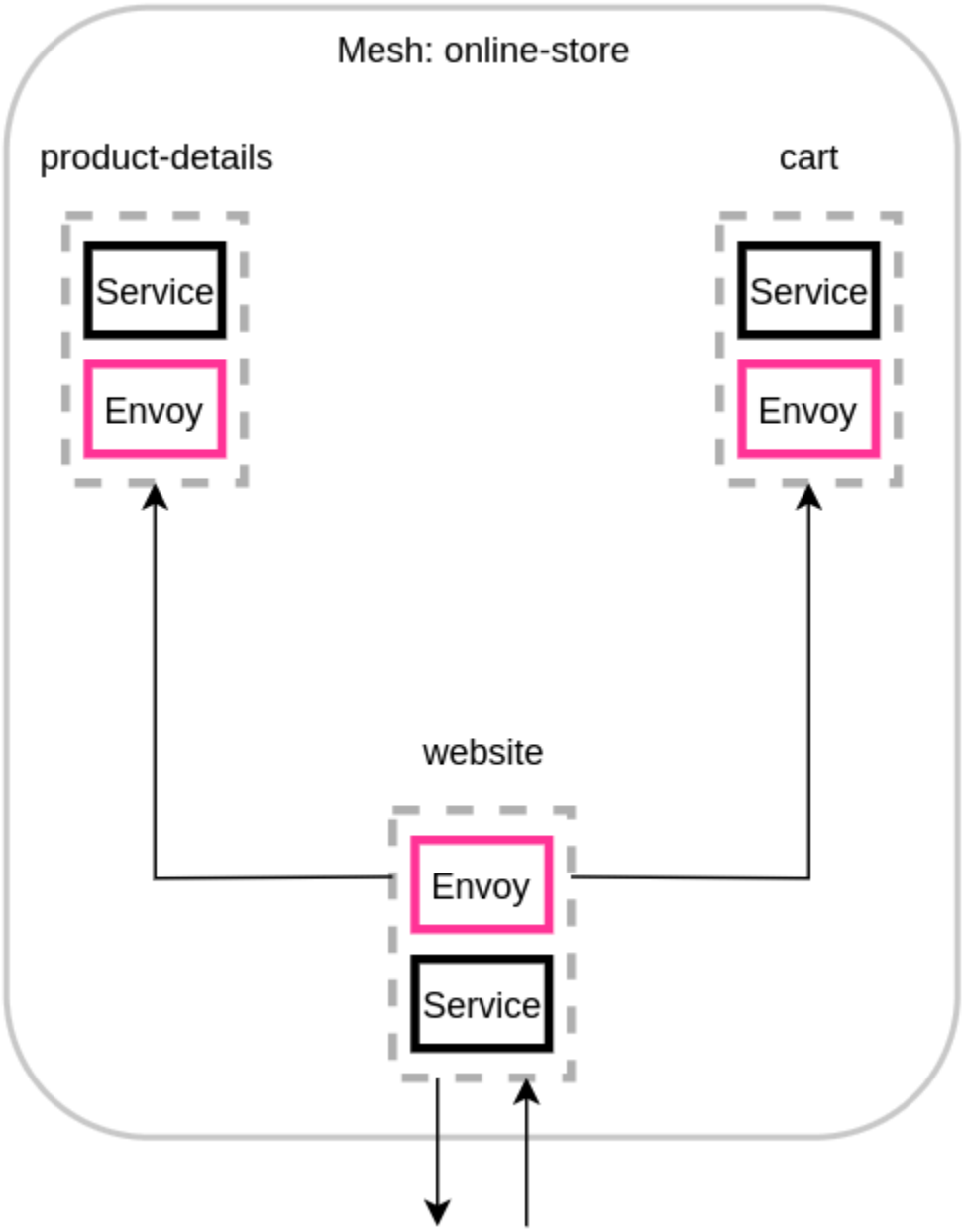
O App Mesh também pode criar os seguintes tipos de clusters especiais adicionais. Esses outros clusters não correspondem necessariamente aos recursos que você define explicitamente em sua malha.

- Clusters usados para alcançar outros serviços da AWS. Esse tipo permite que sua malha alcance a maioria dos serviços da AWS por padrão: `cds_egress_<mesh name>_amazonaws`.
- Cluster usado para realizar roteamento para gateways virtuais. Isso, em geral, pode ser ignorado com segurança:
  - Para receptores únicos: `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<protocol>_<port>`
  - Para vários receptores: `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>`
- O cluster que é o endpoint que você pode definir, como TLS, quando você recupera segredos usando o Serviço de descoberta de segredos do Envoy: `static_cluster_sds_unix_socket`.

Exemplo de métricas de aplicações

Para ilustrar as métricas disponíveis no Envoy, o aplicativo de exemplo a seguir tem três nós virtuais. Os serviços virtuais, roteadores virtuais e rotas na malha podem ser ignorados, pois não são

refletidos nas métricas do Envoy. Neste exemplo, todos os serviços recebem o tráfego http na porta 8080.



Recomendamos adicionar a variável de ambiente `ENABLE_ENVOY_STATS_TAGS=1` aos contêineres proxy do Envoy em execução na sua malha. Isso adiciona as seguintes dimensões de métricas a todas as métricas emitidas pelo proxy:

- `appmesh.mesh`
- `appmesh.virtual_node`

- `appmesh.virtual_gateway`

Essas tags são definidas com o nome de malha, nó virtual ou gateway virtual para permitir a filtragem de métricas usando os nomes dos recursos em sua malha.

Nomes de recurso

O proxy do nó virtual do site tem os seguintes recursos:

- Dois receptores para tráfego de entrada e saída:
  - `lds_ingress_0.0.0.0_15000`
  - `lds_egress_0.0.0.0_15001`
- Dois clusters de saída, representando os dois back-ends de nós virtuais:
  - `cds_egress_online-store_product-details_http_8080`
  - `cds_egress_online-store_cart_http_8080`
- Um cluster de entrada para o contêiner de serviços do site:
  - `cds_ingress_online-store_website_http_8080`

Exemplo de métricas de receptor

- `listener.0.0.0.0_15000.downstream_cx_active`: número de conexões de rede de entrada ativas com o Envoy.
- `listener.0.0.0.0_15001.downstream_cx_active`: número de conexões de rede de saída ativas com o Envoy. As conexões feitas pelo seu aplicativo com serviços externos estão incluídas nessa contagem.
- `listener.0.0.0.0_15000.downstream_cx_total`: número total de conexões de rede de entrada com o Envoy.
- `listener.0.0.0.0_15001.downstream_cx_total`: número total de conexões de rede de saída com o Envoy.

Para ver o conjunto completo de métricas do receptor, consulte Estatísticas na documentação do Envoy.

Exemplos de métricas de cluster

- `cluster_manager.active_clusters`: o número total de clusters com os quais o Envoy estabeleceu pelo menos uma conexão.
- `cluster_manager.warming_clusters`: o número total de clusters aos quais o Envoy ainda não se conectou.

As métricas de cluster a seguir usam o formato de `cluster.<cluster name>.<metric name>`. Esses nomes de métricas são exclusivos do exemplo do aplicativo e são emitidos pelo contêiner Envoy do site:

- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_total`: número total de conexões entre o site e os detalhes do produto.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_connect_fail`: número total de conexões com falha entre o site e os detalhes do produto.
- `cluster.cds_egress_online-store_product-details_http_8080.health_check.failure`: número total de falhas nas verificações de integridade entre o site e os detalhes do produto.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_total`: número total de solicitações feitas entre o site e os detalhes do produto.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_time`: tempo gasto pelas solicitações feitas entre o site e os detalhes do produto.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_2xx`: número de respostas HTTP 2xx recebidas pelo site a partir dos detalhes do produto.

Para ver o conjunto completo de métricas HTTP, consulte Estatísticas na documentação do Envoy.

Métricas do servidor de gerenciamento

O Envoy também emite métricas relacionadas à sua conexão com o ambiente de gerenciamento do App Mesh, que atua como servidor de gerenciamento do Envoy. Recomendamos monitorar algumas dessas métricas como forma de notificá-lo quando seus proxies forem dessincronizados do ambiente

de gerenciamento por longos períodos de tempo. A perda de conectividade com o ambiente de gerenciamento ou falhas nas atualizações impedem que seus proxies recebam novas configurações do App Mesh, incluindo alterações de malha feitas por meio das APIs do App Mesh.

- `control_plane.connected_state`: essa métrica é definida como 1 quando o proxy está conectado ao App Mesh, caso contrário, é 0.
- `*.update_rejected`: número total de atualizações de configuração que são rejeitadas pelo Envoy. Geralmente, isso ocorre devido à configuração incorreta do usuário. Por exemplo, se você configurar o App Mesh para ler um certificado TLS de um arquivo que não pode ser lido pelo Envoy, a atualização contendo o caminho para esse certificado será rejeitada.
  - Para o receptor atualizado rejeitado, as estatísticas serão `listener_manager.lds.update_rejected`.
  - Para o cluster atualizado rejeitado, as estatísticas serão `cluster_manager.cds.update_rejected`.
- `*.update_success`: número de atualizações de configuração bem-sucedidas feitas pelo App Mesh em seu proxy. Isso inclui a carga de configuração inicial enviada quando um novo contêiner Envoy é iniciado.
  - Para o sucesso da atualização do receptor, as estatísticas serão `listener_manager.lds.update_success`.
  - Para o sucesso da atualização do cluster, as estatísticas serão `cluster_manager.cds.update_success`.

Para obter o conjunto de métricas do servidor de gerenciamento, consulte Servidor de gerenciamento na documentação do Envoy.

Como exportar métricas

O Envoy emite muitas estatísticas sobre sua própria operação e sobre várias dimensões do tráfego de entrada e saída. Para saber mais sobre as estatísticas do Envoy, consulte Estatísticas na documentação do Envoy. Essas métricas estão disponíveis por meio do endpoint `/stats` na porta de administração do proxy, o que normalmente é `9901`.

O prefixo `stat` será diferente dependendo se você estiver usando um ou vários receptores. Abaixo estão alguns exemplos para ilustrar as diferenças.

> **Warning**
>
> Se você atualizar seu único receptor para o atributo de vários receptores, poderá enfrentar uma alteração significativa devido ao prefixo estatístico atualizado ilustrado na tabela a seguir.
>
> Sugerimos que você use a imagem do Envoy `1.22.2.1-prod` ou posterior. Isso permite que você veja nomes de métricas semelhantes em seu endpoint Prometheus.

Receptor único (SL)/Estatísticas existentes com o prefixo de receptor "ingress"	Vários receptores (ML)/Novas estatísticas com prefixo do receptor "ingress.<protocol>.<port>"	

`http.*ingress*.rds.rds_ingress_http_5555.version_text`	`http.*ingress.http.5555*.rds.rds_ingress_http_5555.version_text`
	
	`http.*ingress.http.6666*.rds.rds_ingress_http_6666.version_text`	

`listener.0.0.0.0_15000.http.*ingress*.downstream_rq_2xx`	`listener.0.0.0.0_15000.http.*ingress.http.5555*.downstream_rq_2xx`
	
	`listener.0.0.0.0_15000.http.*ingress.http.6666*.downstream_rq_2xx`	

`http.*ingress*.downstream_cx_length_ms`	`http.*ingress.http.5555*.downstream_cx_length_ms`	

Receptor único (SL)/Estatísticas existentes com o prefixo de receptor "ingress"	Vários receptores (ML)/Novas estatísticas com prefixo do receptor "ingress.<protocol>.<port>"	
	`http.*ingress.http.6666*.downstream_cx_length_ms`	

Para mais informações sobre o endpoint de estatísticas, consulte Estatísticas de endpoint na documentação do Envoy. Para mais informações sobre a interface de administração, consulte Ativar a interface de administração do proxy Envoy.

Prometheus para o App Mesh com Amazon EKS

O Prometheus é um toolkit de código aberto para alertas e monitoramento. Um de seus recursos é especificar um formato para emissão de métricas que possam ser consumidas por outros sistemas. Para mais informações sobre o Prometheus, consulte Visão geral na documentação do Prometheus. O Envoy pode emitir suas métricas por meio de seu endpoint de estatísticas passando pelo parâmetro `/stats?format=prometheus`.

Para clientes que estão usando a imagem do Envoy compilação v1.22.2.1-prod, há duas dimensões adicionais para indicar estatísticas específicas do receptor de entrada:

- `appmesh.listener_protocol`
- `appmesh.listener_port`

Abaixo está uma comparação entre as estatísticas existentes do Prometheus e as novas estatísticas.

- Estatísticas existentes com o prefixo de receptor "ingress"

```
envoy_http_downstream_rq_xx{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_node="foodteller-
vn",envoy_response_code_class="2",envoy_http_conn_manager_prefix="ingress"} 931433
```

- Novas estatísticas com "ingress.<protocol>.<port>" + Imagem Appmesh Envoy v1.22.2.1-prod ou posterior

```
envoy_http_downstream_rq_xx{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_node="foodteller-
vn",envoy_response_code_class="2",appmesh_listener_protocol="http",appmesh_listener_port="555
 20
```

- Novas estatísticas com "ingress.<protocol>.<port>" + imagem de compilação personalizada do Envoy

```
envoy_http_http_5555_downstream_rq_xx{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_node="foodteller-
vn",envoy_response_code_class="2",envoy_http_conn_manager_prefix="ingress"} 15983
```

Para vários receptores, o cluster especial `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>` será específico do receptor.

- Estatísticas existentes com o prefixo de receptor "ingress"

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_gateway="tellergateway-vg",Mesh="multiple-listeners-
mesh",VirtualGateway="tellergateway-vg",envoy_cluster_name="cds_ingress_multiple-
listeners-mesh_tellergateway-vg_self_redirect_http_15001"} 0
```

- Novas estatísticas com "ingress.<protocol>.<port>"

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-
listeners-mesh",appmesh_virtual_gateway="tellergateway-
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_tellergateway-
vg_self_redirect_1111_http_15001"} 0
envoy_cluster_assignment_stale{appmesh_mesh="multiple-
listeners-mesh",appmesh_virtual_gateway="tellergateway-
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_tellergateway-
vg_self_redirect_2222_http_15001"} 0
```

Como instalar o Prometheus

1. Adicione o repositório do EKS ao Helm:

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Instale o App Mesh Prometheus

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system
```

Exemplo de Prometheus

Veja a seguir um exemplo de criação de uma `PersistentVolumeClaim` para armazenamento persistente do Prometheus.

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system \
--set retention=12h \
--set persistentVolumeClaim.claimName=prometheus
```

Tutorial para usar o Prometheus

- App Mesh com EKS—Observabilidade: Prometheus

Para saber mais sobre o Prometheus e o Prometheus com o Amazon EKS

- Documentação do Prometheus
- EKS: métricas do ambiente de gerenciamento com o Prometheus

CloudWatch para o App Mesh

Emissão de estatísticas do Envoy para o CloudWatch a partir do Amazon EKS

Você pode instalar o CloudWatch Agent no seu cluster e configurá-lo para coletar um subconjunto de métricas dos seus proxies. Se você ainda não tem um cluster do Amazon EKS, pode criar um com as etapas em Tutorial: App Mesh com Amazon EKS no GitHub. Você pode instalar um aplicativo de amostra no cluster seguindo o mesmo passo a passo.

Para definir as permissões apropriadas do IAM para seu cluster e instalar o agente, siga as etapas em Instalar o agente do CloudWatch com a coleção de métricas do Prometheus. A instalação padrão contém uma configuração de extração do Prometheus que extrai um subconjunto útil das estatísticas do Envoy. Para obter mais informações, consulte Métricas do Prometheus para o App Mesh.

Para criar um painel personalizado do CloudWatch do App Mesh configurado para exibir as métricas que o agente está coletando, siga as etapas no tutorial Como visualizar suas métricas do Prometheus. Seus gráficos começarão a ser preenchidos com as métricas correspondentes à medida que o tráfego entrar no aplicativo App Mesh.

Como filtrar métricas para o CloudWatch

A extensão de métricas do App Mesh fornece um subconjunto de métricas úteis que fornecem informações sobre os comportamentos dos recursos que você define em sua malha. Como o agente do CloudWatch suporta as métricas de extração do Prometheus, você pode fornecer uma configuração de extração para selecionar as métricas que deseja extrair do Envoy e enviar para o CloudWatch.

Você pode encontrar um exemplo de extração de métricas usando o Prometheus em nosso tutorial Extensão de Métricas.

Exemplo do CloudWatch

Você pode encontrar um exemplo de configuração do CloudWatch em nosso repositório de amostras da AWS.

Instruções para usar o CloudWatch

- Adicione recursos de monitoramento e log em nosso Workshop do App Mesh.
- App Mesh com EKS—Observabilidade: CloudWatch
- Como usar a extensão de métricas do App Mesh no ECS

Extensão de métricas para o App Mesh

O Envoy gera centenas de métricas divididas em algumas dimensões diferentes. As métricas não são diretas na forma como se relacionam com o App Mesh. No caso de serviços virtuais, não há mecanismo para saber com certeza qual serviço virtual está se comunicando com um determinado nó virtual ou gateway virtual.

A extensão de métricas do App Mesh aprimora os proxies Envoy em execução na sua malha. Esse aprimoramento permite que os proxies emitam métricas adicionais que estejam cientes dos recursos que você define. Esse pequeno subconjunto de métricas adicionais ajudará você a entender melhor o comportamento dos recursos que você definiu no App Mesh.

Para ativar a extensão de métricas do App Mesh, defina a variável de ambiente `APPMESH_METRIC_EXTENSION_VERSION` como `1`.

```
APPMESH_METRIC_EXTENSION_VERSION=1
```

Para mais informações sobre as variáveis de configuração do Envoy, consulte Variáveis de configuração do Envoy.

Métricas relacionadas ao tráfego de entrada

- **`ActiveConnectionCount`**
  - `envoy.appmesh.ActiveConnectionCount`: número de conexões TCP ativas.
  - Dimensões: Mesh, VirtualNode, VirtualGateway
- **`NewConnectionCount`**
  - `envoy.appmesh.NewConnectionCount`: número total de conexões TCP.
  - Dimensões: Mesh, VirtualNode, VirtualGateway
- **`ProcessedBytes`**
  - `envoy.appmesh.ProcessedBytes`: total de bytes TCP enviados e recebidos de clientes downstream.
  - Dimensões: Mesh, VirtualNode, VirtualGateway
- **`RequestCount`**
  - `envoy.appmesh.RequestCount`: o número de solicitações HTTP processadas.
  - Dimensões: Mesh, VirtualNode, VirtualGateway
- **`GrpcRequestCount`**
  - `envoy.appmesh.GrpcRequestCount`: o número de solicitações gPRC processadas.
  - Dimensões: Mesh, VirtualNode, VirtualGateway

Métricas relacionadas ao tráfego de saída

Você verá dimensões diferentes em suas métricas de saída com base no fato de elas virem de um nó virtual ou de um gateway virtual.

- **`TargetProcessedBytes`**
  - `envoy.appmesh.TargetProcessedBytes`: total de bytes TCP enviados e recebidos dos destinos upstream do Envoy.
  - Dimensões:
    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **`HTTPCode_Target_2XX_Count`**
  - `envoy.appmesh.HTTPCode_Target_2XX_Count`: o número de solicitações HTTP para um destino upstream do Envoy que resultaram em uma resposta HTTP 2xx.
  - Dimensões:
    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **`HTTPCode_Target_3XX_Count`**
  - `envoy.appmesh.HTTPCode_Target_3XX_Count`: o número de solicitações HTTP para um destino upstream do Envoy que resultaram em uma resposta HTTP 3xx.
  - Dimensões:
    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **`HTTPCode_Target_4XX_Count`**
  - `envoy.appmesh.HTTPCode_Target_4XX_Count`: o número de solicitações HTTP para um destino upstream do Envoy que resultaram em uma resposta HTTP 4xx.
  - Dimensões:
    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **`HTTPCode_Target_5XX_Count`**
  - `envoy.appmesh.HTTPCode_Target_5XX_Count`: o número de solicitações HTTP para um destino upstream do Envoy que resultaram em uma resposta HTTP 5xx.
  - Dimensões:

    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **`RequestCountPerTarget`**
  - `envoy.appmesh.RequestCountPerTarget`: o número de solicitações enviadas a um destino upstream do Envoy.
  - Dimensões:
    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode
- **`TargetResponseTime`**
  - `envoy.appmesh.TargetResponseTime`: o tempo decorrido desde o momento em que uma solicitação é feita a um destino upstream do Envoy até o recebimento da resposta completa.
  - Dimensões:
    - Dimensões do nó virtual: Mesh, VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensões do gateway virtual: Mesh, VirtualGateway, TargetVirtualService, TargetVirtualNode

Datadog para o App Mesh

O Datadog é um aplicativo de monitoramento e segurança para monitoramento, métricas e log de ponta a ponta de aplicativos em nuvem. O Datadog torna sua infraestrutura, aplicativos e aplicativos de terceiros completamente observáveis.

Como instalar o Datadog

- EKS: para configurar o Datadog com o EKS, siga estas etapas nos documentos do Datadog.
- ECS EC2: para configurar o Datadog com o ECS EC2, siga estas etapas na documentação do Datadog.

Para saber mais a respeito do Datadog

- Documentação do Datadog

Rastreamento

> **Important**
>
> Para implementar totalmente o rastreamento, você precisará atualizar seu aplicativo. Para ver todos os dados disponíveis do serviço escolhido, você precisará instrumentar seu aplicativo usando as bibliotecas aplicáveis.

Monitore o App Mesh com AWS X-Ray

AWS X-Ray é um serviço que fornece ferramentas que permitem visualizar, filtrar e obter informações sobre dados coletados a partir das solicitações que seu aplicativo atende. Esses insights ajudam você a identificar problemas e oportunidades para otimizar seu aplicativo. Você pode ver informações detalhadas sobre solicitações e respostas e chamadas downstream que seu aplicativo faz para outros serviços da AWS.

O X-Ray se integra ao App Mesh para gerenciar seus microsserviços Envoy. Os dados de rastreamento do Envoy são enviados para o daemon X-Ray em execução no seu contêiner.

Implemente o X-Ray no código do seu aplicativo usando o guia do SDK específico para sua linguagem.

Ative o rastreamento do X-Ray por meio do App Mesh

- Dependendo do tipo de serviço:
  - ECS: na definição do contêiner proxy Envoy, defina a variável de ambiente `ENABLE_ENVOY_XRAY_TRACING` como `1` e a variável de ambiente `XRAY_DAEMON_PORT` como `2000`.
  - EKS: na configuração do App Mesh Controller, inclua `--set tracing.enabled=true` e `--set tracing.provider=x-ray`.
- Em seu contêiner X-Ray, exponha a porta `2000` e execute como usuário `1337`.

Exemplos de X-Ray

Uma definição de contêiner Envoy para o Amazon ECS

```
      {
        "name": "envoy",
        "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
        "essential": true,
        "environment": [
          {
            "name": "APPMESH_VIRTUAL_NODE_NAME",
            "value": "mesh/myMesh/virtualNode/myNode"
          },
          {
            "name": "ENABLE_ENVOY_XRAY_TRACING",
            "value": "1"
           }
        ],
        "healthCheck": {
          "command": [
            "CMD-SHELL",
            "curl -s http://localhost:9901/server_info | cut -d' ' -f3 | grep -q live"
            ],
           "startPeriod": 10,
           "interval": 5,
           "timeout": 2,
           "retries": 3
      }
```

Como atualizar o controlador App Mesh para Amazon EKS

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
--namespace appmesh-system \
--set region=${AWS_REGION} \
--set serviceAccount.create=false \
--set serviceAccount.name=appmesh-controller \
--set tracing.enabled=true \
--set tracing.provider=x-ray
```

Tutoriais para usar o X-Ray

- Monitor com AWS X-Ray

- App Mesh com Amazon EKS—Observabilidade: X-Ray
- Workshop Rastreamento distribuído com X-Ray no AWS App Mesh

Para saber mais sobre o AWS X-Ray

- Documentação do AWS X-Ray

Solução de problemas do AWS X-Ray com o App Mesh

- Não consigo ver os traços de AWS X-Ray dos meus aplicativos.

Jaeger para o App Mesh com Amazon EKS

O Jaeger é um sistema de rastreamento distribuído de código aberto de ponta a ponta. Ele pode ser usado para criar perfis de redes e para monitoramento. A Jaeger pode ajudar a solucionar problemas de aplicações nativas de nuvem complexas.

Para implementar o Jaeger no código do seu aplicativo, você pode encontrar o guia específico para sua linguagem nas bibliotecas de rastreamento da documentação do Jaeger.

Como instalar o Jaeger usando o Helm

1. Adicione o repositório do EKS ao Helm:

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Instale o App Mesh Jaeger

```
helm upgrade -i appmesh-jaeger eks/appmesh-jaeger \
--namespace appmesh-system
```

Exemplo de Jaeger

Veja a seguir um exemplo de criação de uma `PersistentVolumeClaim` para armazenamento persistente do Jaeger.

```
helm upgrade -i appmesh-controller eks/appmesh-controller \
--namespace appmesh-system \
--set tracing.enabled=true \
--set tracing.provider=jaeger \
--set tracing.address=appmesh-jaeger.appmesh-system \
--set tracing.port=9411
```

Passo a passo para usar o Jaeger

- App Mesh com EKS—Observabilidade: Jaeger

Para saber mais a respeito da Jaeger

- Documentação do Jaeger

Datadog para rastreamento

O Datadog pode ser usado para rastreamento e métricas. Para mais informações e instruções de instalação, encontre o guia específico para o idioma do seu aplicativo na documentação do Datadog.

Ferramentas do App Mesh

O App Mesh oferece aos clientes a capacidade de interagir com suas APIs indiretamente usando ferramentas como:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- Controlador do App Mesh para Kubernetes
- Terraform

App Mesh e AWS CloudFormation

O AWS CloudFormation é um serviço que permite criar um modelo com todos os recursos necessários para sua aplicação e, em seguida, o AWS CloudFormation configura e provisiona os recursos para você. Ele também configurará todas as dependências, para que você possa se concentrar mais na aplicação e menos no gerenciamento de recursos.

Para obter mais informações e exemplos sobre como usar o AWS CloudFormation com o App Mesh, consulte a documentação do AWS CloudFormation.

App Mesh e AWS CDK

O AWS CDK é uma framework de desenvolvimento para definir sua estrutura na nuvem por meio de código usando AWS CloudFormation para provisioná-la. O AWS CDK oferece suporte para várias linguagens de programação, incluindo TypeScript, JavaScript, Python, Java e C#/.Net.

Para obter mais informações sobre como usar o AWS CDK com o App Mesh, consulte a documentação do AWS CDK.

Controlador do App Mesh para Kubernetes

O controlador do App Mesh para Kubernetes ajuda você a gerenciar seus recursos do App Mesh para um cluster Kubernetes e a injetar sidecars em pods. Esse controlador é especificamente para uso com o Amazon EKS e permite que gerenciar os recursos de uma forma nativa do Kubernetes.

Para obter mais informações sobre o controlador App Mesh, consulte a documentação do App Mesh Controller.

Para ver um guia sobre a implementação do App Mesh com o Amazon EKS usando o controlador do App Mesh para Kubernetes, confira o Amazon EKS Workshop.

App Mesh e Terraform

O Terraform é uma ferramenta de software de código aberto para infraestrutura como código. O Terraform pode gerenciar serviços em nuvem usando sua CLI e interagir com APIs usando arquivos de configuração declarativos.

Para saber mais sobre como usar o App Mesh com o Terraform, confira a documentação do Terraform.

Como trabalhar com malhas compartilhadas

Você pode compartilhar suas malhas do App Mesh entre AWS contas usando o AWS Resource Access Manager serviço. Uma malha compartilhada permite que recursos criados por AWS contas diferentes se comuniquem entre si na mesma malha.

Uma AWS conta pode ser proprietária de um recurso de malha, um consumidor de malha ou ambos. Os consumidores podem criar recursos em uma malha que é compartilhada com suas contas. Os proprietários podem criar recursos em qualquer malha que a conta possua. Um proprietário de malha pode compartilhar uma malha com os seguintes tipos de consumidores de malha.

- AWS Contas específicas dentro ou fora de sua organização em AWS Organizations
- Uma unidade organizacional dentro de sua organização em AWS Organizations
- Toda a sua organização em AWS Organizations

Para ver end-to-end um resumo do compartilhamento de uma malha, consulte Visão geral da malha entre contas. GitHub

Concedendo permissões para compartilhar malhas

Ao compartilhar malhas entre contas, há permissões obrigatórias para o principal do IAM compartilhar a malha e permissões necessárias em nível de recurso para a malha em si.

Concedendo permissão para compartilhar uma malha

É necessário um conjunto mínimo de permissões para que um diretor do IAM compartilhe uma malha. Recomendamos usar as políticas `AWSResourceAccessManagerFullAccess` gerenciadas `AWSAppMeshFullAccess` e gerenciadas do IAM para garantir que seus diretores do IAM tenham as permissões necessárias para compartilhar e usar malhas compartilhadas.

Se você usa uma política personalizada do IAM`appmesh:PutMeshPolicy`, as `appmesh:DeleteMeshPolicy` ações`appmesh:GetMeshPolicy`, e são necessárias. Essas são ações do IAM somente com permissão. Se um diretor do IAM não tiver essas permissões concedidas, ocorrerá um erro ao tentar compartilhar a malha usando o AWS RAM serviço.

Para obter mais informações sobre como o AWS Resource Access Manager serviço usa o IAM, consulte Como AWS RAM usar o IAM no Guia AWS Resource Access Manager do usuário.

Concedendo permissões para uma malha

Uma malha compartilhada tem as seguintes permissões.

- Os consumidores podem listar e descrever todos os recursos em uma malha compartilhada com a conta.
- Os proprietários podem listar e descrever todos os recursos em qualquer malha que a conta possua.
- Proprietários e consumidores podem modificar os recursos em uma malha criada pela conta, mas não podem modificar os recursos criados por outra conta.
- Os consumidores podem excluir qualquer recurso em uma malha criada pela conta.
- Os proprietários podem excluir qualquer recurso em uma malha criada por qualquer conta.
- Os recursos do proprietário podem referenciar apenas outros recursos na mesma conta. Por exemplo, um nó virtual só pode fazer referência AWS Cloud Map ou um AWS Certificate Manager certificado que esteja na mesma conta do proprietário do nó virtual.
- Proprietários e consumidores podem conectar um proxy Envoy ao App Mesh como um nó virtual que a conta possui.
- Os proprietários podem criar gateways virtuais e rotas de gateway virtuais.
- Proprietários e consumidores podem listar tags e podem marcar/desmarcar recursos em uma malha criada pela conta. Eles não podem listar tags e marcar/desmarcar recursos em uma malha que não tenha sido criada pela conta.

As malhas compartilhadas usam uma autorização baseada em políticas. Uma malha é compartilhada com um conjunto fixo de permissões. Essas permissões são selecionadas para serem adicionadas a uma política de recursos, e uma política do IAM opcional também pode ser selecionada com base no usuário/perfil do IAM. A interseção de permissões possíveis nessas políticas, menos as permissões explícitas negadas, determina o acesso da entidade principal à malha.

Ao compartilhar uma malha, o AWS Resource Access Manager serviço cria uma política gerenciada chamada `AWSRAMDefaultPermissionAppMesh` e a associa ao seu App Mesh que fornece as seguintes permissões.

- `appmesh:CreateVirtualNode`
- `appmesh:CreateVirtualRouter`
- `appmesh:CreateRoute`

- `appmesh:CreateVirtualService`
- `appmesh:UpdateVirtualNode`
- `appmesh:UpdateVirtualRouter`
- `appmesh:UpdateRoute`
- `appmesh:UpdateVirtualService`
- `appmesh:ListVirtualNodes`
- `appmesh:ListVirtualRouters`
- `appmesh:ListRoutes`
- `appmesh:ListVirtualServices`
- `appmesh:DescribeMesh`
- `appmesh:DescribeVirtualNode`
- `appmesh:DescribeVirtualRouter`
- `appmesh:DescribeRoute`
- `appmesh:DescribeVirtualService`
- `appmesh:DeleteVirtualNode`
- `appmesh:DeleteVirtualRouter`
- `appmesh:DeleteRoute`
- `appmesh:DeleteVirtualService`
- `appmesh:TagResource`
- `appmesh:UntagResource`

Pré-requisitos para compartilhar malhas

Para compartilhar uma malha, você deve atender aos seguintes pré-requisitos.

- Você deve possuir a malha em sua AWS conta. Não é possível compartilhar uma malha que tenha sido compartilhada com você.
- Para compartilhar uma malha com a sua organização ou com uma unidade organizacional nas AWS Organizations, é necessário habilitar o compartilhamento com as AWS Organizations. Para obter mais informações, consulte Habilitar o compartilhamento com o AWS Organizations no Guia do usuário do AWS RAM .

- Seus serviços devem ser implantados em uma Amazon VPC que tenha conectividade compartilhada entre as contas que incluem os recursos de malha que você deseja que comuniquem entre si. Uma forma de compartilhar a conectividade de rede é implantar todos os serviços que você deseja usar em sua malha em uma sub-rede compartilhada. Para mais informações e limitações, consulte Compartilhamento de uma sub-rede.
- Os serviços devem ser descobertos por meio do DNS ou. AWS Cloud Map Para mais informações sobre a descoberta de serviços, consulte Nós virtuais.

Serviços relacionados

O compartilhamento de malha se integra com AWS Resource Access Manager (AWS RAM). AWS RAM é um serviço que permite que você compartilhe seus AWS recursos com qualquer AWS conta ou por meio de AWS Organizations. Com AWS RAM, você compartilha recursos de sua propriedade criando um compartilhamento de recursos. Um compartilhamento de atributos especifica os atributos a serem compartilhados, e os consumidores com os quais compartilhá-los. Os consumidores podem ser AWS contas individuais, unidades organizacionais ou uma organização inteira em AWS Organizations.

Para obter mais informações sobre AWS RAM, consulte o Guia AWS RAM do usuário.

Como compartilhar uma malha

Compartilhar uma malha permite que recursos criados por contas diferentes se comuniquem entre si na mesma malha. Só é possível compartilhar uma malha que você possua. Para compartilhar uma malha, é necessário adicioná-la a um compartilhamento de recursos. Um compartilhamento de recursos é um AWS RAM recurso que permite que você compartilhe seus recursos entre AWS contas. Um compartilhamento de recursos especifica os recursos a serem compartilhados, e os consumidores com os quais compartilhá-los. Ao compartilhar uma malha usando o console do Amazon Linux, você a adiciona a um compartilhamento de recursos existente. Para adicionar a malha a um novo compartilhamento de recursos, crie o compartilhamento de recursos usando o console do AWS RAM.

Se você faz parte de uma organização AWS Organizations e o compartilhamento dentro de sua organização está ativado, os consumidores em sua organização podem receber automaticamente acesso à malha compartilhada. Caso contrário, os consumidores receberão um convite para participar do compartilhamento de recursos e terão acesso à malha compartilhada depois de aceitar o convite.

Você pode compartilhar uma malha que você possui usando o AWS RAM console ou AWS CLI o.

Para compartilhar uma malha que você possui usando o AWS RAM console

Para obter instruções, consulte Criação de um recurso compartilhado no Guia do usuário do AWS RAM . Ao selecionar um tipo de recurso, selecione Malhas e, em seguida, selecione a malha que você deseja compartilhar. Se nenhuma malha estiver listada, crie uma malha primeiro. Para ter mais informações, consulte Criação de uma malha de serviços.

Para compartilhar uma malha que você possui usando o AWS CLI

Use o comando create-resource-share. Para a opção `--resource-arns`, especifique o ARN da malha que você deseja compartilhar.

Como cancelar o compartilhamento de uma malha

Quando você cancela o compartilhamento de uma malha, o App Mesh desativa o acesso adicional à malha por antigos consumidores da malha. No entanto, o App Mesh não exclui os recursos criados pelos consumidores. Depois que o compartilhamento da malha for removido, somente o proprietário da malha poderá acessar e excluir os recursos. O App Mesh impede que a conta que possuía recursos na malha receba informações de configuração depois que a malha não for mais compartilhada. O App Mesh também impede que outras contas com recursos na malha recebam informações de configuração de uma malha não compartilhada. Somente o proprietário da malha pode cancelar o compartilhamento.

Para cancelar o compartilhamento de uma malha de sua propriedade, é necessário removê-la do compartilhamento de recursos. Você pode fazer isso usando o AWS RAM console ou AWS CLI o.

Para cancelar o compartilhamento de uma malha compartilhada que você possui usando o console AWS RAM

Para obter instruções, consulte Como atualizar um recurso compartilhado no Guia do usuário do AWS RAM .

Para cancelar o compartilhamento de uma malha compartilhada que você possui usando o AWS CLI

Use o comando disassociate-resource-share.

Como identificar uma malha compartilhada

Proprietários e consumidores podem identificar malhas compartilhadas e recursos de malha usando o console Amazon Linux e AWS CLI

Para identificar uma malha compartilhada usando o console Amazon Linux

1. Abra o console do App Mesh em https://console.aws.amazon.com/appmesh/.
2. No painel de navegação à esquerda, selecione Malhas. O ID da conta do proprietário de cada malha está listado na coluna Proprietário da malha.
3. No painel de navegação à esquerda, selecione Serviços virtuais, roteadores virtuais ou nós virtuais. Você vê o ID da conta do proprietário da malha e do proprietário do recurso para cada um dos recursos.

Para identificar uma malha compartilhada usando o AWS CLI

Use o comando `aws appmesh list resource`, como `aws appmesh list-meshes`. O comando retorna as malhas de sua propriedade e aquelas compartilhados com você. A `meshOwner` propriedade mostra o ID da AWS conta do `meshOwner` e a `resourceOwner` propriedade mostra o ID da AWS conta do proprietário do recurso. Qualquer comando executado em qualquer recurso de malha retorna essas propriedades.

As tags definidas pelo usuário anexadas a uma malha compartilhada estão disponíveis somente na sua Conta da AWS. Elas não estão disponíveis para as outras contas com as quais a malha é compartilhada. O comando `aws appmesh list-tags-for-resource` para uma malha em outra conta tem acesso negado.

Faturamento e medição

Não há cobranças adicionais pelo compartilhamento de malhas.

Cotas de instâncias

Todas as cotas de uma malha também se aplicam às malhas compartilhadas, independentemente de quem criou os recursos na malha. Somente o proprietário da malha pode solicitar aumentos de cota. Para ter mais informações, consulte Service Quotas do App Mesh. O serviço do AWS Resource Access Manager também tem cotas. Para obter mais informações, consulte Service Quotas.

Serviços integrados da AWS com o App Mesh

O App Mesh funciona com outros serviços da AWS para fornecer soluções adicionais para os desafios do seu negócio. Este tópico identifica os serviços que usam o App Mesh para adicionar funcionalidades ou serviços que o App Mesh usa para executar tarefas.

Índice

Criação de recursos do App Mesh com AWS CloudFormation

O App Mesh é integrado ao AWS CloudFormation, um serviço que ajuda você a modelar e configurar seus recursos da AWS, para que você possa passar menos tempo criando e gerenciando seus recursos e sua infraestrutura. Crie um modelo que descreva todos os recursos da AWS que você deseja, por exemplo, uma malha do App Mesh, e o AWS CloudFormation cuidará do provisionamento e da configuração desses recursos para você.

Ao usar o AWS CloudFormation, você poderá reutilizar seu modelo para configurar seus recursos do App Mesh de forma repetida e consistente. Basta descrever seus recursos uma vez e, depois, provisionar os mesmos recursos repetidamente em várias contas e regiões da AWS.

App Mesh e modelos do AWS CloudFormation

Para provisionar e configurar recursos para o App Mesh e serviços relacionados, é necessário entender os modelos do AWS CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar nas suas pilhas do AWS CloudFormation. Se você não estiver familiarizado com JSON ou YAML, poderá usar o AWS CloudFormation Designer para ajudá-lo a começar a usar os modelos do AWS CloudFormation. Para obter mais informações, consulte O que é o Designer? (O que é o AWS CloudFormation Designer) no Manual do usuário do AWS CloudFormation.

O App Mesh suporta a criação de malhas, rotas, nós virtuais, roteadores virtuais e serviços virtuais em AWS CloudFormation. Para mais informações, incluindo exemplos de modelos JSON e YAML para os recursos do App Mesh, consulte a Referência de tipo de recurso do App Mesh no Guia do usuário do AWS CloudFormation.

Saiba mais sobre o AWS CloudFormation

Para mais sobre o AWS CloudFormation, consulte os seguintes atributos:

- AWS CloudFormation
- Manual do usuário do AWS CloudFormation
- Guia do usuário da interface de linha de comando do AWS CloudFormation

App Mesh em AWS Outposts

O AWS Outposts permite serviços nativos, infraestrutura e modelos operacionais da AWS em instalações on-premises. Em ambientes AWS Outposts, é possível usar as mesmas APIs, ferramentas e infraestrutura da AWS que usa na AWS Cloud. O App Mesh no AWS Outposts é ideal para workloads de baixa latência que precisam ser executadas perto dos dados e aplicações on-premises. Para obter mais informações sobre o AWS Outposts, consulte o Guia do usuário do AWS Outposts.

Pré-requisitos

Veja a seguir os pré-requisitos para usar App Mesh no AWS Outposts:

- Um Outpost deve estar instalado e configurado no datacenter on-premises.
- É necessário ter uma conexão de rede confiável entre o Outpost e a região da AWS.
- A região da AWS do Outpost deve ser compatível com o suporte do AWS App Mesh. Para obter uma lista de regiões suportadas, consulte Endpoints e cotas do AWS App Mesh na Referência geral da AWS.

Limitações

Veja a seguir as limitações de uso do App Mesh no AWS Outposts:

- O AWS Identity and Access Management, o Application Load Balancer, o Network Load Balancer , o Classic Load Balancer e o Amazon Route 53 são executados na região da AWS e não no Outposts. Isto irá aumentar as latências entre esses serviços e os contêineres.

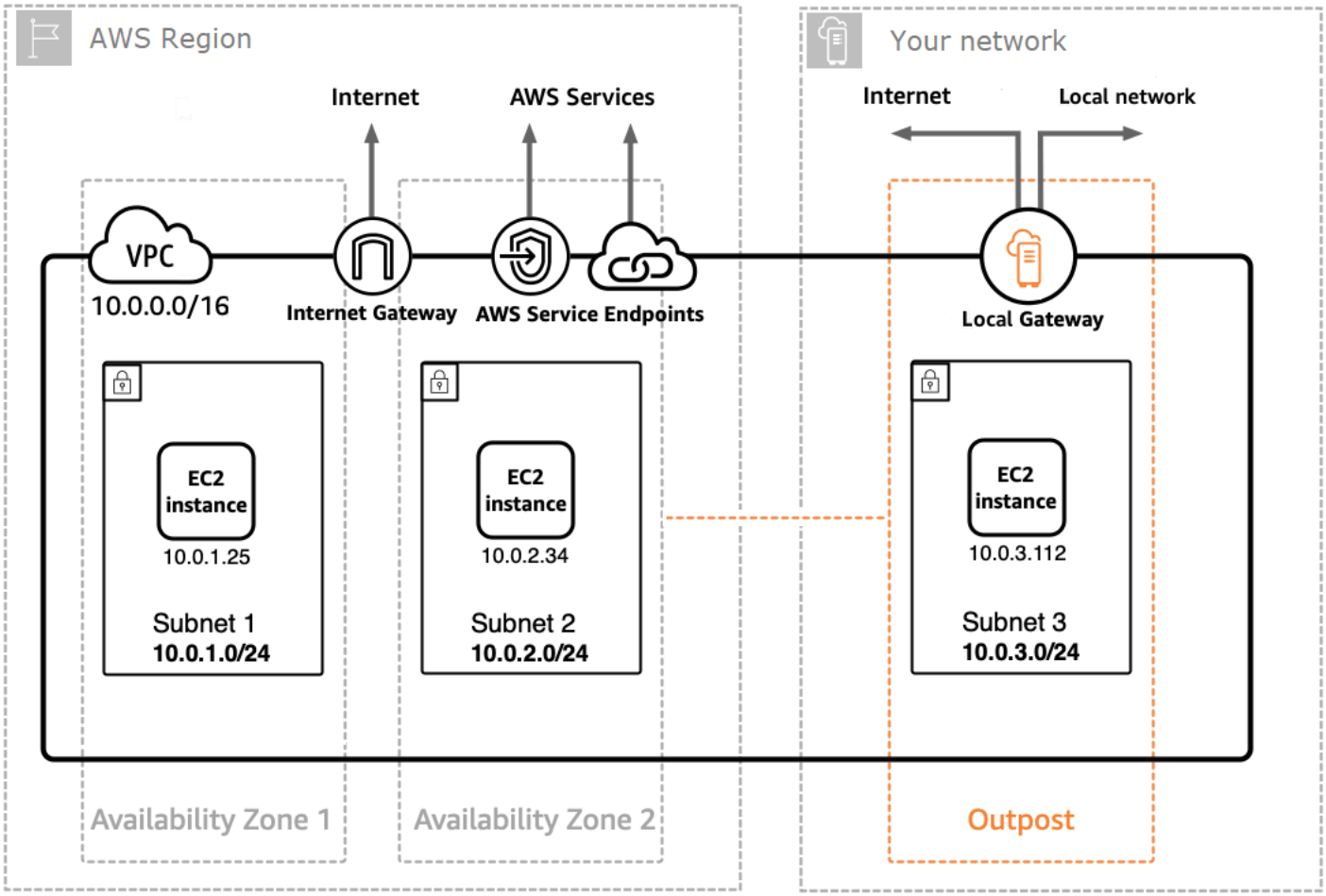
Considerações sobre a conectividade de rede

Veja a seguir as considerações de conectividade de rede para Amazon EKS AWS Outposts:

- Se a conectividade de rede entre o Outpost e sua região da AWS for perdida, os proxies do Envoy do App Mesh continuarão em execução. No entanto, é possível modificar sua malha de serviços até que a conectividade seja restaurada.
- Recomendamos fornecer conectividade confiável, altamente disponível e de baixa latência entre o Outpost e sua região da AWS.

Criando um proxy App Mesh Envoy em um Outpost

Um Outpost é uma extensão de uma região da AWS e é possível estender uma Amazon VPC em uma conta para abranger várias zonas de disponibilidade e qualquer local associado do Outpost. Ao configurar o Outpost, você associa uma sub-rede a ele para estender o ambiente regional da VPC à instalação on-premises. As instâncias em um Outpost aparecem como parte da VPC regional, semelhante a uma zona de disponibilidade com sub-redes associadas.



Para criar um proxy App Mesh Envoy em um Outpost, adicione a imagem do contêiner App Mesh Envoy à tarefa do Amazon ECS ou ao pod Amazon EKS executado em um Outpost. Para obter mais informações, consulte o Amazon Elastic Contêiner Service do AWS Outposts no Guia do desenvolvedor Amazon Elastic Contêiner Service e Amazon Elastic Kubernetes Service do AWS Outposts no Guia do usuário do Amazon EKS.

Práticas recomendadas do App Mesh

Para atingir a meta de zero solicitações com falha durante as implantações planejadas e durante a perda não planejada de alguns hosts, as práticas recomendadas deste tópico implementam a seguinte estratégia:

- Aumente a probabilidade de uma solicitação ser bem-sucedida do ponto de vista do aplicativo usando uma estratégia de repetição padrão segura. Para obter mais informações, consulte Instrumente todas as rotas com novas tentativas.
- Aumente a probabilidade de uma nova solicitação ser bem-sucedida maximizando a probabilidade de que a solicitação repetida seja enviada para um destino real. Para obter mais informações, consulte Ajuste a velocidade de implantação, Aumente a escala horizontalmente antes de reduzi-la. e Implemente verificações de integridade do contêiner.

Para reduzir ou eliminar significativamente as falhas, recomendamos que você implemente as recomendações em todas as práticas a seguir.

Instrumente todas as rotas com novas tentativas

Configure todos os serviços virtuais para usar um roteador virtual e defina uma política de repetição padrão para todas as rotas. Isso atenuará as solicitações com falha selecionando novamente um host e enviando uma nova solicitação. Para políticas de repetição, recomendamos um valor de pelo menos dois para `maxRetries` e especifique as seguintes opções para cada tipo de evento de repetição em cada tipo de rota que suporte o tipo de evento de repetição:

- TCP: `connection-error`
- HTTP e HTTP/2: `stream-error` e `gateway-error`
- gRPC: `cancelled` e `unavailable`

Outros eventos da nova tentativa precisam ser considerados caso a caso, pois podem não ser seguros, como quando a solicitação não é idempotente. Você precisará considerar e testar valores de `maxRetries` e `perRetryTimeout` que façam a compensação adequada entre a latência máxima de uma solicitação (`maxRetries` * `perRetryTimeout`) e o aumento da taxa de sucesso de outras tentativas. Além disso, quando o Envoy tenta se conectar a um endpoint que não está mais presente, você deve esperar que a solicitação consuma o `perRetryTimeout` totalmente. Para

configurar uma política de repetição, consulte Criar uma rota e selecione o protocolo que você deseja rotear.

> **Note**
>
> Se você implementou uma rota em ou após 29 de julho de 2020 e não especificou uma política de repetição, o App Mesh pode ter criado automaticamente uma política de repetição padrão semelhante à política anterior para cada rota criada em ou após 29 de julho de 2020. Para obter mais informações, consulte Política padrão de tentativas de rotas.

Ajuste a velocidade de implantação

Ao usar implantações contínuas, reduza a velocidade geral de implantação. Por padrão, o Amazon ECS configura uma estratégia de implantação de no mínimo 100% de tarefas íntegras e 200% de tarefas totais. Na implantação, isso resulta em dois pontos de grande desvio:

- O tamanho de 100% da frota de novas tarefas pode ser visível para os Envoys antes de estarem prontos para concluir as solicitações (consulte Implemente verificações de integridade do contêiner para mitigações).
- O tamanho de 100% da frota de tarefas antigas pode ser visível para os Envoys enquanto as tarefas estão sendo encerradas.

Quando configurados com essas restrições de implantação, os orquestradores de contêineres podem entrar em um estado em que ocultam simultaneamente todos os destinos antigos e tornam visíveis todos os novos destinos. Como seu plano de dados do Envoy é eventualmente consistente, isso pode resultar em períodos em que o conjunto de destinos visíveis em seu plano de dados divergiu do ponto de vista do orquestrador. Para mitigar isso, recomendamos manter um mínimo de 100% de tarefas íntegras, mas reduzir o total de tarefas para 125%. Isso reduzirá a divergência e melhorará a confiabilidade das novas tentativas. Recomendamos as seguintes configurações para diferentes tempos de execução do contêiner:

Amazon ECS

Se o seu serviço tiver uma contagem desejada de dois ou três, defina `maximumPercent` como 150 por cento. Caso contrário, defina `maximumPercent` como 125 por cento.

Kubernetes

Configure a `update strategy` de sua implantação definindo `maxUnavailable` como 0 por cento e `maxSurge` como 25 por cento. Para mais informações sobre implantações, consulte a documentação de Implantações do Kubernetes.

Aumente a escala horizontalmente antes de reduzi-la.

Tanto o aumento quanto a redução da escala horizontalmente podem resultar em alguma probabilidade de falhas nas solicitações nas novas tentativas. Embora existam recomendações de tarefas que mitiguem o aumento, a única recomendação para a redução é minimizar a porcentagem de tarefas escalonadas a qualquer momento. Recomendamos que você use uma estratégia de implantação que aumente horizontalmente a escala de novas tarefas do Amazon ECS ou implantações do Kubernetes antes de reduzir a escala de tarefas ou implantações antigas. Essa estratégia de escalabilidade mantém mais baixa a sua porcentagem de redução de escala em tarefas ou implantações, mantendo a mesma velocidade. Essa prática se aplica tanto às tarefas do Amazon ECS quanto às implantações do Kubernetes.

Implemente verificações de integridade do contêiner

No cenário de aumento de escala, os contêineres em uma tarefa do Amazon ECS podem ficar fora de ordem e podem não responder inicialmente. Recomendamos as seguintes sugestões para diferentes tempos de execução do contêiner:

Amazon ECS

Para mitigar isso, recomendamos o uso de verificações de integridade de contêiner e ordenação de dependências de contêiner para garantir que o Envoy esteja funcionando e íntegro antes do início de qualquer contêiner que exija conectividade de rede de saída. Para configurar corretamente um contêiner de aplicativo e um contêiner Envoy em uma definição de tarefa, consulte Dependência de contêiner.

Kubernetes

Nenhuma, porque os testes de atividade e prontidão do Kubernetes não estão sendo considerados no registro e cancelamento do registro de instâncias do AWS Cloud Map no controlador do App Mesh para Kubernetes. Para obter mais informações, consulte o problema n.º 132 do GitHub.

Segurança em AWS App Mesh

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de uma arquitetura de data center e rede criada 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 isto como segurança da nuvem e segurança na nuvem.

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços na AWS nuvem. AWS também fornece serviços que você pode usar com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos compliance programs AWS. Para saber mais sobre os programas de conformidade que se aplicam ao AWS App Mesh, consulte AWS Services in Scope by Compliance Program. O App Mesh é responsável por fornecer configurações seguras aos proxies locais, incluindo segredos como chaves privadas de certificados TLS.
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo:
  - A confidencialidade dos dados, os requisitos da sua empresa e as leis e regulamentações aplicáveis.
  - A configuração de segurança do plano de dados do App Mesh, incluindo a configuração dos grupos de segurança que permitem que o tráfego passe entre serviços em sua VPC.
  - A configuração dos seus recursos computacionais associados ao App Mesh.
  - As políticas do IAM associadas aos seus recursos computacionais e qual configuração elas podem recuperar do ambiente de gerenciamento do App Mesh.

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o App Mesh. Os tópicos a seguir mostram como configurar o App Mesh 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 Mesh.

Princípio de segurança do App Mesh

> Os clientes devem ser capazes de ajustar a segurança conforme a extensão necessária. A plataforma não deve impedir que sejam mais seguros. Os recursos da plataforma são seguros por padrão.

Tópicos

Transport Layer Security (TLS)

No App Mesh, o Transport Layer Security (TLS) criptografa a comunicação entre os proxies Envoy implantados em recursos computacionais que são representados no App Mesh por endpoints de malha, como e Nós virtuais e Gateways virtuais. O proxy negocia e encerra o TLS. Quando o proxy é implantado com um aplicação, o código do aplicação não é responsável por negociar uma sessão TLS. O proxy negocia o TLS em nome da sua aplicação.

O App Mesh permite que você forneça o certificado TLS ao proxy das seguintes formas:

- Um certificado privado do AWS Certificate Manager (ACM) emitido por um AWS Private Certificate Authority (AWS Private CA).
- Um certificado armazenado no sistema de arquivos local de um nó virtual emitido por sua própria autoridade de certificação (CA)
- Um certificado fornecido por um endpoint do Secrets Discovery Service (SDS) pelo soquete de domínio Unix local.

O Autorização do Envoy Proxy deve ser habilitado para o proxy Envoy implantado representado por um endpoint de malha. Recomendamos que, ao habilitar a autorização de proxy, restrinja o acesso somente ao endpoint de malha para o qual você está habilitando a criptografia.

Requisitos de certificado

Um dos nomes alternativos de assunto (SANs) no certificado deve corresponder a critérios específicos, dependendo de como o serviço real representado por um endpoint de malha é descoberto.

- DNS: um dos SANs do certificado deve corresponder ao valor fornecido nas configurações de descoberta de serviços DNS. Para uma aplicação com o nome de descoberta de serviços *mesh-endpoint.apps.local*, você pode criar um certificado correspondente a esse nome ou um certificado com o curinga *.*apps.local*.
- AWS Cloud Map— Uma das SANs certificadas deve corresponder ao valor fornecido nas configurações de descoberta do AWS Cloud Map serviço usando o formato*service-name.namespace-name*. Para um aplicativo com as configurações de descoberta de AWS Cloud Map serviço de ServiceName e *mesh-endpoint* *apps.local* NamespaceName, você pode criar um certificado que corresponda ao *mesh-endpoint.apps.local* nome ou um certificado com o caractere curinga. *.*apps.local*.

Para ambos os mecanismos de descoberta, se nenhum dos SANs do certificado corresponder às configurações de descoberta de serviços do DNS, a conexão entre os Envoys falhará com a mensagem de erro a seguir, conforme observado no Envoy do cliente.

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

Certificados de autenticação TLS

O App Mesh oferece suporte a várias fontes de certificados ao usar a autenticação TLS.

AWS Private CA

O certificado deve ser armazenado no ACM na mesma região e conta da AWS do endpoint de malha que usará o certificado. O certificado da CA não precisa estar na mesma AWS conta, mas ainda precisa estar na mesma região do endpoint de malha. Se você não tiver um CA privada da AWS, deverá criar um antes de solicitar um certificado dele. Para obter mais informações sobre como solicitar um certificado de um existente AWS Private CA usando o ACM, consulte Solicitar um certificado privado. O certificado não pode ser público.

As CAs privadas que você usa para políticas de cliente TLS devem ser CAs de usuário raiz.

Para configurar um nó virtual com certificados e CAs de AWS Private CA, o principal (como um usuário ou uma função) que você usa para chamar o App Mesh deve ter as seguintes permissões do IAM:

- Para qualquer certificado que você adiciona à configuração TLS de um receptor, a entidade principal deve ter a permissão `acm:DescribeCertificate`.
- Para todas as CAs configuradas em uma política de cliente TLS, a entidade principal deve ter a permissão `acm-pca:DescribeCertificateAuthority`.

> **Important**
>
> Compartilhar CAs com outras contas pode dar a essas contas privilégios não intencionais à CA. Recomendamos o uso de políticas baseadas em recursos para restringir o acesso a apenas `acm-pca:DescribeCertificateAuthority` e `acm-pca:GetCertificateAuthorityCertificate` para contas que não precisam emitir certificados da CA.

Você pode adicionar essas permissões à uma política do IAM existente que está anexada a uma entidade principal ou criar uma nova entidade principal e uma nova política e anexar a política à entidade principal. Para obter mais informações, consulte Edição de políticas do IAM, criação de políticas do IAM e adição de permissões de identidade do IAM.

> **Note**
>
> Você paga uma taxa mensal pela operação de cada um AWS Private CA até excluí-lo. Você paga também pelos certificados privados que emite a cada mês e pelos certificados privados que exporta. Para obter mais informações, consulte Preços do AWS Certificate Manager.

Ao ativar a autorização de proxy para o Envoy Proxy que um endpoint de malha representa, o perfil do IAM que você usa deve receber as seguintes permissões do IAM:

- Para qualquer certificado configurado no receptor de um nó virtual, a função deve ter a permissão `acm:ExportCertificate`.
- Para todas as CAs configuradas em uma política de cliente TLS, a função deve ter a permissão`acm-pca:GetCertificateAuthorityCertificate`.

Sistema de arquivos

Você pode distribuir certificados para o Envoy usando o sistema de arquivos. É possível fazer isso disponibilizando a cadeia de certificados e a chave privada correspondente disponível no caminho do arquivo. Dessa forma, esses recursos podem ser acessados pelo proxy do Envoy sidecar.

Envoy's Secret Discovery Service (SDS)

O Envoy busca segredos, como certificados TLS, de um endpoint específico por meio do protocolo Secrets Discovery. Para obter mais informações sobre esse protocolo, consulte a documentação do SDS do Envoy.

O App Mesh configura o proxy Envoy para usar um soquete de domínio Unix Local ao proxy como o endpoint do Secret Discovery Service (SDS) quando o SDS é usado como a fonte para seus certificados e cadeias de certificados. Você pode configurar o caminho para esse endpoint usando a variável de ambiente `APPMESH_SDS_SOCKET_PATH`.

> **Important**
>
> O Secrets Discovery Service local usando o soquete de domínio Unix é compatível com o proxy App Mesh Envoy versão 1.15.1.0 e posterior.
> O App Mesh oferece suporte ao protocolo V2 SDS usando gRPC.

Integração com o SPIFFE Runtime Environment (SPIRE)

É possível usar qualquer implementação secundária da API do SDS, incluindo cadeias de ferramentas existentes, como o SPIFFE Runtime Environment (SPIRE). O SPIRE foi projetado para permitir a implantação da autenticação de TLS mútuo entre várias workloads em sistemas distribuídos. Ele atesta a identidade das workloads em tempo de execução. O SPIRE também fornece chaves e certificados específicos para workloads, de curta duração e com rotação automática diretamente para workloads.

Você deve configurar o atendente SPIRE como um provedor de SDS para o Envoy. Permita que ele forneça diretamente ao Envoy o material essencial necessário para fornecer autenticação de TLS mútuo. Execute agentes SPIRE em sidecars próximos aos proxies Envoy. O atendente se encarrega de regenerar as chaves e certificados de curta duração, conforme necessário. O atendente atesta o Envoy e determina quais identidades de serviço e certificados de CA devem

ser disponibilizados ao Envoy quando o Envoy se conecta ao servidor SDS exposto pelo agente SPIRE.

Durante esse processo, as identidades de serviço e os certificados CA são alternados e as atualizações são transmitidas de volta ao Envoy. O Envoy as aplica imediatamente às novas conexões sem interrupções ou tempo de inatividade e sem que as chaves privadas cheguem ao sistema de arquivos.

Como o App Mesh configura os Envoys para negociar o TLS

O App Mesh usa a configuração do endpoint de malha do cliente e do servidor ao determinar como configurar a comunicação entre os Envoys em uma malha.

Com as políticas do cliente

Quando uma política de cliente impõe o uso do TLS e uma das portas na política do cliente corresponde à porta da política do servidor, a política do cliente é usada para configurar o contexto de validação do TLS do cliente. Por exemplo, se a política de cliente de um gateway virtual corresponder à política de servidor de um nó virtual, a negociação de TLS será tentada entre os proxies usando as configurações definidas na política de cliente do gateway virtual. Se a política do cliente não corresponder à porta da política do servidor, o TLS entre os proxies poderá ou não ser negociado, dependendo das configurações de TLS da política do servidor.

Sem políticas de cliente

Se o cliente não tiver configurado uma política de cliente ou se a política do cliente não corresponder à porta do servidor, o App Mesh usará o servidor para determinar se deve ou não negociar o TLS do cliente e como. Por exemplo, se um gateway virtual não tiver especificado uma política de cliente e um nó virtual não tiver configurado o encerramento do TLS, o TLS não será negociado entre os proxies. Se um cliente não tiver especificado uma política de cliente correspondente e um servidor tiver sido configurado com os modos TLS `STRICT` ou `PERMISSIVE`, os proxies serão configurados para negociar o TLS. Dependendo de como os certificados foram fornecidos para o encerramento do TLS, o seguinte comportamento adicional se aplica.

- Certificados TLS gerenciados pelo ACM: quando um servidor configura o encerramento do TLS usando um certificado gerenciado pelo ACM, o App Mesh configura automaticamente os clientes para negociar o TLS e validar o certificado em relação à CA do usuário raiz ao qual o certificado está vinculado.

- Certificados TLS baseados em arquivos: quando um servidor configura o encerramento do TLS usando um certificado do sistema de arquivos local do proxy, o App Mesh configura automaticamente um cliente para negociar o TLS, mas o certificado do servidor não é validado.

Nomes alternativos do assunto

Você pode opcionalmente especificar uma Subject Alternative Names (SANs) nos quais confiar. Os SANs devem estar no formato FQDN ou URI. Se os SANs forem fornecidos, o Envoy verifica se o Nome alternativo do assunto do certificado apresentado corresponde a um dos nomes dessa lista.

Se você não especificar SANs no endpoint da malha de encerramento, o proxy Envoy para esse nó não verificará a SAN em um certificado de cliente do mesmo nível. Se você não especificar SANs no endpoint da malha de origem, a SAN no certificado fornecido pelo endpoint de encerramento deverá corresponder à configuração de descoberta do serviço de endpoints da malha.

Para obter mais informações, consulte App Mesh TLS: requisitos de certificado.

> **Important**
>
> Você só pode usar SANs curinga se a política do cliente para TLS estiver definida como `not enforced`. Se a política de cliente para o nó virtual ou gateway virtual do cliente estiver configurada para aplicar o TLS, ela não poderá aceitar um SAN curinga.

Verifique a criptografia

Depois de habilitar o TLS, você pode consultar o proxy Envoy para confirmar se a comunicação está criptografada. O proxy Envoy emite estatísticas sobre recursos que podem ajudá-lo a entender se sua comunicação TLS está trabalhando corretamente. Por exemplo, o proxy Envoy registra estatísticas sobre o número de handshakes TLS bem-sucedidos que ele negociou para um endpoint de malha especificado. Determine quantos handshakes TLS bem-sucedidos ocorreram para um endpoint de malha nomeado *my-mesh-endpoint* com o comando a seguir.

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep ssl.handshake
```

Na saída retornada do exemplo a seguir, houve três handshakes para o endpoint de malha, portanto, a comunicação é criptografada.

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

O proxy Envoy também emite estatísticas quando a negociação do TLS está falhando. Determine se houve erros de TLS no endpoint da malha.

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep -e "ssl.*\(fail\|error
\)"
```

Na saída retornada do exemplo, não houve nenhum erro em várias estatísticas, então a negociação do TLS foi bem-sucedida.

```
listener.0.0.0.0_15000.ssl.connection_error: 0
listener.0.0.0.0_15000.ssl.fail_verify_cert_hash: 0
listener.0.0.0.0_15000.ssl.fail_verify_error: 0
listener.0.0.0.0_15000.ssl.fail_verify_no_cert: 0
listener.0.0.0.0_15000.ssl.ssl.fail_verify_san: 0
```

Para obter mais informações sobre as estatísticas do Envoy TLS, consulte Estatísticas do receptor Envoy.

Renovação de certificado

AWS Private CA

Quando você renova um certificado com o ACM, o certificado renovado será distribuído automaticamente aos seus proxies conectados dentro de 35 minutos após a conclusão da renovação. Recomendamos usar a renovação gerenciada para renovar automaticamente os certificados perto do final do período de validade. Para obter mais informações, consulte Renovação gerenciada para certificados emitidos pela Amazon do ACM no Guia do AWS Certificate Manager usuário.

Seu próprio certificado

Ao usar um certificado do sistema de arquivos local, o Envoy não recarregará automaticamente o certificado quando ele for alterado. Você pode reiniciar ou reimplantar o processo Envoy para carregar um novo certificado. É possível também colocar um certificado mais novo em um caminho de arquivo diferente e atualizar a configuração do nó virtual ou do gateway com esse caminho de arquivo.

Configure as cargas de trabalho do Amazon ECS para usar a autenticação TLS com AWS App Mesh

Você pode configurar sua malha para usar a autenticação TLS. Certifique-se de que os certificados estejam disponíveis para os sidecars do proxy do Envoy que você adiciona aos workloads. Você pode anexar um volume EBS ou EFS ao seu sidecar do Envoy ou pode armazenar e recuperar certificados do AWS Secrets Manager.

- Se você estiver usando distribuição de certificados baseada em arquivos, conecte um volume do EBS ou do EFS ao seu sidecar do Envoy. Certifique-se de que o caminho para o certificado e a chave privada corresponda ao que está configurado em AWS App Mesh.
- Se você estiver usando a distribuição baseada em SDS, adicione um sidecar que implemente a API SDS do Envoy com acesso ao certificado.

> **Note**
>
> O SPIRE não é compatível com o Amazon ECS.

Configure as cargas de trabalho do Kubernetes para usar a autenticação TLS com AWS App Mesh

Você pode configurar o AWS App Mesh Controller for Kubernetes para habilitar a autenticação TLS para back-ends e ouvintes do serviço de nó virtual e gateway virtual. Certifique-se de que os certificados estejam disponíveis para os sidecars do proxy do Envoy que você adiciona aos workloads. É possível ver um exemplo para cada tipo de distribuição na seção passo a passo da Mutual TLS Authentication.

- Se você estiver usando distribuição de certificados baseada em arquivos, conecte um volume do EBS ou do EFS ao seu sidecar do Envoy. Certifique-se de que o caminho para o certificado e a chave privada corresponda ao configurado no controlador. Como alternativa, é possível usar um Kubernetes Secret montado no sistema de arquivos.
- Se estiver usando a distribuição baseada em SDS, deverá configurar um provedor de SDS local de nó que implemente a API SDS do Envoy. O Envoy chegará até lá via UDS. Para habilitar o suporte a mTLS baseado em SDS no AppMesh controlador EKS, defina o `enable-sds` sinalizador como `true` e forneça o caminho UDS do provedor de SDS local para o controlador por meio

do sinalizador. `sds-uds-path` Se usa o Helm, configure-os como parte da instalação do seu controlador:

```
--set sds.enabled=true
```

> **Note**
>
> Não é possível usar o SPIRE para distribuir seus certificados se estiver usando o Amazon Elastic Kubernetes Service (Amazon EKS) no modo Fargate.

Autenticação TLS mútua

A autenticação TLS (Transport Layer Security) mútua é um componente opcional do TLS que oferece autenticação bidirecional entre pares. A autenticação de TLS mútuo adiciona uma camada de segurança sobre o TLS e permite que seus serviços verifiquem o cliente que está fazendo a conexão.

O cliente na relação cliente-servidor também fornece um certificado X.509 durante o processo de negociação da sessão. O servidor usa esse certificado para identificar e autenticar o cliente. Esse processo ajuda a verificar se o certificado foi emitido por uma autoridade de certificação (CA) confiável e se o certificado é válido. Ele também usa o Nome Alternativo do Assunto (SAN) no certificado para identificar o cliente.

Você pode ativar a autenticação TLS mútua para todos os protocolos suportados pelo AWS App Mesh. Eles são TCP, HTTP/1.1, HTTP/2, gRPC.

> **Note**
>
> Usando o App Mesh, você pode configurar a autenticação de TLS mútuo para comunicações entre proxies Envoy de seus serviços. No entanto, as comunicações entre seus aplicativos e os proxies do Envoy não são criptografadas.

Certificados de autenticação de TLS mútuo

AWS App Mesh suporta duas fontes de certificado possíveis para autenticação TLS mútua. Os certificados de cliente em uma política de cliente TLS e a validação do servidor em uma configuração TLS de receptor podem ser obtidos de:

- Sistema de arquivos: certificados do sistema de arquivos local do proxy Envoy que está sendo executado. Para distribuir certificados para o Envoy, você precisa fornecer caminhos de arquivo para a cadeia de certificados e a chave privada para a API do App Mesh.
- Envoy's Secret Discovery Service (SDS) — B ring-your-own sidecars que implementam o SDS e permitem que certificados sejam enviados ao Envoy. Eles incluem o SPIFFE Runtime Environment (SPIRE).

> **Important**
>
> O App Mesh não armazena os certificados ou as chaves privadas que são usados para autenticação de TLS mútuo. Em vez disso, o Envoy os armazena na memória.

Configuração de endpoints de malha

Configure a autenticação de TLS mútuo para seus endpoints de malha, como nós virtuais ou gateways. Esses endpoints fornecem certificados e especificam autoridades confiáveis.

Para fazer isso, você precisa provisionar certificados X.509 para o cliente e para o servidor e definir explicitamente certificados de autoridade confiável no contexto de validação do encerramento do TLS e do TLS de origem.

Confiança dentro de uma malha

Os certificados do lado do servidor são configurados nos receptores do nó virtual (encerramento do TLS) e os certificados do lado do cliente são configurados nos back-ends do serviço de nós virtuais (TLS de origem). Como alternativa a essa configuração, você pode definir uma política de cliente padrão para todos os back-ends de serviços de um nó virtual e, se necessário, substituir essa política para back-ends específicos, conforme necessário. Os gateways virtuais só podem ser configurados com uma política de cliente padrão que se aplique a todos os seus back-ends.

Você pode configurar a confiança em diferentes malhas ativando a autenticação de TLS mútuo para tráfego de entrada nos gateways virtuais de ambas as malhas.

Confiança fora de uma malha

Especifique certificados do lado do servidor no receptor do Gateway Virtual para o encerramento do TLS. Configure o serviço externo que se comunica com seu Gateway Virtual para apresentar certificados do lado do cliente. Os certificados devem ser derivados de uma das mesmas autoridades de certificação (CAs) que os certificados do lado do servidor usam no receptor do Gateway Virtual para o TLS de origem.

Migração de serviços para a autenticação de TLS mútuo

Siga essas diretrizes para manter a conectividade ao migrar seus serviços existentes no App Mesh para a autenticação de TLS mútuo.

Migração de serviços que se comunicam por texto sem formatação

1. Ative o modo `PERMISSIVE` para a configuração TLS no endpoint do servidor. Esse modo permite que o tráfego de texto sem formatação se conecte ao endpoint.
2. Configure a autenticação de TLS mútuo em seu servidor, especificando o certificado do servidor, a cadeia de confiança e, opcionalmente, os SANs confiáveis.
3. Confirme se a comunicação está acontecendo por meio de uma conexão TLS.
4. Configure a autenticação de TLS mútuo em seus clientes, especificando o certificado de cliente, a cadeia de confiança e, opcionalmente, os SANs confiáveis.
5. Ative o modo `STRICT` para a configuração TLS no servidor.

Migração de serviços que se comunicam por TLS

1. Configure a autenticação de TLS mútuo em seus clientes, especificando o certificado de cliente e, opcionalmente, os SANs confiáveis. O certificado do cliente não é enviado para seu back-end até que o servidor de back-end o solicite.
2. Configure a autenticação de TLS mútuo em seu servidor, especificando a cadeia de confiança e, opcionalmente, os SANs confiáveis. Para isso, seu servidor solicita um certificado de cliente.

Verificação da autenticação de TLS mútuo

Você pode consultar a documentação de Transport Layer Security: Verificação de criptografia para ver exatamente como o Envoy emite estatísticas relacionadas ao TLS. Para a autenticação de TLS mútuo, você deve inspecionar as seguintes estatísticas:

- `ssl.handshake`
- `ssl.no_certificate`
- `ssl.fail_verify_no_cert`
- `ssl.fail_verify_san`

Juntos, os dois exemplos de estatísticas a seguir mostram que as conexões TLS bem-sucedidas que terminaram no nó virtual foram todas originadas de um cliente que forneceu um certificado.

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

```
listener.0.0.0.0_15000.ssl.no_certificate: 0
```

O próximo exemplo de estatística mostra que as conexões de um nó cliente virtual (ou gateway) com um nó virtual de back-end falharam. O SAN (Nome alternativo do assunto) apresentado no certificado do servidor não corresponde a nenhum dos SANs confiados pelo cliente.

```
cluster.cds_egress_my-mesh_my-backend-node_http_9080.ssl.fail_verify_san: 5
```

Tutoriais de autenticação de TLS mútuo do App Mesh

- Tutorial da autenticação de TLS mútuo: este passo a passo descreve como você pode usar a CLI do App Mesh para criar um aplicativo de cor com autenticação de TLS mútuo.
- Tutorial baseado em TLS SDS mútuo do Amazon EKS: este passo a passo mostra como você pode usar a autenticação mútua baseada em TLS SDS com o Amazon EKS e o SPIFFE Runtime Environment (SPIRE).
- Tutorial baseado em arquivo TLS mútuo do Amazon EKS: este passo a passo mostra como você pode usar a autenticação mútua baseada em arquivo TLS com o Amazon EKS e o SPIFFE Runtime Environment (SPIRE).

Como AWS App Mesh funciona com o IAM

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 (fazer login) e autorizado (ter permissões) para usar recursos do App Mesh. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz no App Mesh.

Usuário do serviço: se você usa o serviço App Mesh para realizar o trabalho, o administrador fornece as credenciais e as permissões necessárias. À medida que usar mais recursos do App Mesh para realizar seu trabalho, você poderá precisar de permissões adicionais. Entender como o acesso é gerenciado pode ajudá-lo a solicitar as permissões corretas ao seu administrador. Se não for possível acessar um atributo no App Mesh, consulte Solução de problemas AWS App Mesh de identidade e acesso.

Administrador do serviço: se você for o responsável pelos recursos do App Mesh na empresa, provavelmente terá acesso total ao App Mesh. Cabe a você determinar quais recursos e funcionalidades do App Mesh os usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise

as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como a empresa pode usar o IAM com o App Mesh, consulte Como AWS App Mesh funciona com o IAM.

Administrador do IAM: se você for um administrador do IAM, talvez queira saber detalhes sobre como é possível criar políticas para gerenciar o acesso ao App Mesh. Para visualizar exemplos de políticas baseadas em identidade do App Mesh que podem ser usadas no IAM, consulte AWS App Mesh exemplos de políticas baseadas em identidade.

Autenticando com identidades

A autenticação é a forma 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 uma 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 assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independentemente do método de autenticação usado, também pode ser exigido que você forneça 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 . Usar a autenticação multifator (MFA) na AWS 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 pode 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.

Grupos e usuários 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, recomendamos depender de 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, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne 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 IAMAdmins e atribuir 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 Quando criar um usuário do IAM (em vez de uma função) 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. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. 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 o uso de perfis, consulte Usar perfis do IAM 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, você pode 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 pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do usuário do IAM. Se você usar o IAM Identity Center, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões noAWS IAM Identity Center Guia do usuário do .
- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode assumir 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 (um 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 saber a diferença entre funções e políticas baseadas em recurso para acesso entre contas, consulte Como as funções do IAMdiferem das políticas baseadas em recurso 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 aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de 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 Encaminhar sessões de acesso.

  - Função de serviço: uma função de serviço é um perfil do IAMque um serviço assume para realizar 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 assumir o perfil para 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 pode editar as permissões para perfis vinculados ao serviço.
- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Usar uma função do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar os perfis do IAM, consulte Quando criar uma função do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciamento do 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 funções não têm permissões. Para conceder aos usuários permissão para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do

IAM. O administrador pode então adicionar as políticas do IAM a 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 funções podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda mais como políticas em linha ou políticas gerenciadas. As políticas embutidas são anexadas diretamente a um único usuário, grupo ou função. 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 recurso 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 recurso ao qual a política está anexada, a política define quais ações uma entidade principal especificada pode executar nesse recurso 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 que estão 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 entidades principais (membros, usuários ou funções da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Para saber mais sobre ACLs, consulte 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 e dos seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou a função 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 vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte How SCPs work (Como os SCPs funcionam) noAWS Organizations Guia do usuário do .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para uma função 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 recurso. 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 AWS App Mesh funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao App Mesh, você precisa saber quais recursos do IAM estão disponíveis para uso com o App Mesh. Para ter uma visão geral de como o App Mesh e outros AWS serviços funcionam com o IAM, consulte AWS Serviços que funcionam com o IAM no Guia do usuário do IAM.

Tópicos

- Políticas baseadas em identidade do App Mesh
- Políticas baseadas em recursos do App Mesh
- Autorização baseada em tags do App Mesh
- Perfis do IAM do App Mesh

Políticas baseadas em identidade do App Mesh

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. O App Mesh 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 você pode usar 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. Há também algumas operações que 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 políticas no App Mesh usam o seguinte prefixo antes da ação: `appmesh:`. Por exemplo, para conceder permissão para alguém listar malhas de uma conta com a operação da API `appmesh:ListMeshes`, inclua a ação `appmesh:ListMeshes` na política da pessoa. As instruções de política devem incluir um elemento `Action` ou `NotAction`.

Para especificar várias ações em uma única declaração, separe-as com vírgulas, conforme a seguir.

```
"Action": [
      "appmesh:ListMeshes",
      "appmesh:ListVirtualNodes"
]
```

Você também pode especificar várias ações utilizando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a ação a seguir:

```
"Action": "appmesh:Describe*"
```

Para ver uma lista de ações do App Mesh, consulte Ações definidas pelo AWS App Mesh no Manual do usuário do IAM.

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 `Resource` de política JSON especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou um elemento `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem suporte a um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem suporte a permissões em nível de recurso, como operações de listagem, use um asterisco (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

O recurso App Mesh do `mesh` tem o seguinte ARN.

```
arn:${Partition}:appmesh:${Region}:${Account}:mesh/${MeshName}
```

Para obter mais informações sobre o formato dos ARNs, consulte Amazon Resource Names (ARNs) e AWS Service Namespaces.

Por exemplo, para especificar o mesh chamado *`apps`* na *`Region-code`* da região em sua instrução, use o seguinte ARN.

```
arn:aws:appmesh:Region-code:111122223333:mesh/apps
```

Para especificar todas as instâncias que pertencem a uma conta específica, use o caractere curinga (*).

```
"Resource": "arn:aws:appmesh:Region-code:111122223333:mesh/*"
```

Algumas ações do App Mesh, como as ações para a criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, você deve utilizar o caractere curinga (*).

```
"Resource": "*"
```

Muitas ações da App Mesh API envolvem vários recursos. Por exemplo, a `CreateRoute` cria uma rota com um destino de nó virtual, portanto, um usuário do IAM deve ter permissões para usar a rota e o nó virtual. Para especificar vários recursos em uma única instrução, separe os ARNs com vírgulas.

```
"Resource": [
      "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualRouter/serviceB/route/
*",
      "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualNode/serviceB"
]
```

Para ver uma lista de tipos de recurso do App Mesh e seus ARNs, consulte Recursos definidos pelo AWS App Mesh do Guia do usuário do IAM. Para saber com quais ações é possível especificar o ARN de cada recurso, consulte Ações definidas pelo AWS App Mesh.

Chaves de condição

O App Mesh oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição globais da AWS , consulte Chaves de contexto de condição globais da AWS no Guia do usuário do IAM. Para ver uma lista das chaves de condição globais que o App Mesh suporta, consulte Chaves de condição para AWS App Mesh no Guia do usuário do IAM. Para saber com quais ações e recursos você pode usar com uma chave de condição, consulte Ações definidas por AWS App Mesh.

Exemplos

Para visualizar exemplos de políticas baseadas em identidade do App Mesh, consulte AWS App Mesh exemplos de políticas baseadas em identidade.

Políticas baseadas em recursos do App Mesh

O App Mesh não oferece suporte a políticas baseadas em recursos. No entanto, se você usar o serviço AWS Resource Access Manager (AWS RAM) para compartilhar uma malha entre AWS serviços, uma política baseada em recursos será aplicada à sua malha pelo AWS RAM serviço. Para ter mais informações, consulte Concedendo permissões para uma malha.

Autorização baseada em tags do App Mesh

Você pode anexar tags a recursos do App Mesh ou passar tags em uma solicitação ao App Mesh. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as chaves de condição `appmesh:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`. Para obter mais informações sobre como marcar recursos do App Mesh, consulte Como marcar AWS recursos.

Para visualizar um exemplo de política baseada em identidade para limitar o acesso a um atributo baseado em tags desse atributo, consulte Criação de malhas do App Mesh com tags restritas.

Perfis do IAM do App Mesh

Uma função do IAM é uma entidade dentro da sua AWS conta que tem permissões específicas.

Usar credenciais temporárias com o App Mesh

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de AWS STS API, como AssumeRoleou GetFederationToken.

O App Mesh oferece suporte ao uso de credenciais temporárias.

Funções vinculadas 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 na 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 Mesh é compatível com funções vinculadas ao serviço. Para obter detalhes sobre como criar ou gerenciar funções vinculadas ao serviço do App Mesh, consulte Usar funções vinculadas ao serviço do App Mesh.

Funções 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 indica que um administrador do IAM pode alterar as permissões para essa função. Porém, fazer isso pode alterar a funcionalidade do serviço.

O App Mesh não suporta funções de serviço.

AWS App Mesh exemplos de políticas baseadas em identidade

Por padrão, os usuários e os perfis do IAM não têm permissão para criar ou modificar recursos do App Mesh. 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 Guia do usuário do IAM.

Tópicos

- Melhores práticas de política
- Usar o console do App Mesh
- Permitir que os usuários visualizem suas próprias permissões
- Crie uma malha

- Liste e descreva todas as malhas
- Criação de malhas do App Mesh com tags restritas

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do App Mesh em sua conta. Essas ações podem incorrer em custos para a 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, consulteAWS Políticas gerenciadas pela AWS ou Políticas gerenciadas pela 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 de política JSON do IAM: condições no Manual 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 acionáveis para ajudar você 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 a 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 mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Usar o console do App Mesh

Para acessar o AWS App Mesh console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os recursos do App Mesh em sua AWS conta. Se você criar uma política baseada em identidade que seja mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis do IAM) com essa política. É possível anexar a política gerenciada `AWSAppMeshReadOnly` aos usuários. 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 correspondem à operação da API que você está tentando executar.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permite que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
```

```
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Crie uma malha

Este exemplo mostra como você pode criar uma política que permita que um usuário crie uma malha para uma conta, em qualquer região.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:CreateMesh",
            "Resource": "arn:aws:appmesh:*:123456789012:CreateMesh"
        }
    ]
}
```

Liste e descreva todas as malhas

Este exemplo mostra como você pode criar uma política que permita a um usuário acesso para listar ou descrever todos as malhas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "appmesh:DescribeMesh",
                "appmesh:ListMeshes"
            ],
            "Resource": "*"
        }
    ]
}
```

Criação de malhas do App Mesh com tags restritas

É possível usar tags em suas políticas do IAM para controlar quais tags podem ser transmitidas na solicitação do IAM. É possível especificar quais pares de chave e valor de tag podem ser adicionados, alterados ou removidos de um usuário do IAM ou de um perfil do IAM . Este exemplo mostra como você pode criar uma política que permita criar uma malha, mas somente se a malha for criada com uma tag chamada *teamName* e um valor de *BookSteam*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:CreateMesh",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/teamName": "booksTeam"
                }
            }
        }
    ]
}
```

É possível anexar essa política aos usuários do IAM na sua conta. Se um usuário tentar criar uma malha, a malha deverá incluir uma tag chamada `teamName` e um valor de `booksTeam`. Se a malha não incluir essa tag e esse valor, a tentativa de criar a malha falhará. Para obter mais informações,

consulte IAM JSON Policy Elements: Condition (Elementos da política JSON do IAM: Condição) no Guia do usuário do IAM.

AWS políticas gerenciadas para App Mesh

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 políticas gerenciadas pelo cliente da 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) for lançada ou novas operações de API forem disponibilizadas para serviços existentes.

Para mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

AWS política gerenciada: AWSAppMeshServiceRolePolicy

Você pode anexar `AWSAppMeshServiceRolePolicy` às entidades do IAM. Permite o acesso aos AWS serviços e recursos usados ou gerenciados por AWS App Mesh.

Para ver as permissões dessa política, consulte AWSAppMeshServiceRolePolicyna Referência de política AWS gerenciada.

Para obter informações sobre os detalhes da permissão para o `AWSAppMeshServiceRolePolicy`, consulte Permissões de funções vinculadas ao serviço para o App Mesh.

AWS política gerenciada: AWSAppMeshEnvoyAccess

Você pode anexar `AWSAppMeshEnvoyAccess` às entidades do IAM. Política do App Mesh Envoy para acessar a configuração de nó virtual.

Para ver as permissões dessa política, consulte AWSAppMeshEnvoyAccessna Referência de política AWS gerenciada.

AWS política gerenciada: AWSAppMeshFullAccess

Você pode anexar `AWSAppMeshFullAccess` às entidades do IAM. Fornece acesso total às AWS App Mesh APIs e. AWS Management Console

Para ver as permissões dessa política, consulte AWSAppMeshFullAccessna Referência de política AWS gerenciada.

AWS política gerenciada: AWSAppMeshPreviewEnvoyAccess

Você pode anexar `AWSAppMeshPreviewEnvoyAccess` às entidades do IAM. Política do App Mesh Preview Envoy para acessar a configuração de nó virtual.

Para ver as permissões dessa política, consulte AWSAppMeshPreviewEnvoyAccessna Referência de política AWS gerenciada.

AWS política gerenciada: AWSAppMeshPreviewServiceRolePolicy

Você pode anexar `AWSAppMeshPreviewServiceRolePolicy` às entidades do IAM. Permite o acesso aos AWS serviços e recursos usados ou gerenciados por AWS App Mesh.

Para ver as permissões dessa política, consulte AWSAppMeshPreviewServiceRolePolicyna Referência de política AWS gerenciada.

AWS política gerenciada: AWSAppMeshReadOnly

Você pode anexar `AWSAppMeshReadOnly` às entidades do IAM. Fornece acesso somente de leitura às AWS App Mesh APIs e. AWS Management Console

Para ver as permissões dessa política, consulte AWSAppMeshReadOnlyna Referência de política AWS gerenciada.

AWS App Mesh atualizações nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas AWS App Mesh desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações feitas nesta página, inscreva-se no feed RSS na página de histórico de documentos do AWS App Mesh .

Alteração	Descrição	Data
AWSAppMeshFullAccess— Política atualizada.	Atualizado `AWSAppMeshFullAccess` para permitir o acesso às `UntagResource` APIs `TagResource` e.	24 de abril de 2024
AWSAppMeshServiceRolePolicy, AWSServiceRoleForAppMesh— Política atualizada.	Atualizado `AWSServiceRoleForAppMesh` e `AWSAppMeshServiceRolePolicy` para permitir o acesso à AWS Cloud Map `DiscoverInstancesRevision` API.	12 de outubro de 2023

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em AWS IAM Identity Center:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do AWS IAM Identity Center .
- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criar um perfil para um provedor de identidades de terceiros (federação) no Guia do usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Usar funções vinculadas ao serviço do App Mesh

O AWS App Mesh usa funções vinculadas ao serviço do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de perfil do IAM vinculado diretamente ao App Mesh. As funções vinculadas ao serviço são predefinidas pelo App Mesh e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do App Mesh porque você não precisa adicionar as permissões necessárias manualmente. O App Mesh define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o App Mesh pode assumir suas funções. As permissões definidas incluem as políticas de confiança e de permissões, e essa política de permissões não pode ser anexada a nenhuma outra entidade do IAM.

Você pode excluir uma função vinculada ao serviço somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos do App Mesh, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas a serviços, consulte Serviços da AWS compatíveis com o IAM e procure os serviços que contenham Yes (Sim) na coluna Service-Linked Role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação da função vinculada a esse serviço.

Permissões de função vinculada ao serviço do App Mesh

O App Mesh usa a função vinculada ao serviço chamada AWSServiceRoleForAppMesh: a função permite que o App Mesh chame serviços da AWS em seu nome.

A função vinculada ao serviço AWSServiceRoleForAppMesh confia no serviço `appmesh.amazonaws.com` para assumir o perfil.

Detalhes de permissões

- `servicediscovery:DiscoverInstances`: permite que o App Mesh conclua ações em todos os recursos da AWS.
- `servicediscovery:DiscoverInstancesRevision`: permite que o App Mesh conclua ações em todos os recursos da AWS.

AWSServiceRoleForAppMesh

Esta política inclui as seguintes permissões:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CloudMapServiceDiscovery",
   "Effect": "Allow",
   "Action": [
    "servicediscovery:DiscoverInstances",
    "servicediscovery:DiscoverInstancesRevision"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ACMCertificateVerification",
   "Effect": "Allow",
   "Action": [
    "acm:DescribeCertificate"
   ],
   "Resource": "*"
  }
 ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte Permissões de função vinculada ao serviço no Guia do usuário do IAM.

Criar uma função vinculada ao serviço do App Mesh

Se você criou uma malha depois de 5 de junho de 2019 no AWS Management Console, na AWS CLI ou na API da AWS, o App Mesh criou a função vinculada ao serviço para você. Para a função vinculada ao serviço ter sido criada para você, a conta do IAM que você usou para criar a malha deve ter a política do IAM `AWSAppMeshFullAccess` anexada ou uma política anexada que contivesse a permissão `iam:CreateServiceLinkedRole`. Se excluir essa função vinculada ao serviço e precisar criá-la novamente, você poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma malha, o App Mesh cria uma função vinculada ao serviço para você novamente. Se sua conta contiver apenas malhas criadas antes de 5 de junho de 2019 e você quiser usar a função vinculada ao serviço com essas malhas, poderá criar a função usando o console do IAM.

Você pode usar o console do IAM para criar uma função vinculada ao serviço com o caso de uso do App Mesh. Na AWS CLI ou na API do AWS, crie uma função vinculada ao serviço com o nome de serviço `appmesh.amazonaws.com`. Para obter mais informações, consulte Criar uma função vinculada ao serviço no Guia do usuário do IAM. Se você excluir essa função vinculada ao serviço, será possível usar esse mesmo processo para criar a função novamente.

Editar uma função vinculada ao serviço do App Mesh

O App Mesh não permite que você edite a função vinculada ao serviço AWSServiceRoleForAppMesh. Depois que criar uma função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência a ela. No entanto, será possível editar a descrição da função usando o IAM. Para obter mais informações, consulte Editar uma função vinculada ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do App Mesh

Se você não precisar mais usar um recurso 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 os recursos de sua função vinculada ao serviço antes de excluí-la manualmente.

> **Note**
>
> Se o serviço App Mesh 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.

Para excluir os recursos do App Mesh usados pela AWSServiceRoleForAppMesh

1. Exclua todas as rotas definidas para todos os roteadores na malha.
2. Exclua todos os roteadores virtuais na malha.
3. Exclua todos os serviços virtuais na malha.
4. Exclua todos os nós virtuais na malha.
5. Exclua a malha.

Conclua as etapas anteriores para todas as malhas em sua conta.

Como excluir manualmente a função vinculada ao serviço usando o IAM

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço AWSServiceRoleForAppMesh. Para obter mais informações, consulte Excluir uma função vinculada ao serviço no Guia do usuário do IAM.

Regiões compatíveis com funções vinculadas ao serviço do App Mesh

O App Mesh oferece suporte a funções vinculadas ao serviço em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte Endpoints e cotas do App Mesh.

Autorização do Envoy Proxy

A autorização de proxy autoriza o proxy Envoy executado em uma tarefa do Amazon ECS, em um pod Kubernetes executado no Amazon EKS ou em execução em uma instância do Amazon EC2 a ler a configuração de um ou mais endpoints de malha do App Mesh Envoy Management Service. Para contas de clientes que já têm Envoys conectados ao endpoint do App Mesh antes de 26/04/2021, a autorização de proxy é necessária para nós virtuais que usam Transport Layer Security (TLS) e para gateways virtuais (com ou sem TLS). Para contas de clientes que desejam conectar os Envoys ao endpoint do App Mesh após 26/04/2021, a autorização de proxy é necessária para todos os recursos do App Mesh. É recomendável que todas as contas de clientes habilitem a autorização de proxy para todos os nós virtuais, mesmo que não usem TLS, para ter uma experiência segura e consistente usando o IAM para autorização de recursos específicos. A autorização do proxy exige que a permissão `appmesh:StreamAggregatedResources` seja especificada em uma política do IAM. A política deve ser anexada a um perfil do IAM, e esse perfil do IAM deve ser anexado ao recurso computacional no qual você host o proxy.

Criar uma política do IAM

Se quiser que todos os endpoints de malha em uma malha de serviços possam ler a configuração de todos os endpoints de malha, pule para Crie a função do IAM. Se quiser limitar os endpoints de malha a partir dos quais a configuração pode ser lida por endpoints de malha individuais, é necessário criar uma ou mais políticas do IAM. É recomendável limitar os endpoints de malha dos quais a configuração pode ser lida apenas ao proxy Envoy em execução nos recursos computacionais específicos. Crie uma política do IAM e adicione a permissão `appmesh:StreamAggregatedResources` à política. O exemplo de política a seguir permite a configuração dos nós virtuais nomeados `serviceBv1` e `serviceBv2` a serem lidos em uma malha de serviços. A configuração não pode ser lida para nenhum outro nó virtual definido na malha de

serviços. Para obter mais informações sobre como criar e editar uma política do IAM, consulte Criar política do IAM e Editar política do IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "appmesh:StreamAggregatedResources",
            "Resource": [
                "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/
serviceBv1",
                "arn:aws:appmesh:us-east-1:123456789012:mesh/app1/virtualNode/
serviceBv2"
            ]
        }
    ]
}
```

É possível criar várias políticas, sendo que cada política restringe o acesso a diferentes endpoints da malha.

Crie a função do IAM

Se quiser que todos os endpoints de malha em uma malha de serviços possam ler a configuração de todos os endpoints de malha, você só precisa criar um perfil do IAM. Se quiser limitar os endpoints de malha a partir dos quais a configuração pode ser lida por endpoints de malha individuais, é necessário criar um perfil para cada política criada na etapa anterior. Conclua as instruções para o recurso computacional no qual o proxy é executado.

- Amazon EKS: se quiser usar um único perfil, poderá usar o perfil existente que foi criado e atribuído aos nós de processamento ao criar seu cluster. Para usar vários perfis, seu cluster deve atender aos requisitos definidos em Habilitar funções do IAM para contas de serviço em seu cluster. Crie os perfis do IAM e os associe às contas de serviço do Kubernetes. Para obter mais informações, consulte Criação de um perfil do IAM e política para sua conta de serviço e Especificação de um perfil do IAM para sua conta de serviço.
- Amazon ECS: selecione serviço da AWS, selecione o Elastic Container Service e, em seguida, selecione o caso de uso da tarefa do Elastic Container Service ao criar seu perfil do IAM.

- Amazon EC2: selecione serviço da AWS, selecione EC2 e, em seguida , selecione o caso de uso do EC2 ao criar seu perfil do IAM. Isso se aplica se host o proxy diretamente em uma instância do Amazon EC2 ou no Kubernetes em execução em uma instância.

Para obter mais informações sobre como criar um perfil do IAM, consulte Criando um perfil para um serviço AWS.

Anexar política do IAM

Se quiser que todos os endpoints de malha em uma malha de serviços possam ler a configuração de todos os endpoints de malha, anexe a política do IAM `AWSAppMeshEnvoyAccess` gerenciada ao perfil do IAM que você criou na etapa anterior. Se quiser limitar os endpoints de malha a partir dos quais a configuração pode ser lida por endpoints de malha individuais, anexe cada política que você criou a cada perfil criado. Para obter mais informações sobre como anexar uma política do IAM personalizada ou gerenciada a um perfil do IAM, consulte Adicionar permissões de identidade do IAM.

Anexar um perfil do IAM

Anexe cada perfil do IAM ao recurso computacional apropriado:

- Amazon EKS: se você anexou a política do perfil associado aos seus nós de processamento, pode pular esta etapa. Se criou perfis separados, atribua cada perfil a uma conta de serviço separada do Kubernetes e atribua cada conta de serviço a uma especificação individual de implantação do pod do Kubernetes que inclua o proxy Envoy. Para obter mais informações, consulte Especificação de um perfil do IAM para sua conta de serviço no Guia do usuário do Amazon EKS e Configurar contas de serviço para pods na documentação do Kubernetes.
- Amazon ECS: anexe uma função de tarefa do Amazon ECS à definição da tarefa que inclui o proxy Envoy. A tarefa pode ser implantada com o tipo de inicialização EC2 ou Fargate. Para obter mais informações sobre como criar uma função de tarefa do Amazon ECS e anexá-la a uma tarefa, consulte Especificando um perfil do IAM para suas tarefas.
- Amazon EC2: o perfil do IAM deve ser anexado a instância do Amazon EC2 que hospeda o proxy Envoy. Para obter mais informações sobre como anexar uma função a uma instância do Amazon EC2, consulte Criei um perfil do IAM e agora quero atribuí-lo a uma instância do EC2.

Confirmar permissão

Confirme se a permissão `appmesh:StreamAggregatedResources` foi atribuída ao recurso computacional no qual você host o proxy selecionando um dos nomes do serviço de computação.

Amazon EKS

Uma política personalizada pode ser atribuída à função atribuída aos nós de processamento, aos pods individuais ou a ambos. No entanto, é recomendável atribuir à política somente em pods individuais, para que você possa restringir o acesso de pods individuais a endpoints de malha individuais. Se a política estiver anexada a função atribuída aos nós de processamento, selecione a guia Amazon EC2 e conclua as etapas encontradas lá para suas instâncias de nós de processamento. Para determinar qual perfil do IAM é atribuído a um pod do Kubernetes, conclua as etapas a seguir.

1. Veja os detalhes de uma implantação do Kubernetes que inclui o pod ao qual você deseja confirmar se uma conta de serviço do Kubernetes estiver atribuída. O comando a seguir exibe os detalhes de uma implantação chamada *my-deployment*.

   ```
   kubectl describe deployment my-deployment
   ```

   Na saída retornada, observe o valor à direita de `Service Account:`. Se uma linha que começa com `Service Account:` não existir, então uma conta de serviço personalizada do Kubernetes não está atualmente atribuída à implantação. Você precisará atribuir uma. Para obter mais informações, consulte Configurar contas de serviço para pods na documentação do Kubernetes.

2. Veja os detalhes da conta de serviço retornada na etapa anterior. O comando a seguir exibe os detalhes de uma conta de serviço chamada *my-service-account*.

   ```
   kubectl describe serviceaccount my-service-account
   ```

   Desde que a conta de serviço do Kubernetes esteja associada a uma função do AWS Identity and Access Management, uma das linhas retornadas será semelhante ao exemplo a seguir.

   ```
   Annotations:         eks.amazonaws.com/role-arn=arn:aws:iam::123456789012:role/
   my-deployment
   ```

No exemplo anterior, `my-deployment` é o nome do perfil do IAM ao qual a conta de serviço está associada. Se a saída da conta de serviço não contiver uma linha semelhante ao exemplo acima, a conta de serviço do Kubernetes não está associada a uma conta AWS Identity and Access Management e você precisa associá-la a uma. Para obter mais informações, consulte Especificar uma função do IAM para a conta de serviço.

3. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
4. No painel de navegação à esquerda, selecione Roles. Selecione o nome do perfil do IAM que você anotou em uma etapa anterior.
5. Confirme se a política personalizada que criou anteriormente ou a política `AWSAppMeshEnvoyAccess` gerenciada está listada. Se nenhuma política estiver anexada, anexe uma política do IAM ao perfil do IAM. Se quiser anexar uma política do IAM personalizada, mas não tiver uma, precisará criar uma política do IAM personalizada com as permissões necessárias. Se uma política do IAM personalizada estiver anexada, selecione a política e confirme se ela contém `"Action": "appmesh:StreamAggregatedResources"`. Caso contrário, será preciso adicionar essa permissão à sua política do IAM personalizada. Também é possível confirmar se o nome do recurso da Amazon (ARN) adequado para um endpoint de malha específico está listado. Se nenhum ARNs estiver listado, é possível editar a política para adicionar, remover ou alterar os ARNs listados. Para obter mais informações, consulte Editar políticas do IAM e Criar uma política do IAM .
6. Repita as etapas anteriores para cada pod do Kubernetes que contém o proxy Envoy.

Amazon ECS

1. No console do Amazon ECS, escolha Definições de tarefas.
2. Selecione sua tarefa do Amazon ECS.
3. Na página Nome da definição da tarefa, selecione sua definição de tarefa.
4. Na página Definição da tarefa, selecione o link do nome do perfil do IAM que está à direita da Função da tarefa. Se um perfil do IAM não estiver listado, é preciso criar um perfil do IAM e anexá-lo à sua tarefa atualizando a definição da tarefa.
5. Na página Resumo, na guia Permissões, confirme se a política personalizada criada anteriormente ou a política gerenciada `AWSAppMeshEnvoyAccess` está listada. Se nenhuma política estiver anexada, anexe uma política do IAM ao perfil do IAM. Se quiser anexar

uma política personalizada do IAM, mas não tiver uma, precisará criar a política do IAM personalizada. Se uma política do IAM personalizada estiver anexada, selecione a política e confirme se ela contém `"Action": "appmesh:StreamAggregatedResources"`. Caso contrário, será preciso adicionar essa permissão à sua política do IAM personalizada. Você também pode confirmar se o nome do recurso da Amazon (ARN) adequado para um endpoint de malha específico está listado. Se nenhum ARNs estiver listado, é possível editar a política para adicionar, remover ou alterar os ARNs listados. Para obter mais informações, consulte Editar políticas do IAM e Criar uma política do IAM .

6. Repita as etapas anteriores para cada definição de tarefa que contém o proxy Envoy.

Amazon EC2

1. No console do Amazon EC2, selecione Instâncias no painel de navegação à esquerda.
2. Selecione uma de suas instâncias que hospeda o proxy Envoy.
3. Na guia Descrição, selecione o link do nome do perfil do IAM que está à direita do perfil do IAM. Se um perfil do IAM não estiver listado, é preciso criar um perfil do IAM.
4. Na página Resumo, na guia Permissões, confirme se a política personalizada criada anteriormente ou a política gerenciada `AWSAppMeshEnvoyAccess` está listada. Se nenhuma política estiver anexada, anexe a política do IAM ao perfil do IAM. Se quiser anexar uma política personalizada do IAM, mas não tiver uma, precisará criar a política do IAM personalizada. Se uma política do IAM personalizada estiver anexada, selecione a política e confirme se ela contém `"Action": "appmesh:StreamAggregatedResources"`. Caso contrário, será preciso adicionar essa permissão à sua política do IAM personalizada. Você também pode confirmar se o nome do recurso da Amazon (ARN) adequado para um endpoint de malha específico está listado. Se nenhum ARNs estiver listado, é possível editar a política para adicionar, remover ou alterar os ARNs listados. Para obter mais informações, consulte Editar políticas do IAM e Criar uma política do IAM .
5. Repita as etapas anteriores para cada instância em que você host o proxy Envoy.

Solução de problemas AWS App Mesh de identidade e acesso

Use as seguintes informações para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao se trabalhar com o App Mesh e o IAM.

Tópicos

- Não tenho autorização para executar uma ação no App Mesh
- Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do App Mesh

Não tenho autorização para executar uma ação no App Mesh

Se isso AWS Management Console indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. Caso seu administrador seja a pessoa que forneceu suas credenciais de início de sessão.

O erro a seguir ocorre quando o usuário do `mateojackson` IAM tenta usar o console para criar um nó virtual chamado *`my-mesh my-virtual-nodena malha`*, mas não tem a `appmesh:CreateVirtualNode` permissão.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: appmesh:CreateVirtualNode on resource: arn:aws:appmesh:us-
east-1:123456789012:mesh/my-mesh/virtualNode/my-virtual-node
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir que crie um nó virtual usando a ação `appmesh:CreateVirtualNode`.

> **Note**
>
> Como um nó virtual é criado dentro de uma malha, a conta de Mateo também exige as ações `appmesh:DescribeMesh` e `appmesh:ListMeshes` para criar o nó virtual no console.

Quero permitir que pessoas fora da minha AWS conta acessem meus recursos do App Mesh

Você pode criar uma função que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode 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 Mesh oferece suporte a esses recursos, consulte Como AWS App Mesh funciona com o IAM.

- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro 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 saber a diferença entre usar perfis e políticas baseadas em recursos para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

Registrando chamadas de AWS App Mesh API usando AWS CloudTrail

AWS App Mesh é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou um AWS service (Serviço da AWS). CloudTrail captura todas as chamadas de API para o App Mesh como eventos. As chamadas capturadas incluem as chamadas do console do App Mesh e as chamadas de código para as operações da API do App Mesh. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita ao App Mesh, o endereço IP do qual a solicitação foi feita, quando foi feita e detalhes adicionais.

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 credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

CloudTrail está ativo Conta da AWS quando você cria a conta e você tem acesso automático ao histórico de CloudTrail eventos. O histórico de CloudTrail eventos fornece um registro visível, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento registrados em um. Região da AWS Para obter mais informações, consulte Trabalhando com o histórico

de CloudTrail eventos no Guia AWS CloudTrail do usuário. Não há CloudTrail cobrança pela visualização do histórico de eventos.

Para um registro contínuo dos eventos dos Conta da AWS últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do CloudTrailLake.

CloudTrail trilhas

Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Todas as trilhas criadas usando o AWS Management Console são multirregionais. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. É recomendável criar uma trilha multirregional porque você captura todas as atividades Regiões da AWS em sua conta. Se você criar uma trilha de região única, poderá visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte Criar uma trilha para a Conta da AWS e Criar uma trilha para uma organização no Guia do usuário do AWS CloudTrail .

Você pode entregar uma cópia dos seus eventos de gerenciamento em andamento para o bucket do Amazon S3 sem nenhum custo CloudTrail criando uma trilha. No entanto, existem taxas de armazenamento do Amazon S3. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços. Para receber informações sobre a definição de preço do Amazon S3, consulte Definição de preço do Amazon S3.

CloudTrail Armazenamentos de dados de eventos em Lake

CloudTrail O Lake permite que você execute consultas baseadas em SQL em seus eventos. CloudTrail O Lake converte eventos existentes no formato JSON baseado em linhas para o formato Apache ORC. O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de seletores de eventos avançados. Os seletores que você aplica a um armazenamento de dados de eventos controlam quais eventos persistem e estão disponíveis para você consultar. Para obter mais informações sobre o CloudTrail Lake, consulte Trabalhando com o AWS CloudTrail Lake no Guia AWS CloudTrail do Usuário.

CloudTrail Os armazenamentos e consultas de dados de eventos em Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a opção de preço que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Eventos de gerenciamento do App Mesh em CloudTrail

Os eventos de gerenciamento fornecem informações sobre as operações de gerenciamento que são realizadas nos recursos do seu Conta da AWS. Elas também são conhecidas como operações de plano de controle. Por padrão, CloudTrail registra eventos de gerenciamento.

AWS App Mesh registra todas as operações do plano de controle do App Mesh como eventos de gerenciamento. Para ver uma lista das operações do plano de AWS App Mesh controle nas quais o App Mesh registra CloudTrail, consulte a Referência da AWS App Mesh API.

Exemplos de eventos do App Mesh

Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a operação de API solicitada, a data e a hora da operação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, os eventos não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro que demonstra a `StreamAggregatedResources` ação.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:d060be4ac3244e05aca4e067bfe241f8",
        "arn": "arn:aws:sts::123456789012:assumed-role/Application-TaskIamRole-
C20GBLBRLBXE/d060be4ac3244e05aca4e067bfe241f8",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "invokedBy": "appmesh.amazonaws.com"
    },
    "eventTime": "2021-06-09T23:09:46Z",
    "eventSource": "appmesh.amazonaws.com",
    "eventName": "StreamAggregatedResources",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "appmesh.amazonaws.com",
    "userAgent": "appmesh.amazonaws.com",
    "eventID": "e3c6f4ce-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
```

```
    "serviceEventDetails": {
        "connectionId": "e3c6f4ce-EXAMPLE",
        "nodeArn": "arn:aws:appmesh:us-west-2:123456789012:mesh/CloudTrail-Test/
virtualNode/cloudtrail-test-vn",
        "eventStatus": "ConnectionEstablished",
        "failureReason": ""
    },
    "eventCategory": "Management"
}
```

Para obter informações sobre o conteúdo do CloudTrail registro, consulte o conteúdo do CloudTrail registro no Guia AWS CloudTrail do usuário.

Proteção de dados em AWS App Mesh

O modelo de responsabilidade AWS compartilhada modelo se aplica à proteção de dados em AWS App Mesh. 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 seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para ter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para ter mais informações sobre a proteção de dados na Europa, consulte a AWS postagem do blog Shared Responsibility Model and GDPR no AWS Blog de segurança da.

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.
- 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 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para ter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de email dos seus clientes, em marcações ou campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com o App Mesh ou outro Serviços da AWS usando o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou 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, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia de dados

Seus dados são criptografados ao usar o App Mesh.

Criptografia inativa

Por padrão, as configurações do App Mesh que você cria são criptografadas em repouso.

Criptografia em trânsito

Os endpoints do serviço App Mesh usam o protocolo HTTPS. Toda a comunicação entre o proxy Envoy e o serviço de gerenciamento do App Mesh Envoy é criptografada. Se você precisar de criptografia compatível com FIPS para a comunicação entre o proxy Envoy e o App Mesh Envoy Management Service, há uma variante FIPS da imagem do contêiner proxy Envoy que você pode usar. Para ter mais informações, consulte Imagem do Envoy.

A comunicação entre contêineres em nós virtuais não é criptografada, mas esse tráfego não sai do namespace da rede.

Validação de conformidade para AWS App Mesh

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:

- Guias de início rápido sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos AWS focados em segurança e conformidade.
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos qualificados para a HIPAA.

> **Note**
>
> Nem todos Serviços da AWS são elegíveis para a HIPAA. Para obter mais informações, consulte a Referência dos serviços qualificados pela 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 atributos 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.

Segurança da infraestrutura em AWS App Mesh

Como serviço gerenciado, AWS App Mesh é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar o App Mesh pela rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com sigilo de encaminhamento perfeito (perfect forward secrecy, ou PFS) como DHE (Ephemeral Diffie-Hellman, ou Efêmero Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman, ou Curva elíptica efêmera Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas utilizando 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.

Você pode melhorar a postura de segurança da sua VPC configurando o App Mesh para usar um endpoint da VPC de interface. Para ter mais informações, consulte Pontos de extremidade VPC da interface App Mesh ()AWS PrivateLink.

Pontos de extremidade VPC da interface App Mesh ()AWS PrivateLink

Você pode melhorar a postura de segurança da sua Amazon VPC configurando o App Mesh para usar um endpoint da VPC de interface. Os endpoints de interface são alimentados por AWS PrivateLink, uma tecnologia que permite acessar de forma privada as APIs do App Mesh usando

endereços IP privados. PrivateLinkrestringe todo o tráfego de rede entre sua Amazon VPC e App Mesh até a rede Amazon.

Você não precisa configurar PrivateLink, mas nós recomendamos. Para obter mais informações PrivateLink e fazer a interface dos VPC endpoints, consulte Acessando serviços por meio de. AWS PrivateLink

Considerações sobre endpoints VPC da interface App Mesh

Antes de configurar endpoints da VPC de interface do App Mesh, fique atento às seguintes considerações:

- Se sua Amazon VPC não tiver um gateway de internet e suas tarefas usarem o driver de `awslogs` log para enviar informações de log para CloudWatch Logs, você deverá criar uma interface VPC endpoint para Logs. CloudWatch Para obter mais informações, consulte Como usar CloudWatch registros com endpoints VPC de interface no Guia do usuário do Amazon CloudWatch Logs.
- Os VPC endpoints não oferecem suporte AWS a solicitações entre regiões. Crie o endpoint na mesma região em que você planeja emitir as chamadas de API para o App Mesh.
- Os endpoints da VPC oferecem suporte somente a DNS fornecidos pela Amazon por meio do Amazon Route 53. Se quiser usar seu próprio DNS, você pode usar o encaminhamento de DNS condicional. Para obter mais informações, consulte Conjuntos de Opções de DHCP no Manual do Usuário da Amazon VPC.
- O grupo de segurança anexado ao endpoint da VPC deve permitir conexões de entrada na porta 443 na sub-rede privada da Amazon VPC.

> **Note**
>
> O controle do acesso ao App Mesh anexando uma política de endpoint ao endpoint da VPC (por exemplo, usando o nome do serviço `com.amazonaws.`*`Region`*`.appmesh-envoy-management`) não é compatível com a conexão Envoy.

Para considerações e limitações adicionais, consulte Considerações sobre a zona de disponibilidade do endpoint de interface e Propriedades e limitações do endpoint de interface.

Crie a interface VPC endpoint para o App Mesh

Para criar o endpoint da VPC de interface para o serviço App Mesh, use o procedimento Criação de um endpoint da interface no Manual do usuário da Amazon VPC. Especifique

`com.amazonaws.Region.appmesh-envoy-management` como o nome do serviço para que seu proxy Envoy se conecte ao serviço público de gerenciamento Envoy do App Mesh e `com.amazonaws.Region.appmesh` para operações de malha.

> **Note**
>
> *Região* representa o identificador de uma AWS região compatível com o App Mesh, como `us-east-2` a região Leste dos EUA (Ohio).

Embora você possa definir um endpoint da VPC de interface para o App Mesh em qualquer região em que o App Mesh seja compatível, talvez você não consiga definir um endpoint para todas as zonas de disponibilidade em cada região. Para descobrir quais zonas de disponibilidade são compatíveis com endpoints de VPC de interface em uma região, use o describe-vpc-endpoint-services comando ou use o. AWS Management Console Por exemplo, os comandos a seguir retornam as zonas de disponibilidade nas quais você pode implantar endpoints da VPC de interface do App Mesh na região Leste dos EUA (Ohio):

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.us-east-2.appmesh-envoy-management`].AvailabilityZones[]'
```

```
aws --region us-east-2 ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.us-east-2.appmesh`].AvailabilityZones[]'
```

Resiliência no AWS App Mesh

A infraestrutura global da AWS é criada com base em regiões da AWS e zonas de disponibilidade da AWS. As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, as quais são conectadas com baixa latência, alto throughput e redes altamente redundantes. Com as zonas de disponibilidade, você pode 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.

O App Mesh executa suas instâncias do ambiente de gerenciamento entre várias Zonas de disponibilidade para garantir alta disponibilidade. O App Mesh detecta e substitui automaticamente instâncias não íntegras do ambiente de gerenciamento, além de fornecer atualizações de versão e correções automatizadas para elas.

Recuperação de desastres no AWS App Mesh

O serviço App Mesh gerencia backups dos dados do cliente. Não há nada que você precise fazer para gerenciar os backups. Os dados de backup são criptografados.

Análise de configuração e vulnerabilidade em AWS App Mesh

O App Mesh vende uma imagem do contêiner do Docker do proxy Envoy gerenciado que você implanta junto com os microsserviços. O App Mesh garante que a imagem do contêiner seja corrigida com os patches de vulnerabilidade e desempenho mais recentes. O App Mesh testa as novas versões do proxy Envoy em relação ao conjunto de atributos do App Mesh antes de disponibilizar as imagens.

É preciso atualizar seus microsserviços para usar a versão atualizada da imagem do contêiner. A seguir a versão mais recente da imagem.

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.27.3.0-prod
```

Solução de problemas do App Mesh

Este capítulo discute práticas recomendadas para a solução de problemas e de problemas comuns que podem ser encontrados durante o uso do App Mesh. Selecione uma das áreas a seguir para analisar as melhores práticas e os problemas comuns dessa área.

Tópicos

Solução de problemas e práticas recomendadas

Recomendamos seguir as práticas recomendadas deste tópico para solucionar problemas ao usar o App Mesh.

Ativar a interface de administração do proxy Envoy

O proxy Envoy vem com uma interface de administração que pode ser usada para descobrir configurações e estatísticas e realizar outras funções administrativas, como drenagem da conexão. Para obter mais informações, consulte Interface de administração, na documentação do Envoy.

Se você usar o Imagem do Envoy gerenciado, o endpoint de administração será habilitado por padrão na porta 9901. Os exemplos fornecidos em Solução de problemas de configuração do App Mesh exibem o exemplo de URL do endpoint de administração como `http://my-app.default.svc.cluster.local:9901/`.

> **Note**
>
> O endpoint da administração nunca deve ser exposto à internet pública. Além disso, recomendamos monitorar os registros do endpoint de administração, que são

definidos pela variável de ambiente `ENVOY_ADMIN_ACCESS_LOG_FILE` para `/tmp/envoy_admin_access.log`, por padrão.

Habilite a integração do Envoy DogStats D para descarga métrica

O proxy Envoy pode ser configurado para descarregar estatísticas para o tráfego das camadas OSI 4 e 7 e para a integridade do processo interno. Embora este tópico mostre como usar essas estatísticas sem transferir as métricas para coletores como métricas CloudWatch e Prometheus., ter essas estatísticas em um local centralizado para todos os seus aplicativos pode ajudá-lo a diagnosticar problemas e confirmar o comportamento mais rapidamente. Para obter mais informações, consulte Usando o Amazon CloudWatch Metrics e a documentação do Prometheus.

Você pode configurar métricas DogStats D definindo os parâmetros definidos emDogStatsVariáveis D. Para obter mais informações sobre DogStats D, consulte a documentação de DogStatsD. Você pode encontrar uma demonstração da transferência de métricas para AWS CloudWatch métricas no tutorial básico do App Mesh com o Amazon ECS. GitHub

Habilitar logs de acesso

Recomendamos ativar os registros de acesso em seu Nós virtuais e Gateways virtuais para descobrir detalhes sobre o tráfego que transita entre suas aplicações. Para obter mais informações, consulte Logs de acesso na documentação do Envoy. Os registros fornecem informações detalhadas sobre o comportamento do tráfego nas camadas OSI 4 e 7. Ao usar o formato padrão do Envoy, você pode analisar os registros de acesso com CloudWatch o Logs Insights usando a seguinte declaração de análise.

```
parse @message "[*] \"* * *\" * * * * * * * * * * *" as StartTime, Method, Path,
 Protocol, ResponseCode, ResponseFlags, BytesReceived, BytesSent, DurationMillis,
 UpstreamServiceTimeMillis, ForwardedFor, UserAgent, RequestId, Authority, UpstreamHost
```

Ative o registro de depuração do Envoy em ambientes de pré-produção

Recomendamos definir o nível de log do proxy Envoy como `debug` em um ambiente de pré-produção. Os logs de depuração podem ajudar a identificar problemas antes de promover a definição associada ao App Mesh para seu ambiente de produção.

Se estiver usando a imagem do Envoy, poderá definir o nível do log para `debug` por meio da variável de ambiente `ENVOY_LOG_LEVEL`.

> **Note**
>
> Não é recomendado usar esse nível `debug` em ambientes de produção. Definir o nível para `debug` aumentar o registro e pode afetar o desempenho e o custo geral dos registros transferidos para soluções como CloudWatch o Logs.

Ao usar o formato padrão do Envoy, você pode analisar os registros do processo com CloudWatch o Logs Insights usando a seguinte declaração de análise:

```
parse @message "[*][*][*][*] [*] *" as Time, Thread, Level, Name, Source, Message
```

Monitore a conectividade do Envoy Proxy com o ambiente de gerenciamento App Mesh

É recomendável monitorar as métricas do Envoy `control_plane.connected_state` para garantir que o proxy do Envoy se comunique com o ambiente de gerenciamento do App Mesh para buscar os recursos de configuração dinâmica. Para obter mais informações, consulte Servidor de gerenciamento.

Solução de problemas de configuração do App Mesh

Este tópico detalha problemas comuns que você pode enfrentar com a configuração do App Mesh.

Não é possível extrair a imagem do contêiner Envoy

Sintomas

A seguinte mensagem de erro é recebida em uma tarefa do Amazon ECS. O *ID da conta* do Amazon ECR e a *região* na mensagem a seguir podem ser diferentes, dependendo do repositório do Amazon ECR do qual extraiu a imagem do contêiner.

```
CannotPullContainerError: Error response from daemon: pull access denied
 for 840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy, repository does
 not exist or may require 'docker login'
```

Resolução

Esse erro indica que a função de execução de tarefas que está sendo usada não tem permissão para se comunicar com o Amazon ECR e não pode extrair a imagem do contêiner Envoy do repositório. A função de execução de tarefas atribuída à sua tarefa do Amazon ECS precisa de uma política do IAM com as seguintes declarações:

```
{
  "Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
  ],
  "Resource": "arn:aws:ecr:us-west-2:111122223333:repository/aws-appmesh-envoy",
  "Effect": "Allow"
},
{
  "Action": "ecr:GetAuthorizationToken",
  "Resource": "*",
  "Effect": "Allow"
}
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível conectar-se ao serviço de gerenciamento do App Mesh Envoy

Sintomas

Seu proxy Envoy não consegue se conectar ao serviço de gerenciamento do App Mesh Envoy. Você está vendo:

- Erros de conexão recusada
- Tempos limite de conexão
- Erros ao resolver o endpoint do serviço de gerenciamento do App Mesh Envoy
- Erros de gRPC

Resolução

Certifique-se de que seu proxy Envoy tenha acesso à Internet ou a um endpoint da VPC privado e que seus grupos de segurança permitam tráfego de saída na porta 443. Os endpoint públicos do

serviço de gerenciamento do App Mesh Envoy seguem o formato de nome de domínio totalmente qualificado (FQDN).

```
# App Mesh Production Endpoint
appmesh-envoy-management.Region-code.amazonaws.com

# App Mesh Preview Endpoint
appmesh-preview-envoy-management.Region-code.amazonaws.com
```

É possível depurar sua conexão com o EMS usando o comando abaixo. Uma solicitação gRPC válida é enviada, porém vazia, para o Envoy Management Service.

```
curl -v -k -H 'Content-Type: application/grpc' -X POST https://
appmesh-envoy-management.Region-code.amazonaws.com:443/
envoy.service.discovery.v3.AggregatedDiscoveryService/StreamAggregatedResources
```

Se receber essas mensagens de volta, sua conexão com o Envoy Management Service está funcional. Para depurar erros relacionados ao gRPC, consulte os erros no Envoy desconectado do serviço de gerenciamento do App Mesh Envoy com texto de erro.

```
grpc-status: 16
grpc-message: Missing Authentication Token
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Envoy desconectado do serviço de gerenciamento do App Mesh Envoy com texto de erro

Sintomas

Seu proxy Envoy não consegue se conectar ao serviço de gerenciamento do App Mesh Envoy e receber a configuração. Seus logs de proxy do Envoy contêm uma entrada de log semelhante à seguinte.

```
gRPC config stream closed: gRPC status code, message
```

Resolução

Na maioria dos casos, a parte da mensagem do log deve indicar o problema. A tabela a seguir lista os códigos de status do gRPC mais comuns que podem ser vistos, suas causas e suas resoluções.

Código de status do gRPC	Causa	Resolução
`0`	Desconexão tranquila do serviço de gerenciamento Envoy.	Não há problema. O App Mesh ocasionalmente desconecta os proxies do Envoy com esse código de status. O Envoy se reconectará e continuará recebendo atualizações.
3	O endpoint de malha (nó virtual ou gateway virtual), ou um de seus recursos associados, não pôde ser encontrado.	Verifique novamente a configuração do Envoy para garantir que ela tenha o nome apropriado do recurso App Mesh que representa. Se seu recurso do App Mesh estiver integrado a outros recursos da AWS, como namespaces do AWS Cloud Map ou certificados do ACM, certifique-se de que esses recursos existam.
7	O proxy Envoy não está autorizado a realizar uma ação, como conectar-se ao serviço de gerenciamento do Envoy ou recuperar recursos associados.	Certifique-se de criar uma política do IAM que tenha as declarações de política apropriadas para o App Mesh e outros serviços e anexe essa política ao usuário do usuário do IAM ou perfil do IAM que seu proxy Envoy está usando para se conectar ao serviço de gerenciamento do Envoy.

Código de status do gRPC	Causa	Resolução
8	O número de proxies do Envoy para um determinado recurso do App Mesh excede a cota de serviço no nível da conta.	Para obter mais informaçõ es sobre as cotas padrão da conta e como solicitar um aumento de cota, consulte Service Quotas do App Mesh.
16	O proxy Envoy não tem credenciais de autenticação válidas para a AWS.	Certifique-se de que o Envoy tenha as credenciais apropriad as para se conectar aos serviços da AWS por meio de um usuário do usuário do IAM ou perfil do IAM. Um problema conhecido, #24136, no Envoy for version `v1.24` e anterior não consegue obter as credenciais se o processo Envoy usar mais de descritor es de arquivo `1024`. Isso acontece quando o Envoy está atendendo a um alto volume de tráfego. É possível confirmar esse problema verificando os logs do Envoy no nível de depuração para ver o texto "`A libcurl function was given a bad argument`". Para mitigar esse problema, atualize para a versão `v1.25.1.0-prod` do Envoy ou posterior.

Você pode observar os códigos de status e as mensagens do seu proxy Envoy com o Amazon CloudWatch Insights usando a seguinte consulta:

```
filter @message like /gRPC config stream closed/
| parse @message "gRPC config stream closed: *, *" as StatusCode, Message
```

Se a mensagem de erro fornecida não foi útil ou seu problema ainda não foi resolvido, considere abrir um GitHub problema.

Falha na verificação de integridade do contêiner Envoy, na sonda de prontidão ou na sonda de vitalidade

Sintomas

Seu proxy Envoy está falhando nas verificações de integridade em uma tarefa do Amazon ECS, uma instância do Amazon EC2 ou um pod do Kubernetes. Por exemplo, você consulta a interface de administração do Envoy com o comando a seguir e recebe um status diferente de `LIVE`.

```
curl -s http://my-app.default.svc.cluster.local:9901/server_info | jq '.state'
```

Resolução

Veja a seguir uma lista de etapas de remediação, dependendo do status retornado pelo proxy Envoy.

- O `PRE_INITIALIZING` ou `INITIALIZING`: o proxy Envoy ainda não recebeu a configuração ou não pode se conectar e recuperar a configuração do serviço de gerenciamento App Mesh Envoy. O Envoy pode estar recebendo um erro do serviço de gerenciamento do Envoy ao tentar se conectar. Para obter mais informações, consulte erros no Envoy desconectado do serviço de gerenciamento do App Mesh Envoy com texto de erro.
- `DRAINING`: o proxy Envoy começou a drenar conexões em resposta a uma solicitação `/healthcheck/fail` ou `/drain_listeners` na interface de administração do Envoy. Não recomendamos invocar esses caminhos na interface de administração, a menos que você esteja prestes a encerrar sua tarefa do Amazon ECS, instância do Amazon EC2 ou pod do Kubernetes.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

A verificação de integridade do balanceador de carga até o endpoint da malha está falhando

Sintomas

O endpoint de malha é considerado íntegro pela verificação de integridade ou pela sonda de prontidão do contêiner, mas a verificação de integridade do balanceador de carga para o endpoint de malha está falhando.

Resolução

Para resolver esse problema, conclua as tarefas a seguir.

- Certifique-se de que o grupo de segurança associado ao seu endpoint de malha aceite tráfego de entrada na porta que você configurou para sua verificação de integridade.
- Certifique-se de que a verificação de integridade seja bem-sucedida de forma consistente quando solicitada manualmente, por exemplo, de um bastion host dentro da sua VPC.
- Se estiver configurando uma verificação de integridade para um nó virtual, recomendamos implementar um endpoint de verificação de integridade em sua aplicação, por exemplo, /ping para HTTP. Isso garante que tanto o proxy Envoy quanto sua aplicação sejam roteáveis a partir do balanceador de carga.
- É possível usar qualquer tipo de balanceador de carga elástico para o nó virtual, dependendo dos recursos necessários. Para obter mais informações, consulte Recursos do Elastic Load Balancing.
- Se estiver configurando uma verificação de integridade para um gateway virtual, recomendamos usar um Network Load Balancer com uma verificação de integridade TCP ou TLS na porta do receptor do gateway virtual. Isso garante que o receptor do gateway virtual esteja inicializado e pronto para aceitar conexões.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

O gateway virtual não aceita tráfego nas portas 1024 ou inferiores

Sintomas

Seu gateway virtual não aceita tráfego na porta 1024 ou inferior, mas aceita tráfego em um número de porta maior que 1024. Por exemplo, você consulta as estatísticas do Envoy com o comando a seguir e recebe um valor diferente de zero.

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "update_rejected"
```

Talvez veja um texto semelhante ao texto a seguir em seus logs descrevendo uma falha na vinculação a uma porta privilegiada:

```
gRPC config for type.googleapis.com/envoy.api.v2.Listener rejected: Error adding/
updating listener(s) lds_ingress_0.0.0.0_port_<port num>: cannot bind '0.0.0.0:<port
 num>': Permission denied
```

Resolução

Para resolver o problema, o usuário especificado para o gateway precisa ter capacidade linux `CAP_NET_BIND_SERVICE`. Para obter mais informações, consulte Capacidades no Manual do Programador Linux, Parâmetros do Linux nos parâmetros de definição de tarefas do ECS e Definir capacidades para um contêiner na documentação do Kubernetes.

> **Important**
>
> O Fargate deve usar um valor de porta maior que 1024.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Solução de problemas de conectividade do App Mesh

Este tópico detalha problemas comuns que você pode enfrentar com a conectividade do App Mesh.

Não é possível resolver o nome DNS para um serviço virtual

Sintomas

A aplicação não consegue resolver o nome DNS de um serviço virtual ao qual está tentando se conectar.

Resolução

Esse é um problema conhecido. Para obter mais informações, consulte o problema Nome VirtualServices por qualquer nome de host ou FQDN GitHub . Os serviços virtuais no App Mesh podem ter qualquer nome. Desde que haja um registro DNS `A` para o nome do serviço virtual e a aplicação possa resolver o nome do serviço virtual, a solicitação será intermediada por proxy pelo Envoy e roteada para seu destino apropriado. Para resolver o problema, adicione um registro DNS `A` a qualquer endereço IP não loopback, como `10.10.10.10`, para o nome do serviço virtual. O registro DNS `A` pode ser adicionado nas seguintes condições:

- No Amazon Route 53, se o nome for sufixado pelo nome da sua zona hospedada privada
- Dentro do arquivo `/etc/hosts` do contêiner da aplicação
- Em um servidor DNS de terceiros que você gerencia

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível conectar-se a um back-end de serviço virtual

Sintomas

Sua aplicação não consegue estabelecer uma conexão com um serviço virtual definido como back-end em seu nó virtual. Ao tentar estabelecer uma conexão, a conexão pode falhar completamente ou a solicitação, do ponto de vista da aplicação, pode falhar com um código de resposta `HTTP 503`.

Resolução

Se a aplicação não conseguir se conectar (nenhum código de resposta `HTTP 503` retornado), faça o seguinte:

- Certifique-se de que seu ambiente computacional tenha sido configurado para funcionar com o App Mesh.
  - Para o Amazon ECS, certifique-se de que a configuração de proxy apropriada esteja habilitada. Para ver um end-to-end passo a passo, consulte Getting Started with App Mesh e Amazon ECS.
  - Para o Kubernetes, incluindo o Amazon EKS, certifique-se de ter o mais recente controlador do App Mesh instalado via Helm. Para obter mais informações, consulte Controlador do App Mesh no Helm Hub ou Tutorial: Configurar a integração do App Mesh com o Kubernetes.
  - Para o Amazon EC2, certifique-se de configurar sua instância do Amazon EC2 para fazer proxy de tráfego do App Mesh. Para obter mais informações, consulte Atualizar serviço.
- Certifique-se de que o contêiner Envoy que está sendo executado em seu serviço de computação tenha se conectado com êxito ao serviço de gerenciamento do App Mesh Envoy. É possível confirmar isso verificando as estatísticas do Envoy para o campo `control_plane.connected_state`. Para obter mais informações sobre o `control_plane.connected_state`, consulte Monitorar a conectividade do proxy Envoy em nossas melhores práticas de solução de problemas.

Se o Envoy conseguiu estabelecer a conexão inicialmente, mas depois foi desconectado e nunca reconectado, consulte Envoy desconectado do serviço de gerenciamento do App Mesh Envoy com texto de erro para solução de problemas sobre motivo pelo qual ele foi desconectado.

Se a aplicação se conectar, mas a solicitação falhar com um código de resposta `HTTP 503`, tente o seguinte:

- Certifique-se de que o serviço virtual ao qual você está se conectando exista na malha.
- Certifique-se de que o serviço virtual tenha um provedor (um roteador virtual ou um nó virtual).
- Ao usar o Envoy como um proxy HTTP, se estiver vendo tráfego de saída chegando em `cluster.cds_egress_*_mesh-allow-all` ao invés do destino correto dos dados estatísticos do Envoy, é muito provável que o Envoy não esteja roteando corretamente as solicitações `filter_chains`. Isso pode ser resultado do uso de um nome de serviço virtual não qualificado. Recomendamos usar o nome de descoberta de serviços real como nome do serviço virtual, pois o proxy Envoy se comunica com outros serviços virtuais por meio de seus nomes.

  Para obter mais informações, consulte serviços virtuais.
- Inspecione os logs do proxy do Envoy em busca de qualquer uma das seguintes mensagens de erro:
  - `No healthy upstream`: o nó virtual para o qual o proxy Envoy está tentando rotear não tem nenhum endpoint resolvido ou não tem nenhum endpoint íntegro. Certifique-se de que o nó virtual de destino tenha as configurações corretas de descoberta de serviços e verificação de integridade.

    Se as solicitações para o serviço falharem durante uma implantação ou escalonamento do serviço virtual de back-end, siga as orientações em Algumas solicitações falham com o código de status HTTP 503 quando um serviço virtual tem um provedor de nó virtual.
  - `No cluster match for URL`: isso provavelmente é causado quando uma solicitação é enviada para um serviço virtual que não corresponde aos critérios definidos por nenhuma das rotas definidas em um provedor de roteador virtual. Certifique-se de que as solicitações doa aplicação sejam enviadas para uma rota compatível, garantindo que o caminho e os cabeçalhos da solicitação HTTP estejam corretos.
  - `No matching filter chain found`: isso provavelmente é causado quando uma solicitação é enviada para um serviço virtual em uma porta inválida. Verifique se as solicitações da aplicação estão usando a mesma porta especificada no roteador virtual.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível conectar-se a um serviço externo

Sintomas

Sua aplicação não consegue se conectar a um serviço fora da malha, como o `amazon.com`.

Resolução

Por padrão, o App Mesh não permite tráfego de saída de aplicações dentro da malha para nenhum destino fora da malha. Para habilitar a comunicação com um serviço externo, há duas opções:

- Definir o filtro de saída no recurso de malha como `ALLOW_ALL`. Essa configuração permitirá que qualquer aplicação dentro da malha se comunique com qualquer endereço IP de destino dentro ou fora da malha.
- Modele o serviço externo na malha usando um serviço virtual, roteador virtual, rota e nó virtual. Por exemplo, para modelar o serviço externo do `example.com`, você pode criar um serviço virtual chamado `example.com` com um roteador virtual e uma rota que envia todo o tráfego para um nó virtual com um nome de host de descoberta de serviços DNS de `example.com`.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível conectar-se a um servidor MySQL ou SMTP

Sintomas

Ao permitir tráfego de saída para todos os destinos (Mesh `EgressFilter type=ALLOW_ALL`), como um servidor SMTP ou um banco de dados MySQL usando uma definição de nó virtual, a conexão do seu aplicativo falha. Como exemplo, a seguir está uma mensagem de erro da tentativa de conexão com um servidor MySQL.

```
ERROR 2013 (HY000): Lost connection to MySQL server at 'reading initial communication
 packet', system error: 0
```

Resolução

Esse é um problema conhecido que é resolvido com o uso da imagem do App Mesh versão 1.15.0 ou posterior. Para obter mais informações, consulte o problema Não é possível se conectar ao MySQL com o App Mesh GitHub . Este erro ocorre porque o receptor de saída no Envoy configurado pelo App Mesh adiciona o filtro de receptor Inspetor de TLS do Envoy. Para obter mais informações, consulte Inspector de TLS na documentação do Envoy. Esse filtro avalia se uma conexão está ou não usando TLS inspecionando o primeiro pacote enviado pelo cliente. Com o MySQL e o SMTP, no entanto, o servidor envia o primeiro pacote após a conexão. Para obter mais informações sobre o MySQL, consulte Initial Handshake na documentação do MySQL. Como o servidor envia o primeiro pacote, a inspeção no filtro falha.

Para contornar esse problema, dependendo da sua versão do Envoy:

- Se a versão do App Mesh image Envoy for 1.15.0 ou posterior, não modele serviços externos como MySQL, SMTP, MSSQL, etc. como back-end para o nó virtual da sua aplicação.
- Se a versão do Envoy na imagem do App Mesh for anterior à 1.15.0, adicione a porta `3306` à lista de valores para o `APPMESH_EGRESS_IGNORED_PORTS` em seus serviços para MySQL e como a porta que você está usando para STMP.

> **Important**
>
> Embora as portas SMTP padrão sejam `25`, `587` e `465`, você só deve adicionar a porta que está usando ao `APPMESH_EGRESS_IGNORED_PORTS` e não todas as três.

Para obter mais informações, consulte Serviços de atualização para Kubernetes, Serviços de atualização para Amazon ECS ou Serviços de atualização para Amazon EC2.

Se o problema ainda não foi resolvido, você pode nos fornecer detalhes sobre o que está enfrentando ao usar o GitHub problema existente ou entrar em contato com o AWSSupport.

Não é possível se conectar a um serviço modelado como um nó virtual TCP ou roteador virtual no App Mesh

Sintomas

Seu aplicativo não consegue se conectar a um back-end que usa a configuração do protocolo TCP na definição do App Mesh PortMapping.

Resolução

Esse é um problema conhecido. Para obter mais informações, consulte Roteamento para vários destinos TCP na mesma porta ativada. GitHub Atualmente, o App Mesh não permite que vários destinos de back-end modelados como TCP compartilhem a mesma porta devido a restrições nas informações fornecidas ao proxy Envoy na camada 4 da OSI. Para garantir que o tráfego TCP possa ser roteado adequadamente para todos os destinos de back-end, faça o seguinte:

- Certifique-se de que todos os destinos estejam usando uma porta exclusiva. Se estiver usando um provedor de roteador virtual para o serviço virtual de back-end, poderá alterar a porta do roteador virtual sem alterar a porta nos nós virtuais para os quais ele é roteado. Isso permite que as aplicações abram conexões na porta do roteador virtual enquanto o proxy Envoy continua usando a porta definida no nó virtual.
- Se o destino modelado como TCP for um servidor MySQL, ou qualquer outro protocolo baseado em TCP no qual o servidor envia os primeiros pacotes após a conexão, consulte Não é possível conectar-se a um servidor MySQL ou SMTP.

Se o problema ainda não foi resolvido, você pode nos fornecer detalhes sobre o que está enfrentando ao usar o GitHub problema existente ou entrar em contato com o AWSSupport.

A conectividade é bem-sucedida em um serviço não listado como back-end de serviço virtual para um nó virtual

Sintomas

Sua aplicação é capaz de se conectar e enviar tráfego para um destino que não está especificado como back-end de serviço virtual em seu nó virtual.

Resolução

Se as solicitações forem bem-sucedidas em um destino que não tenha sido modelado nas APIs do App Mesh, a causa mais provável é que o tipo de filtro de saída da malha tenha sido definido como `ALLOW_ALL`. Quando o filtro de saída estiver definido como `ALLOW_ALL`, uma solicitação de saída da sua aplicação que não corresponda a um destino modelado (back-end) será enviada para o endereço IP de destino definido pela aplicação.

Se quiser proibir o tráfego para destinos não modelados na malha, considere definir o valor do filtro de saída como `DROP_ALL`.

> **Note**
>
> A configuração do valor do filtro de saída da malha afeta todos os nós virtuais dentro da malha.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Algumas solicitações falham com o código de status HTTP **503** quando um serviço virtual tem um provedor de nó virtual

Sintomas

Uma parte das solicitações da sua aplicação falha em um back-end de serviço virtual que está usando um provedor de nó virtual em vez de um provedor de roteador virtual. Ao usar um provedor de roteador virtual para o serviço virtual, as solicitações não falham.

Resolução

Esse é um problema conhecido. Para obter mais informações, consulte a política de repetição do provedor de nó virtual para um serviço virtual em GitHub. Ao usar um nó virtual como provedor de um serviço virtual, não é possível especificar a política padrão de tentativas que deseja que os clientes do seu serviço virtual usem. Em comparação, os provedores de roteadores virtuais permitem que políticas de tentativas sejam especificadas porque elas são uma propriedade dos recursos da rota subordinada..

Para reduzir as falhas de solicitação aos provedores de nós virtuais, use um provedor de roteador virtual e, em vez disso, especifique uma política de tentativas em suas rotas. Para outras formas de reduzir as falhas de solicitação em seus aplicativos, consulte Práticas recomendadas do App Mesh.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível conectar-se a um sistema de arquivos do Amazon EFS

Sintomas

Ao configurar uma tarefa do Amazon ECS com um sistema de arquivos do Amazon EFS como volume, a tarefa falha ao iniciar com o seguinte erro.

```
ResourceInitializationError: failed to invoke EFS utils commands to set up EFS volumes:
 stderr: mount.nfs4: Connection refused : unsuccessful EFS utils command execution;
 code: 32
```

Resolução

Esse é um problema conhecido. Esse erro ocorre porque a conexão do NFS com o Amazon EFS ocorre antes que qualquer contêiner em sua tarefa tenha sido iniciado. Esse tráfego é roteado pela configuração do proxy para o Envoy, que não estará em execução neste momento. Devido à ordem de startup, o cliente NFS falha ao se conectar ao sistema de arquivos do Amazon EFS e a tarefa falha ao ser iniciada. Para resolver o problema, adicione a porta `2049` à lista de valores para a configuração `EgressIgnoredPorts` na configuração de proxy da sua definição de tarefa do Amazon ECS. Para obter mais informações, consulte Configuração de proxy.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

A conectividade é bem-sucedida no serviço, mas a solicitação recebida não aparece nos registros de acesso, rastreamentos ou métricas do Envoy

Sintomas

Mesmo que a aplicação possa se conectar e enviar solicitações para outra aplicação, você não pode ver as solicitações recebidas nos logs de acesso ou nas informações de rastreamento do proxy Envoy.

Resolução

Esse é um problema conhecido. Para obter mais informações, consulte configuração das regras do iptables em problema no GitHub. O proxy Envoy só intercepta apenas o tráfego de entrada para a porta na qual o nó virtual correspondente está escutando. As solicitações para qualquer outra porta irão ignorar o proxy Envoy e chegar diretamente ao serviço por trás dele. Para permitir que o proxy Envoy intercepte o tráfego de entrada do seu serviço, é preciso configurar seu nó virtual e o serviço para escutar na mesma porta.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Definir as variáveis de ambiente **HTTP_PROXY/HTTPS_PROXY** no nível do contêiner não funciona conforme o esperado.

Sintomas

Quando HTTP_PROXY/HTTPS_PROXY é definido como uma variável de ambiente no:

- No contêiner da aplicação na definição da tarefa com o App Mesh ativado, as solicitações enviadas para o namespace dos serviços do App Mesh receberão respostas de erro `HTTP 500` do sidecar do Envoy.
- O contêiner Envoy na definição da tarefa com o App Mesh ativado, as solicitações provenientes do sidecar do Envoy não passarão pelo servidor proxy `HTTP/HTTPS` e a variável de ambiente não funcionará.

Resolução

Para o contêiner da aplicação:

O App Mesh funciona fazendo com que o tráfego em sua tarefa passe pelo proxy Envoy. A configuração `HTTP_PROXY/HTTPS_PROXY` substitui esse comportamento configurando o tráfego do contêiner para passar por um proxy externo diferente. O tráfego ainda será interceptado pelo Envoy, mas ele não oferece suporte ao proxy do tráfego de malha usando um proxy externo.

Se quiser fazer o proxy de todo o tráfego que não seja de malha, defina o `NO_PROXY` para incluir o CIDR/namespace, o host local e os endpoints da credencial da malha, como no exemplo a seguir.

```
NO_PROXY=localhost,127.0.0.1,169.254.169.254,169.254.170.2,10.0.0.0/16
```

Para o contêiner Envoy:

O Envoy não oferece suporte a um proxy genérico. Não recomendamos definir essas variáveis.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Tempos limite de solicitação upstream mesmo depois de definir o tempo limite para rotas.

Sintomas

Tempo limite definido para:

- As rotas, mas você ainda está recebendo um erro de tempo limite da solicitação upstream.
- O receptor do nó virtual, o tempo limite e o tempo limite de novas tentativas das rotas, mas ainda assim está recebendo um erro de tempo limite de solicitação upstream.

Resolução

Para que as solicitações de alta latência superiores a 15 segundos sejam concluídas com êxito, é preciso especificar um tempo limite no nível da rota e do receptor do nó virtual.

Se você especificar um tempo limite de rota maior que o padrão de 15 segundos, certifique-se de que o tempo limite também seja especificado para o receptor de todos os nós virtuais participantes. No entanto, se você diminuir o tempo limite para um valor menor que o padrão, é opcional atualizar os tempos limite nos nós virtuais. Para obter mais informações sobre as opções ao configurar nós e rotas virtuais, consulte nós virtuais e rotas virtuais.

Se especificou uma política de tentativa, a duração especificada para o tempo limite da solicitação deve sempre ser maior ou igual ao tempo limite de tentativa multiplicado pelo máximo de tentativas que você definiu na política de tentativas. Isso permite que a solicitação com todas as novas tentativas seja concluída com êxito. Para obter mais informações, consulte rotas.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

O Envoy responde com uma solicitação HTTP inválida.

Sintomas

O Envoy responde com solicitação HTTP 400 inválida para todas as solicitações enviadas pelo Network Load Balancer (NLB). Ao verificamos os logs do Envoy, vemos:

- Logs de depuração:

```
dispatch error: http/1.1 protocol error: HPE_INVALID_METHOD
```

- Logs de acesso:

```
"- - HTTP/1.1" 400 DPE 0 11 0 - "-" "-" "-" "-" "-"
```

Resolução

A resolução é desativar o protocolo proxy versão 2 (PPv2) em atributos do grupo de destino do seu NLB.

Atualmente, o PPv2 não é compatível com gateway virtual e pelo nó virtual Envoy que são executados usando o ambiente de gerenciamento do App Mesh. Se você implantar o NLB usando o controlador de balanceador de carga AWS no Kubernetes, desative o PPv2 definindo o seguinte atributo como `false`:

```
service.beta.kubernetes.io/aws-load-balancer-target-group-attributes:
 proxy_protocol_v2.enabled
```

Consulte Anotações do controlador balanceador de carga da AWS para obter mais detalhes sobre os atributos de recursos do NLB.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não foi possível configurar o tempo limite corretamente.

Sintomas

Sua solicitação atinge o tempo limite em 15 segundos, mesmo depois de configurar o tempo limite no receptor do nó virtual e o tempo limite na rota em direção ao back-end do nó virtual.

Resolução

Certifique-se de que o serviço virtual correto esteja incluído na lista de back-end.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Solução de problemas de escalabilidade do App Mesh

Este tópico detalha problemas comuns que você pode enfrentar com o escalonamento do App Mesh.

A conectividade falha e as verificações de integridade do contêiner falham ao escalar além de 50 réplicas para um nó/gateway virtual

Sintomas

Ao escalonar o número de réplicas, como tarefas do Amazon ECS, pods do Kubernetes ou instâncias do Amazon EC2, para um nó virtual/gateway virtual além de 50, as verificações de integridade do contêiner Envoy para Envoys novos e atualmente em execução começam a falhar. As aplicações downstream que enviam tráfego para o nó/gateway virtual começam a ver falhas de solicitação com o código de status HTTP `503`.

Resolução

A cota padrão do App Mesh para o número de Envoys por nó /gateway virtual é 50. Quando o número de Envoys em execução excede essa cota, os Envoys novos e atualmente em execução não conseguem se conectar ao serviço de gerenciamento Envoy do App Mesh com o código de status gRPC 8 (`RESOURCE_EXHAUSTED`). Essa cota pode ser aumentada. Para ter mais informações, consulte Service Quotas do App Mesh.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

As solicitações falham com o **`503`** quando um back-end de serviço virtual se expande horizontalmente para fora ou para dentro

Sintomas

Quando um serviço virtual de back-end é dimensionado horizontalmente para fora ou para dentro, as solicitações de aplicações downstream falham com um código de status `HTTP 503`.

Resolução

O App Mesh recomenda várias abordagens para mitigar os casos de falha ao mesmo tempo em que escalam as aplicações horizontalmente. Para obter informações detalhadas sobre como evitar essas falhas, consulte Práticas recomendadas do App Mesh.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

O contêiner Envoy trava com segfault sob carga aumentada

Sintomas

Sob uma alta carga de tráfego, o proxy Envoy falha devido a uma falha de segmentação (código `139` de saída do Linux). Os logs do processo do Envoy contêm uma declaração como a seguinte.

```
Caught Segmentation fault, suspect faulting address 0x0"
```

Resolução

O proxy Envoy provavelmente violou o nofile ulimit padrão do sistema operacional, o limite do número de arquivos que um processo pode ter abertos por vez. Essa violação se deve ao tráfego que causa mais conexões, que consomem soquetes adicionais do sistema operacional. Para resolver esse problema, aumente o valor de ulimit nofile no sistema operacional do host. Se estiver usando o Amazon ECS, esse limite pode ser alterado por meio das configurações de limite máximo nas configurações do configuração de limites de recursos da definição da tarefa.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

O aumento nos recursos padrão não se reflete nos limites de serviço

Sintomas

Depois de aumentar o limite padrão dos recursos do App Mesh, o novo valor não é refletido ao analisar seus limites de serviço.

Resolução

Embora os novos limites não estejam sendo exibidos atualmente, os clientes ainda podem utilizá-los..

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

A aplicação falha devido a um grande número de chamadas de verificação de integridade.

Sintomas

Depois de ativar a verificação de integridade ativa para um nó virtual, há um aumento no número de chamadas de verificação de integridade. A aplicação trava devido ao grande aumento do volume de chamadas de verificação de integridade feitas à aplicação.

Resolução

Quando a verificação ativa de integridade está ativada, cada endpoint Envoy do downstream (cliente) envia solicitações de integridade para cada endpoint do cluster upstream (servidor) para tomar decisões de roteamento. Como resultado, o número total de solicitações de verificação de integridade seria `number of client Envoys * number of server Envoys * active health check frequency`.

Para resolver esse problema, modifique a frequência da sonda de verificação de integridade, o que reduziria o volume total de sondas de verificação de integridade. Além das verificações de integridade ativas, o App Mesh permite configurar a detecção de discrepâncias como meio de verificação de integridade passiva. Use a detecção de discrepâncias para configurar quando remover um determinado host com base em respostas consecutivas `5xx`.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Solução de problemas de observabilidade do App Mesh

Este tópico detalha problemas comuns que você pode enfrentar com a observabilidade do App Mesh.

Não consigo ver os rastros AWS X-Ray das minhas aplicações

Sintomas

Seu aplicativo no App Mesh não está exibindo informações de rastreamento do X-Ray no console do X-Ray ou nas APIs.

Resolução

Para usar o X-Ray no App Mesh, é necessário configurar corretamente os componentes para permitir a comunicação entre a aplicação, os contêineres auxiliares e o serviço X-Ray. Siga as etapas a seguir para confirmar se o X-Ray foi configurado corretamente:

- Certifique-se de que o protocolo do receptor do nó virtual do App Mesh não esteja definido como `TCP`.
- Certifique-se de que o contêiner X-Ray implantado com sua aplicação exponha a porta UDP `2000` e seja executado como usuário `1337`. Para obter mais informações, consulte o exemplo do Amazon ECS X-Ray em GitHub.

- Certifique-se de que o contêiner Envoy tenha o rastreamento ativado. Se estiver usando a imagem do App Mesh Envoy, você pode ativar o X-Ray definindo a variável de `ENABLE_ENVOY_XRAY_TRACING` ambiente como um valor de `1` e a variável de `XRAY_DAEMON_PORT` ambiente como `2000`.
- Se você instrumentou o X-Ray no código da sua aplicação com um dos SDKs específicos de idiomas, certifique-se de que ele esteja configurado corretamente seguindo os guias para seu idioma.
- Se todos os itens anteriores estiverem configurados corretamente, revise os logs do contêiner X-Ray em busca de erros e siga as orientações em Solução de problemas AWS X-Ray. Uma explicação mais detalhada da integração do X-Ray no App Mesh pode ser encontrada em Integrando o X-Ray com o App Mesh.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não consigo ver as métricas do Envoy para meus aplicativos nas métricas da Amazon CloudWatch

Sintomas

Seu aplicativo no App Mesh não está emitindo métricas geradas pelo proxy Envoy para métricas. CloudWatch

Resolução

Ao usar CloudWatch métricas no App Mesh, você deve configurar corretamente vários componentes para permitir a comunicação entre o proxy do Envoy, o sidecar do CloudWatch agente e o serviço de métricas. CloudWatch Siga as etapas a seguir para confirmar se as CloudWatch métricas do proxy Envoy foram configuradas corretamente:

- Verifique se você está usando a imagem do CloudWatch agente para o App Mesh. Para obter mais informações, consulte CloudWatchAgente do App Mesh ativado GitHub.
- Certifique-se de ter configurado o CloudWatch agente para o App Mesh adequadamente seguindo as instruções de uso específicas da plataforma. Para obter mais informações, consulte CloudWatchAgente do App Mesh ativado GitHub.

- Se todos os itens anteriores estiverem configurados corretamente, revise os registros do contêiner do CloudWatch agente em busca de erros e siga as orientações fornecidas em Solução de problemas do CloudWatch agente.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível configurar regras de amostragem personalizadas para rastreamentos AWS X-Ray

Sintomas

Sua aplicação está usando o rastreamento X-Ray, mas não foi possível configurar regras de amostragem para seus rastreamentos.

Resolução

Como o App Mesh Envoy atualmente não oferece suporte à configuração de amostragem do Dynamic X-Ray, as seguintes soluções alternativas estão disponíveis.

Se sua versão do Envoy for `1.19.1` ou posterior, você tem as seguintes opções.

- Para definir apenas a taxa de amostragem, use a variável de ambiente `XRAY_SAMPLING_RATE` no contêiner do Envoy. O valor deve ser especificado como um decimal entre `0` e `1.00` (100%). Para ter mais informações, consulte AWS X-Ray variáveis.
- Para configurar as regras de amostragem personalizadas localizadas para o rastreador X-Ray, use a variável de ambiente `XRAY_SAMPLING_RULE_MANIFEST` para especificar um caminho de arquivo no sistema de arquivos do contêiner do Envoy. Para obter mais informações, consulte Regras de amostragem no Guia do desenvolvedor do AWS X-Ray.

Se sua versão do Envoy for anterior a `1.19.1`, faça o seguinte.

- Use a variável de ambiente `ENVOY_TRACING_CFG_FILE` para alterar sua taxa de amostragem. Para ter mais informações, consulte Variáveis de configuração do Envoy. Especifique uma configuração de rastreamento personalizada e defina as regras locais de amostragem. Para obter mais informações, consulte Envoy X-Ray config.
- Configuração de rastreamento personalizada para o exemplo da variável de ambiente `ENVOY_TRACING_CFG_FILE`:

```
tracing:
   http:
     name: envoy.tracers.xray
     typedConfig:
       "@type": type.googleapis.com/envoy.config.trace.v3.XRayConfig
       segmentName: foo/bar
       segmentFields:
         origin: AWS::AppMesh::Proxy
         aws:
           app_mesh:
             mesh_name: foo
             virtual_node_name: bar
       daemonEndpoint:
             protocol: UDP
             address: 127.0.0.1
             portValue: 2000
       samplingRuleManifest:
             filename: /tmp/sampling-rules.json
```

- Para obter detalhes sobre a configuração do manifesto da regra de amostragem na propriedade `samplingRuleManifest`, consulte Configurando o X-Ray SDK para Go.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Solução de problemas de segurança do App Mesh

Este tópico detalha problemas comuns que podem ser enfrentados com a segurança do App Mesh.

Não é possível se conectar a um serviço virtual de back-end com uma política de cliente TLS

Sintomas

Ao adicionar uma política de cliente TLS a um back-end de serviço virtual em um nó virtual, a conectividade com esse back-end falha. Ao tentar enviar tráfego para o serviço de back-end, as solicitações falham com um código de resposta `HTTP 503` e a mensagem de erro: `upstream connect error or disconnect/reset before headers. reset reason: connection failure`.

Resolução

Para determinar a causa raiz do problema, recomendamos usar os registros do processo de proxy do Envoy para ajudá-lo a diagnosticar o problema. Para ter mais informações, consulte Ative o registro de depuração do Envoy em ambientes de pré-produção. Use a lista a seguir para determinar a causa da falha de conexão:

- Certifique-se de que a conectividade com o back-end seja bem-sucedida, descartando os erros mencionados em Não é possível conectar-se a um back-end de serviço virtual.
- Nos logs do processo do Envoy, procure os seguintes erros (logados no nível de depuração).

  ```
  TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
  ```

  Esse erro é causado por um ou mais dos motivos a seguir:

  - O certificado não foi assinado por uma das autoridades de certificação definidas no pacote de confiança da política do cliente TLS.
  - O certificado não é mais válido (expirou).
  - O nome alternativo do assunto (SAN) não corresponde ao nome de host DNS solicitado.
  - Certifique-se de que o certificado oferecido pelo serviço de back-end seja válido, assinado por uma das autoridades de certificação em seu pacote confiável de políticas de cliente TLS e atenda aos critérios definidos em Transport Layer Security (TLS).
  - Se o erro recebido for como o abaixo, isso significa que a solicitação está ignorando o proxy do Envoy e acessando diretamente a aplicação. Ao enviar tráfego, as estatísticas no Envoy não mudam, indicando que o Envoy não está no caminho certo para decifrar o tráfego. Na configuração de proxy do nó virtual, verifique se as `AppPorts` contêm o valor correto que o aplicativo está receptando.

    ```
    upstream connect error or disconnect/reset before headers. reset reason:
     connection failure, transport failure reason: TLS error: 268435703:SSL
     routines:OPENSSL_internal:WRONG_VERSION_NUMBER
    ```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport. Se acredita ter encontrado uma vulnerabilidade de segurança ou se tiver dúvidas sobre a segurança do App Mesh, consulte as diretrizes do relatório de vulnerabilidade da AWS.

Não é possível se conectar a um serviço virtual de back-end quando a aplicação está originando o TLS

Sintomas

Ao originar uma sessão TLS de uma aplicação, em vez do proxy Envoy, a conectividade com um serviço virtual de back-end falha.

Resolução

Esse é um problema conhecido. Para obter mais informações, consulte a Solicitação de recurso: negociação de TLS entre o aplicativo downstream e o problema do proxy upstream. GitHub No App Mesh, a TLS de origem atualmente é suportada pelo proxy Envoy, mas não pela aplicação. Para usar o suporte à originação TLS no Envoy, desative a originação TLS no aplicativo. Isso permite que o Envoy leia os cabeçalhos da solicitação de saída e encaminhe a solicitação para o destino apropriado por meio de uma sessão TLS. Para ter mais informações, consulte Transport Layer Security (TLS).

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport. Se acredita ter encontrado uma vulnerabilidade de segurança ou se tiver dúvidas sobre a segurança do App Mesh, consulte as diretrizes do relatório de vulnerabilidade da AWS.

Não é possível afirmar que a conectividade entre os proxies do Envoy está usando TLS

Sintomas

Sua aplicação habilitou o encerramento de TLS no nó virtual ou no receptor do gateway virtual, ou o TLS de origem na política de cliente TLS de back-end, mas você não consegue afirmar que a conectividade entre os proxies do Envoy está ocorrendo em uma sessão negociada por TLS.

Resolução

As etapas definidas nesta resolução usam a interface de administração do Envoy e as estatísticas do Envoy. Para obter ajuda para configurá-los, consulte Ativar a interface de administração do proxy Envoy e Habilite a integração do Envoy DogStats D para descarga métrica. Os exemplos de estatísticas a seguir usam a interface de administração para simplificar.

- Para o proxy Envoy executando o encerramento do TLS:
  - Certifique-se de que o certificado TLS tenha sido inicializado na configuração do Envoy com o comando a seguir.

    ```
    curl http://my-app.default.svc.cluster.local:9901/certs
    ```

    Na saída retornada, é possível visualizar pelo menos uma entrada em `certificates[].cert_chain` para o certificado usado no terminal TLS.
  - Certifique-se de que o número de conexões de entrada bem-sucedidas com o receptor do proxy seja exatamente o mesmo que o número de handshakes SSL mais o número de sessões SSL reutilizadas, conforme mostrado nos comandos e na saída do exemplo a seguir.

    ```
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep downstream_cx_total
    listener.0.0.0.0_15000.downstream_cx_total: 11
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep ssl.connection_error
    listener.0.0.0.0_15000.ssl.connection_error: 1
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep ssl.handshake
    listener.0.0.0.0_15000.ssl.handshake: 9
    curl -s http://my-app.default.svc.cluster.local:9901/stats | grep
     "listener.0.0.0.0_15000" | grep ssl.session_reused
    listener.0.0.0.0_15000.ssl.session_reused: 1
    # Total CX (11) - SSL Connection Errors (1) == SSL Handshakes (9) + SSL Sessions
     Re-used (1)
    ```
- Para o proxy Envoy que executa a origem do TLS:
  - Certifique-se de que o armazenamento confiável do TLS tenha sido inicializado na configuração do Envoy com o comando a seguir.

    ```
    curl http://my-app.default.svc.cluster.local:9901/certs
    ```

    Deve ser visualizada pelo menos uma entrada abaixo `certificates[].ca_certs` para os certificados usados na validação do certificado do back-end durante a origem do TLS.
  - Certifique-se de que o número de conexões de saída bem-sucedidas com o cluster de back-end seja exatamente o mesmo que o número de handshakes SSL mais o número de sessões SSL reutilizadas, conforme mostrado nos seguintes exemplos de comandos e resultados.

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-
name" | grep upstream_cx_total
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.upstream_cx_total: 11
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-
name" | grep ssl.connection_error
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.connection_error:
 1
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-
name" | grep ssl.handshake
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.handshake: 9
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "virtual-node-
name" | grep ssl.session_reused
cluster.cds_egress_mesh-name_virtual-node-name_protocol_port.ssl.session_reused: 1
# Total CX (11) - SSL Connection Errors (1) == SSL Handshakes (9) + SSL Sessions
 Re-used (1)
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport. Se acredita ter encontrado uma vulnerabilidade de segurança ou se tiver dúvidas sobre a segurança do App Mesh, consulte as diretrizes do relatório de vulnerabilidade da AWS.

Solução de problemas do TLS com o Elastic Load Balancing

Sintomas

Ao tentar configurar um Application Load Balancer ou Network Load Balancer para criptografar o tráfego em um nó virtual, as verificações de integridade e conectividade do balanceador de carga podem falhar.

Resolução

Para determinar a causa raiz do problema, é preciso verificar o seguinte:

- Para que o proxy Envoy execute a terminação de TLS, é preciso descartar qualquer configuração incorreta. Siga as etapas fornecidas acima no Não é possível se conectar a um serviço virtual de back-end com uma política de cliente TLS.
- Para o balanceador de carga, é preciso examinar a configuração do `TargetGroup`:
  - Certifique-se de que a `TargetGroup` porta corresponda à porta do receptor definida pelo nó virtual.

- Para Application Load Balancers que estão originando conexões TLS via HTTP para seu serviço, verifique se o protocolo `TargetGroup` está definido como `HTTPS`. Se as verificações de integridade estiverem sendo utilizadas, verifique se a `HealthCheckProtocol` está definida como `HTTPS`.
- Para Network Load Balancers que estão originando conexões TLS via TCP para seu serviço, verifique se o protocolo `TargetGroup` está definido como `TLS`. Se as verificações de integridade estiverem sendo utilizadas, verifique se a `HealthCheckProtocol` está definida como `TCP`.

> **Note**
>
> Qualquer atualização do `TargetGroup` exige a alteração do nome do `TargetGroup`.

Com essa configuração correta, seu balanceador de carga deve fornecer uma conexão segura ao seu serviço usando o certificado fornecido ao proxy Envoy.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport. Se acredita ter encontrado uma vulnerabilidade de segurança ou se tiver dúvidas sobre a segurança do App Mesh, consulte as diretrizes do relatório de vulnerabilidade da AWS.

Solução de problemas do App Mesh para o Kubernetes

Este tópico detalha problemas comuns que podem ser enfrentados ao usar o App Mesh com o Kubernetes.

Os recursos do App Mesh criados no Kubernetes não podem ser encontrados no App Mesh

Sintomas

Você criou os recursos do App Mesh usando a definição personalizada de recursos (CRD) do Kubernetes, mas os recursos criados não são visíveis no App Mesh quando usa as APIs ou o AWS Management Console.

Resolução

A causa provável é um erro no controlador Kubernetes para o App Mesh. Para obter mais informações, consulte Solução de problemas em GitHub. Verifique se há erros ou avisos nos registros do controlador que indiquem que o controlador não conseguiu criar nenhum recurso.

```
kubectl logs -n appmesh-system -f \
    $(kubectl get pods -n appmesh-system -o name | grep controller)
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Os pods estão falhando nas verificações de prontidão e liveness depois que o sidecar do Envoy é injetado

Sintomas

Anteriormente, o pod do seu aplicativo foi executado com êxito, mas depois que o sidecar do Envoy é injetado em um pod, as verificações de prontidão e liveness começam a falhar.

Resolução

Certifique-se de que o contêiner Envoy que foi injetado no pod tenha sido inicializado com o serviço de gerenciamento Envoy do App Mesh. É possível verificar quaisquer erros referenciando os códigos de erro em Envoy desconectado do serviço de gerenciamento do App Mesh Envoy com texto de erro. É possível usar o comando a seguir para inspecionar os registros do Envoy para o pod relevante.

```
kubectl logs -n appmesh-system -f \
    $(kubectl get pods -n appmesh-system -o name | grep controller) \
    | grep "gRPC config stream closed"
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Os pods não estão se registrando ou cancelando o registro como instâncias do AWS Cloud Map

Sintomas

Seus pods do Kubernetes não estão sendo registrados ou cancelados do AWS Cloud Map como parte de seu ciclo de vida. Um pod pode iniciar com sucesso e estar pronto para atender ao tráfego,

mas não receber nenhum. Quando um pod é encerrado, os clientes ainda podem reter seu endereço IP e tentar enviar tráfego para ele, falhando.

Resolução

Esse é um problema conhecido. Para obter mais informações, consulte o problema em que os pods não são registrados/cancelados automaticamente no Kubernetes. AWS Cloud Map GitHub Devido à relação entre pods, nós virtuais do App Mesh e recursos do AWS Cloud Map, o controlador do App Mesh para Kubernetes pode ficar dessincronizado e perder recursos. Por exemplo, isso pode acontecer se um recurso de nó virtual for excluído do Kubernetes antes de encerrar os pods associados.

Para mitigar esse problema:

- Verifique se está executando a versão mais recente do controlador do App Mesh para Kubernetes.
- Verifique se AWS Cloud Map `namespaceName` e `serviceName` estão corretos na definição do nó virtual.
- Certifique-se de excluir todos os pods associados antes de excluir a definição do nó virtual. Se precisar de ajuda para identificar quais pods estão associados a um nó virtual, consulte Não é possível determinar onde um pod para um recurso do App Mesh está em execução.
- Se o problema persistir, execute o comando a seguir para inspecionar os registros do controlador em busca de erros que possam ajudar a revelar o problema subjacente.

```
kubectl logs -n appmesh-system \
    $(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

- Considere usar o comando a seguir para reiniciar os pods do controlador. Isso pode corrigir problemas de sincronização.

```
kubectl delete -n appmesh-system \
    $(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível determinar onde um pod para um recurso do App Mesh está em execução

Sintomas

Ao executar o App Mesh em um cluster do Kubernetes, um operador não pode determinar onde uma workload, ou pod, está sendo executada para um determinado recurso do App Mesh.

Resolução

Os recursos do pod do Kubernetes são anotados com a malha e o nó virtual aos quais estão associados. É possível consultar quais pods estão sendo executados para um determinado nome de nó virtual com o comando a seguir.

```
kubectl get pods --all-namespaces -o json | \
    jq '.items[] | { metadata } | select(.metadata.annotations."appmesh.k8s.aws/
virtualNode" == "virtual-node-name")'
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Não é possível determinar em qual recurso do App Mesh um pod está sendo executado

Sintomas

Ao executar o App Mesh em um cluster Kubernetes, um operador não pode determinar com qual recurso do App Mesh um determinado pod está sendo executado.

Resolução

Os recursos do pod do Kubernetes são anotados com a malha e o nó virtual aos quais estão associados. É possível gerar os nomes da malha e do nó virtual consultando o pod diretamente usando o comando a seguir.

```
kubectl get pod pod-name -n namespace -o json | \
    jq '{ "mesh": .metadata.annotations."appmesh.k8s.aws/mesh",
 "virtualNode": .metadata.annotations."appmesh.k8s.aws/virtualNode" }'
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Os Envoys do cliente não conseguem se comunicar com o App Mesh Envoy Management Service com o IMDSv1 desativado

Sintomas

Quando o `IMDSv1` está desativado, os Envoys do cliente não conseguem se comunicar com o ambiente de gerenciamento do App Mesh (Envoy Management Service). O suporte `IMDSv2` não está disponível na versão do App Mesh Envoy anterior a `v1.24.0.0-prod`.

Resolução

Para resolver esse problema, você pode realizar uma das três seguintes ações.

- Atualize para a versão App Mesh Envoy `v1.24.0.0-prod` ou posterior, que tem suporte `IMDSv2`.
- Reative o `IMDSv1` na instância em que o Envoy está sendo executado. Para obter instruções sobre restauração `IMDSv1`, consulte Configurar as opções de metadados da instância.
- Se seus serviços estiverem em execução no Amazon EKS, é recomendável usar perfis do IAM para contas de serviço (IRSA) para obter credenciais. Para obter instruções sobre como ativar o IRSA, consulte Perfis do IAM para contas de serviço.

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

O IRSA não funciona no contêiner da aplicação quando o App Mesh está ativado e o Envoy é injetado

Sintomas

Quando o App Mesh é habilitado em um cluster do Amazon EKS com a ajuda do controlador App Mesh para o Amazon EKS, o Envoy e os contêineres `proxyinit` são injetados no pod da aplicação. A aplicação não é capaz de assumir o `IRSA` e, em vez disso, assume o `node role`. Ao descrevemos os detalhes do pod, percebemos que tanto a variável de ambiente `AWS_WEB_IDENTITY_TOKEN_FILE` quanto a `AWS_ROLE_ARN` não estão incluídas no contêiner do aplicação.

Resolução

Se uma variáveis de ambiente `AWS_WEB_IDENTITY_TOKEN_FILE` ou `AWS_ROLE_ARN` forem definidas, o webhook ignorará o pod. Não forneça nenhuma dessas variáveis e o webhook se encarregará de injetá-las para você.

```
reservedKeys := map[string]string{
        "AWS_ROLE_ARN":                "",
        "AWS_WEB_IDENTITY_TOKEN_FILE": "",
    }
    ...
    for _, env := range container.Env {
        if _, ok := reservedKeys[env.Name]; ok {
            reservedKeysDefined = true
        }
```

Se o problema ainda não tiver sido resolvido, considere abrir um GitHub problema ou entre em contato com o AWSSupport.

Canal de demonstração do App Mesh

O Canal de demonstração do App Mesh é uma variante distinta do serviço App Mesh fornecido na região `us-west-2`. O Canal de demonstração expõe os próximos recursos para você experimentar à medida que são desenvolvidos. Ao usar recursos no Canal de demonstração, você pode fornecer feedback via GitHub para definir como os recursos se comportam. Quando um atributo tiver a funcionalidade completa no Canal de demonstração, com todas as integrações e verificações necessárias concluídas, ele passará para o serviço App Mesh de produção.

O AWS App Mesh Canal de demonstração é um serviço beta e todos os recursos são demonstrações, conforme definido nos Termos de serviço da AWS. Sua participação no Canal de demonstração é regida pelo seu Contrato com a AWS e pelos Termos de serviço da AWS, em particular, pelos termos de Participação no serviço universal e beta, e é confidencial.

As perguntas a seguir são frequentes sobre o Canal de demonstração.

O que é o Canal de demonstração?

O Canal de demonstração é um endpoint de serviço público que permite experimentar e fornecer feedback sobre os novos atributos do serviço antes de estarem disponíveis ao público. O endpoint de serviço do Canal de demonstração é separado do endpoint de produção padrão. Você interage com o endpoint usando a AWS CLI, um arquivo de modelo de serviço para o Canal de demonstração e arquivos de entrada de comando para a AWS CLI. O Canal de demonstração permite experimentar novos recursos sem afetar sua infraestrutura de produção atual. Recomendamos que você forneça feedback à equipe do App Mesh para ajudar a garantir que o App Mesh atenda aos requisitos mais importantes dos clientes. Seu feedback sobre os recursos enquanto estão no Canal de demonstração pode ajudar a moldar os recursos do App Mesh para que possamos oferecer o melhor serviço possível.

Como o Canal de demonstração é diferente do App Mesh de produção?

A tabela a seguir lista aspectos do serviço App Mesh que são diferentes do Canal de demonstração.

Aspecto	Serviço de produção do App Mesh	Serviço do canal de demonstração do App Mesh

Frontend endpoint	appmesh.us-west-2.amazonaws.com	appmesh-preview.us-west-2.amazonaws.com
Envoy management service endpoint	appmesh-envoy-management.us-west-2.amazonaws.com	appmesh-preview-envoy-management.us-west-2.amazonaws.com
CLI	AWS App Mesh list-meshes	AWS App Mesh-preview list-meshes (only available after adding the Preview Channel service model)
Signing name	appmesh	appmesh-preview
Service principal	appmesh.amazonaws.com	appmesh-preview.amazonaws.com

> **Note**
>
> Embora o exemplo na tabela do serviço de produção do App Mesh liste a região `us-west-2`, o serviço de produção está disponível na maioria das regiões. Para ver uma lista de todas as regiões nas quais o serviço de produção do App Mesh está disponível, consulte Endpoints e cotas do AWS App Mesh. No entanto, o serviço do Canal de demonstração do App Mesh está disponível somente na região `us-west-2`.

Como posso usar os recursos do Canal de demonstração?

1. Adicione o modelo de serviço do Canal de demonstração que inclui o atributo Canal de demonstração à AWS CLI com o comando a seguir.

```
aws configure add-model \
    --service-name appmesh-preview \
    --service-model https://raw.githubusercontent.com/aws/aws-app-mesh-roadmap/main/appmesh-preview/service-model.json
```

2. Crie um arquivo JSON que inclua o recurso, com base no exemplo JSON e nas instruções fornecidas no Guia do usuário AWS App Mesh para o atributo.

3. Implemente o atributo com o comando AWS CLI e o arquivo de entrada de comando apropriados. Por exemplo, o comando a seguir cria uma rota com os recursos do Canal de demonstração usando o arquivo *route.json*.

```
aws appmesh-preview create-route --cli-input-json file://route.json
```

4. Adicione `APPMESH_PREVIEW = 1` como variável de configuração para o contêiner Envoy ao adicioná-lo às suas definições de tarefas do Amazon ECS, às especificações do Kubernetes Pod ou às instâncias do Amazon EC2. Essa variável permite que o contêiner Envoy se comunique com os endpoints do Canal de demonstração. Para obter mais informações sobre a adição de variáveis de configuração, consulte Atualização de serviços no Amazon ECS, Atualização de serviços no Kubernetes e Atualização de serviços no Amazon EC2.

Como fornecer feedback?

É possível fornecer comentários diretamente no problema do roteiro do App Mesh no Repositório do GitHub, que é vinculado à documentação sobre o atributo.

Quanto tempo tenho para fornecer feedback sobre um atributo do Canal de demonstração?

O período de feedback varia de acordo com o tamanho e a complexidade do atributo que está sendo introduzido. Pretendemos dar um período de comentários de 14 dias entre o lançamento de um recurso no endpoint de pré-visualização e o lançamento do atributo para produção. A equipe do App Mesh pode estender o período de feedback para recursos específicos.

Qual nível de suporte é fornecido ao Canal de demonstração?

Embora incentivemos a fornecer feedback e relatórios de bugs diretamente no problema do roteiro no GitHub do App Mesh, entendemos que pode ter dados confidenciais para compartilhar, ou pode encontrar um problema que não considere seguro divulgar publicamente. Para esses problemas, você pode entrar em contato com AWS Support ou dar feedback enviando um e-mail diretamente para a equipe do App Mesh.

Meus dados estão seguros no endpoint do Canal de demonstração?

Sim. O Canal de demonstração tem o mesmo nível de segurança do endpoint de produção padrão.

Por quanto tempo minha configuração ficará disponível?

É possível trabalhar com uma malha no Canal de demonstração por trinta dias. Trinta dias após a criação de uma malha, só é possível listar, ler ou excluir a malha. Se tentar criar ou atualizar recursos após trinta dias, receberá uma `BadRequest` de exceção explicando que a malha está arquivada.

Quais ferramentas posso usar para trabalhar com o Canal de demonstração?

É possível usar a AWS CLI com um arquivo de modelo de serviço do Canal de demonstração e arquivos de entrada de comando. Para obter mais informações sobre como trabalhar com recursos, consulte Como posso usar os recursos do Canal de demonstração?. É possível usar as opções de comando da AWS CLI, o AWS Management Console, os SDKs ou o AWS CloudFormation para trabalhar com os recursos do Canal de demonstração. No entanto, é possível usar todas as ferramentas, uma vez que um atributo é lançado para o serviço de produção.

Haverá compatibilidade progressiva das APIs do Canal de demonstração?

Não, as APIs podem mudar com base no feedback.

Os recursos do Canal de demonstração são completos?

Não. Novos objetos de API podem não estar totalmente integrados ao AWS Management Console, AWS CloudFormation ou AWS CloudTrail. À medida que os recursos se solidificam no Canal de demonstração e se aproximam da disponibilidade geral, as integrações eventualmente serão disponibilizadas.

Há documentação disponível para os recursos do Canal de demonstração?

Sim. A documentação dos recursos do Canal de demonstração está incluída na documentação de produção. Por exemplo, se os recursos do recurso de rota forem liberados para o Canal de demonstração, as informações sobre como usar os recursos estarão no documento existente do recurso de rota. Os recursos do Canal de demonstração são rotulados como disponíveis somente no Canal de demonstração.

Como saber quando novos recursos estão disponíveis no Canal de demonstração?

Quando novos recursos são introduzidos no Canal de demonstração, uma entrada é adicionada ao Histórico de documentos do App Mesh. Você pode revisar a página regularmente ou assinar o Feed RSS do Histórico de documentos do App Mesh. Além disso, você pode analisar os problemas do roteiro do AWS App Mesh no Repositório do GitHub. Um link para download do arquivo JSON do modelo de serviço do Canal de demonstração é adicionado a um problema quando ele é lançado no Canal de demonstração. Para obter mais informações sobre como usar o modelo e o atributo, consulte Como posso usar os recursos do Canal de demonstração?.

Como saber quando um atributo foi promovido para o serviço de produção?

O texto na documentação do App Mesh observando que o atributo está disponível somente no Canal de demonstração é removido e uma entrada é adicionada ao Histórico de documentos do App Mesh. Você pode revisar a página regularmente ou assinar o Feed RSS do Histórico de documentos do App Mesh.

Service Quotas do App Mesh

O AWS App Mesh foi integrado ao Service Quotas, um serviço da AWS que permite visualizar e gerenciar as cotas em um local central. As Service Quotas também são referidas como limites. Para obter mais informações, consulte O que são Service Quotas? no Guia do usuário do Service Quotas.

O Service Quotas facilita a pesquisa do valor de todas as cotas de serviço do App Mesh.

Como exibir cotas de serviço no App Mesh usando a AWS Management Console

1. Abra o console do Service Quotas em https://console.aws.amazon.com/servicequotas/.
2. No painel de navegação, escolha AWSServiços da .
3. Na lista de serviços da AWS, procure e selecione AWS App Mesh.

   Na lista Service quotas (Cotas de serviço) é possível ver o nome da cota de serviço, o valor aplicado (se estiver disponível), a cota da AWS padrão e se o valor da cota é ajustável.
4. Para visualizar informações adicionais sobre uma cota de serviço, como a descrição, escolha o nome da cota.

Para solicitar o aumento da cota, consulte Requesting a Quota Increase (Solicitação de aumento de cota) no Guia do usuário do Service Quotas.

Como exibir cotas de serviço no App Mesh usando a AWS CLI

Execute o comando a seguir.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code appmesh \
    --output table
```

Para trabalhar mais com cotas de serviço usando a AWS CLI, consulte a Referência de comandos da AWS CLI do Service Quotas.

Histórico da documentação do App Mesh

A tabela a seguir descreve as principais atualizações e novos atributos para o Guia do usuário do AWS App Mesh . Também atualizamos a documentação com frequência para abordar os comentários enviados por você.

Alteração	Descrição	Data
AWSAppMeshFullAccess Política atualizada	Atualizado `AWSAppMeshFullAccess` para permitir o acesso às `UntagResource` APIs `TagResource` e.	24 de abril de 2024
CloudTrail documentação de integração atualizada	A documentação que descreve a integração do App Mesh com CloudTrail a atividade da API para registrar foi atualizada.	28 de março de 2024
Políticas atualizadas	Atualizado `AWSServiceRoleForAppMesh` e `AWSAppMeshServiceRolePolicy` para permitir o acesso à AWS Cloud Map `DiscoverInstancesRevision` API.	12 de outubro de 2023
Suporte à política do endpoint da VPC para o App Mesh	O App Mesh é compatível com políticas de endpoint da VPC	11 de maio de 2023
Vários receptores para o App Mesh	O App Mesh agora oferece suporte a vários receptores.	18 de agosto de 2022
IPv6 para o App Mesh	O App Mesh agora oferece suporte a IPv6.	18 de maio de 2022

CloudTrail suporte de registro para o App Mesh Envoy Management Service	O App Mesh agora oferece suporte de CloudTrail registro para o App Mesh Envoy Management Service.	18 de março de 2022
Agente do App Mesh para Envoy	O App Mesh agora oferece suporte ao Agente para Envoy.	25 de fevereiro de 2022
Vários receptores para o App Mesh	(Somente para o Canal de demonstração do App Mesh) Você pode implementar vários receptores para o App Mesh.	23 de novembro de 2021
Suporte ao ARM64 para App Mesh	O App Mesh agora oferece suporte ao ARM64.	19 de novembro de 2021
Extensão de métricas para o App Mesh	Você pode implementar extensões de métricas para o App Mesh.	29 de outubro de 2021
Implementação de aprimoramentos de tráfego de entrada	Você pode implementar a correspondência do nome do host e do cabeçalho e regravar o nome e o caminho do host.	14 de junho de 2021
Implementação de autenticação de TLS mútuo	Você pode implementar a autenticação de TLS mútuo.	4 de fevereiro de 2021
Lançamento regional em af-south-1	App Mesh não está disponível na Região af-south-1.	22 de janeiro de 2021
Implementação de autenticação de TLS mútuo	(Somente para o Canal de demonstração do App Mesh) Você pode implementar a autenticação de TLS mútuo.	23 de novembro de 2020

Implementação do grupo de conexões para um receptor de gateway virtual	Você pode implementar o grupo de conexões para um receptor de gateway virtual.	5 de novembro de 2020
Implementação do grupo de conexões e a detecção de discrepâncias para um receptor de nó virtual	Você pode implementar o grupo de conexões e detecção de discrepâncias para um receptor de nó virtual.	5 de novembro de 2020
Lançamento regional em eu-south-1	App Mesh não está disponível na Região eu-south-1.	21 de outubro de 2020
Implementação do grupo de conexões para um receptor de gateway virtual	(Somente para o Canal de demonstração do App Mesh) Você pode implementar o grupo de conexões para um receptor de gateway virtual.	28 de setembro de 2020
Implementação do grupo de conexões e a detecção de discrepâncias para um receptor de nó virtual	(Somente para o Canal de demonstração do App Mesh) Você pode implementar o grupo de conexões e detecção de discrepâncias para um receptor de nó virtual.	28 de setembro de 2020
Criação de um gateway virtual e uma rota de gateway para entrada em malha	Permissão para que recursos fora da malha se comunique m com recursos que estão dentro da malha.	10 de julho de 2020
Criação e gerenciamento de recursos do App Mesh de dentro do Kubernetes com o controlador do App Mesh para Kubernetes	Você pode criar e gerenciar recursos do App Mesh no Kubernetes. O controlador também injeta automatic amente o proxy Envoy e os contêineres de init nos pods que você implanta.	18 de junho de 2020

Adição de um valor de tempo limite a um receptor de nó virtual e rota	Você pode adicionar um valor de tempo limite a um receptor de nó virtual e a uma rota.	18 de junho de 2020
Adição de um valor de tempo limite a um receptor de nó virtual	(Somente para o Canal de demonstração do App Mesh) Você pode adicionar um valor de tempo limite a um receptor de nó virtual.	29 de maio de 2020
Criação de um gateway virtual para entrada em malha	(Somente para o Canal de demonstração do App Mesh) Permite que recursos fora de uma malha se comuniquem com recursos dentro de uma malha.	8 de abril de 2020
Criptografia TLS	(Somente para o App Mesh Preview Channel) Use certificados de uma autoridade de certificação AWS Private Certificate Authority ou de sua própria autoridade de certificação para criptografar a comunicação entre nós virtuais usando TLS.	17 de janeiro de 2020
Compartilhamento de uma malha com outra conta	(Somente para o Canal de demonstração do App Mesh) Você pode compartilhar uma malha com outra conta. Os recursos criados por qualquer conta podem se comunicar com outros recursos na malha.	17 de janeiro de 2020

Adição de um valor de tempo limite a uma rota	(Somente para o Canal de demonstração do App Mesh) Você pode adicionar um valor de tempo limite a uma rota.	17 de janeiro de 2020
Crie um proxy App Mesh em um AWS Outpost	Você pode criar um proxy App Mesh Envoy em um AWS Outpost.	3 de dezembro de 2019
Suporte a HTTP/2 e gRPC para rotas, roteadores virtuais e nós virtuais	Você pode rotear o tráfego que usa os protocolos HTTP/2 e gRPC. Você também pode adicionar um receptor para esses protocolos a nós virtuais e roteadores virtuais.	25 de outubro de 2019
Política de repetição	Uma política de repetição permite que os clientes se protejam contra falhas de rede intermitentes ou falhas intermitentes no lado do servidor. Você pode adicionar a lógica de repetição a uma rota.	10 de setembro de 2019
Criptografia TLS	(Somente para o Canal de demonstração do App Mesh) Criptografia da comunicação entre nós virtuais usando TLS.	6 de setembro de 2019
Roteamento baseado no cabeçalho HTTP	Roteamento de tráfego com base na presença e nos valores dos cabeçalhos HTTP em uma solicitação.	15 de agosto de 2019

Disponibilidade do Canal de demonstração do App Mesh	O Canal de demonstração do App Mesh é uma variante distinta do serviço App Mesh. O Canal de demonstração expõe os próximos recursos para você experimentar à medida que são desenvolv idos. Ao usar recursos no Preview Channel, você pode fornecer feedback GitHub para definir como os recursos se comportam.	19 de julho de 2019
Endpoints VPC da interface App Mesh ()AWS PrivateLink	Melhore a postura de segurança da sua VPC configurando o App Mesh para usar um endpoint da VPC de interface. Os endpoints de interface são alimentad os por AWS PrivateLink, uma tecnologia que permite acessar de forma privada as APIs do App Mesh usando endereços IP privados. PrivateLink restringe todo o tráfego de rede entre sua VPC e o App Mesh para a rede Amazon.	14 de junho de 2019

Adicionado AWS Cloud Map como um método de descoberta de serviços de nós virtuais	Você pode especificar o DNS ou AWS Cloud Map como um método de descobert a de serviços de nó virtual. Para usar a AWS Cloud Map na descoberta de serviços, sua conta deve ter a função vinculada ao serviço App Mesh.	13 de junho de 2019
Criação de recursos do App Mesh automaticamente no Kubernetes	Criação de recursos do App Mesh e adição automátic a de imagens do contêiner auxiliar do App Mesh às suas implantações do Kubernetes ao criar recursos no Kubernete s.	11 de junho de 2019
Disponibilidade geral do App Mesh	O App Mesh agora está amplamente disponível para uso em produção.	27 de março de 2019
Atualização da API do App Mesh	As APIs do App Mesh foram atualizadas para melhorar a usabilidade. Para obter mais informações, consulte [ATRIBUTO] Adicionar receptores a roteadore s virtuais e [BUG] Rotas a nós virtuais de destino com blackholes de portas incompatíveis.	7 de março de 2019
Lançamento inicial do App Mesh	Documentação inicial para pré-visualização pública do serviço	28 de novembro de 2018

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.