



Guide de l'utilisateur

AWS App Mesh



AWS App Mesh: Guide de l'utilisateur

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

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

Table of Contents

Qu'est-ce qu'AWS App Mesh ?	1
Ajouter App Mesh à un exemple d'application	1
Composants de l'App Mesh	2
Comment démarrer	4
Accès à App Mesh	4
Démarrer	6
App Mesh et Amazon ECS	6
Scénario	6
Prérequis	7
Étape 1 : Créer un maillage et un service virtuel	8
Étape 2 : Créer un nœud virtuel	9
Étape 3 : Créer un routeur virtuel et un routage	10
Étape 4 : vérifier et créer	12
Étape 5 : Créer des ressources supplémentaires	13
Étape 6 : Mettre à jour les services	19
Rubriques avancées	36
App Mesh et Kubernetes	36
Prérequis	37
Étape 1 : Installer les composants d'intégration	38
Étape 2 : Déployer les ressources App Mesh	44
Étape 3 : Créer ou mettre à jour des services	58
Étape 4 : Nettoyer	65
App Mesh et Amazon EC2	66
Scénario	6
Prérequis	7
Étape 1 : Créer un maillage et un service virtuel	67
Étape 2 : Créer un nœud virtuel	68
Étape 3 : Créer un routeur virtuel et un routage	10
Étape 4 : vérifier et créer	12
Étape 5 : Créer des ressources supplémentaires	13
Étape 6 : Mettre à jour les services	19
App Mesh	89
App Mesh	89
Concepts	90

Maillages	90
Créer un mesh de services	90
Supprimer un maillage	93
Services virtuels	94
Création d'un service virtuel	95
Supprimer un service virtuel	97
Passerelle	99
Création d'une passerelle virtuelle	100
Déploiement une passerelle virtuelle	105
Supprimer une passerelle virtuelle	106
itinéraires de passerelle	107
Nœuds virtuels	114
Création d'un nœud virtuel	115
Supprimer un nœud virtuel	126
Routeurs virtuels	127
Création d'un routeur virtuel	128
Supprimer un routeur virtuel	130
Acheminements	132
Envoy	144
Variantes d'image Envoy	144
Variables de configuration Envoy	149
Variables obligatoires	149
Variables facultatives	150
Paramètres par défaut d'Envoy définis par App Mesh	157
Politique de nouvelle tentative d'itinéraire par défaut	157
Disjoncteur par défaut	159
Mise à jour/migration vers Envoy 1.17	159
Service de découverte secrète avec SPIRE	159
Modifications des expressions régulières	160
Références rétrospectives	162
Agent pour Envoyé	163
Observabilité	165
Journalisation	165
Firelens et Cloudwatch	167
Métriques Envoy	168
Exemples de métriques d'application	171

Exportation des métriques	174
Tracing	183
X-Ray	183
Jaeger	185
Datadog pour le suivi	153
Outillage	187
AWS CloudFormation	187
AWS CDK	187
Contrôleur App Mesh pour Kubernetes	187
Terraform	188
Utilisation de maillages partagés	189
Octroi d'autorisations pour partager des maillages	189
Octroi de l'autorisation de partager un maillage	189
Octroi d'autorisations pour un maillage	190
Conditions préalables au partage de maillages	191
Services connexes	192
Partage d'un maillage	192
Annulation du partage d'un maillage partagé	193
Identifier un maillage partagé	194
Facturation et mesures	194
Quotas d'instances	194
Utilisation avec d'autres services	196
Création de ressources App Mesh avecAWS CloudFormation	196
App Mesh etAWS CloudFormation modèles	196
En savoir plus sur AWS CloudFormation	197
App Mesh surAWS Outposts	197
Prérequis	197
Limites	197
Considérations relatives à la connectivité réseau	198
Création d'un proxy App Mesh Envoy sur un Outpost	198
Bonnes pratiques	200
Instrumenter tous les itinéraires avec de nouvelles tentatives	200
Ajuster la vitesse de déploiement	201
Élargir avant d'intégrer	202
Mettre en œuvre des contrôles de santé des conteneurs	202
Optimisation de la résolution DNS	203

Sécurisation des applications	204
protocole TLS (Transport Layer Security)	205
Exigences du certificat	206
Certificats d'authentification TLS	206
Comment App Mesh configure Envoy pour négocier le TLS	209
Vérifier le chiffrement	210
Renouvellement des certificats	211
Configurer les charges de travail Amazon ECS pour utiliser l'authentification TLS avec AWS App Mesh	212
Configurer les charges de travail Kubernetes pour utiliser l'authentification TLS avec AWS App Mesh	212
Authentification TLS mutuelle	213
Certificats d'authentification TLS mutuels	214
Configurer les points de terminaison du maillage	214
Migrer les services vers l'authentification TLS mutuelle	215
Vérification de l'authentification TLS mutuelle	216
Procédures pas à pas de l'authentification TLS mutuelle App Mesh	216
Gestion des identités et des accès	217
Public ciblé	217
Authentification par des identités	218
Gestion des accès à l'aide de politiques	221
Comment AWS App Mesh fonctionne avec IAM	224
Exemples de politiques basées sur l'identité	229
AWS politiques gérées	234
Utilisation des rôles liés à un service	236
Automatisation du proxy Envoy	240
Résolution des problèmes	246
CloudTrail journaux	247
Événements de gestion d'App Mesh dans CloudTrail	249
Exemples d'événements App Mesh	249
Protection des données	250
Chiffrement des données	252
Validation de conformité	252
Sécurité de l'infrastructure	253
Points de terminaison d'un VPC d'interface (AWS PrivateLink)	254
Résilience	256

Reprise après sinistre dans AWS App Mesh	256
Analyse de la configuration et des vulnérabilités	256
Résolution des problèmes	258
Bonnes pratiques	258
Activer l'interface d'administration du proxy Envoy	258
Activer l'intégration d'Envoy DogStats D pour le téléchargement métrique	259
Activer les journaux d'accès	259
Activer la journalisation du débogage d'Envoy dans les environnements de pré-production .	259
Surveillez la connectivité du proxy Envoy avec le plan de contrôle App Mesh	260
Configuration	260
Impossible d'extraire l'image du conteneur Envoy	260
Impossible de se connecter au service de gestion App Mesh Envoy	261
Envoy s'est déconnecté du service de gestion App Mesh Envoy avec un texte d'erreur	262
Échec de la vérification de l'état du conteneur Envoy, de la sonde de disponibilité ou de la sonde de vivacité	266
Health : le contrôle de santé entre l'équilibreur de charge et le point de terminaison du maillage échoue	266
La passerelle virtuelle n'accepte pas le trafic sur les ports 1024 ou moins	267
Connectivité	268
Impossible de résoudre le nom DNS d'un service virtuel	268
Impossible de se connecter à un backend de service virtuel	269
Impossible de se connecter à un service externe	271
Impossible de se connecter à un serveur MySQL ou SMTP	271
Impossible de se connecter à un service modélisé comme un nœud virtuel TCP ou un routeur virtuel dans App Mesh	273
La connectivité réussit à ce que le service ne soit pas répertorié en tant que backend de service virtuel pour un nœud virtuel	273
Certaines demandes échouent avec le code d'état HTTP 503 lorsqu'un service virtuel dispose d'un fournisseur de nœuds virtuels	274
Impossible de se connecter à un système de fichiers Amazon EFS	275
La connectivité réussit à assurer le service, mais la demande entrante n'apparaît pas dans les journaux d'accès, les traces ou les métriques d'Envoy	275
La définition des variables d'HTTPS_PROXY/environnement HTTP_PROXY/au niveau du conteneur ne fonctionne pas comme prévu.	276
Expiration des demandes en amont, même après avoir défini le délai d'expiration pour les itinéraires.	277

Envoy répond par une requête HTTP Bad.	278
Impossible de configurer correctement le délai d'expiration.	278
Mise à l'échelle	279
La connectivité échoue et les vérifications de l'état du conteneur échouent lorsque le dimensionnement dépasse les 50 répliques pour un nœud virtuel/une passerelle virtuelle ...	279
Les demandes échouent 503 lorsqu'un backend de service virtuel évolue horizontalement vers l'extérieur ou vers l'intérieur	279
Le conteneur Envoy se bloque avec segfault en cas de charge accrue	280
L'augmentation des ressources par défaut n'est pas reflétée dans les limites de service	280
L'application se bloque en raison d'un grand nombre d'appels de bilan de santé.	281
Observabilité	281
Impossible de voir les AWS X-Ray traces de mes applications	282
Impossible de voir les métriques Envoy pour mes applications dans CloudWatch les métriques Amazon	282
Impossible de configurer des règles d'échantillonnage personnalisées pour les AWS X-Ray traces	283
Sécurité	285
Impossible de se connecter à un service virtuel principal avec une politique client TLS	285
Impossible de se connecter à un service virtuel principal lorsque l'application est à l'origine du protocole TLS	286
Impossible d'affirmer que la connectivité entre les proxys Envoy utilise le protocole TLS	287
Résolution des problèmes liés au protocole TLS avec Elastic Load Balancing	289
Kubernetes	290
Les ressources App Mesh créées dans Kubernetes sont introuvables dans App Mesh	290
Les dosettes échouent aux contrôles de préparation et de vivacité après l'injection du sidecar Envoy	290
Les pods ne s'enregistrent pas ou ne se désenregistrent pas en tant qu'instances AWS Cloud Map	291
Impossible de déterminer où s'exécute un pod pour une ressource App Mesh	292
Impossible de déterminer la ressource App Mesh sous laquelle un pod est exécuté	293
Les envoyés clients ne sont pas en mesure de communiquer avec le service de gestion App Mesh Envoy lorsque IMDSv1 est désactivé	293
L'IRSA ne fonctionne pas sur le conteneur d'applications lorsque App Mesh est activé et qu'Envoy est injecté	294
Version préliminaire	295
Service Quotas	301

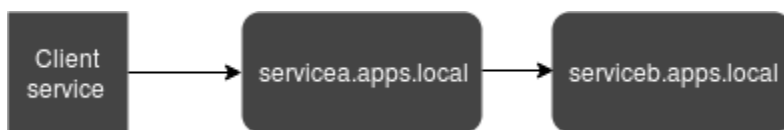
Historique de la documentation	302
.....	CCCX

Qu'est-ce qu'AWS App Mesh ?

AWS App Mesh est un maillage de service qui permet de surveiller et de contrôler facilement les services. Un maillage de services est une couche d'infrastructure dédiée à service-to-service la gestion des communications, généralement via un ensemble de proxys réseau légers déployés parallèlement au code de l'application. App Mesh normalise la façon dont vos services communiquent, vous end-to-end donnant et contribuant à garantir la haute disponibilité de vos applications. App Mesh vous fournit une visibilité constante ainsi que des contrôles de trafic réseau pour chaque service dans une application.

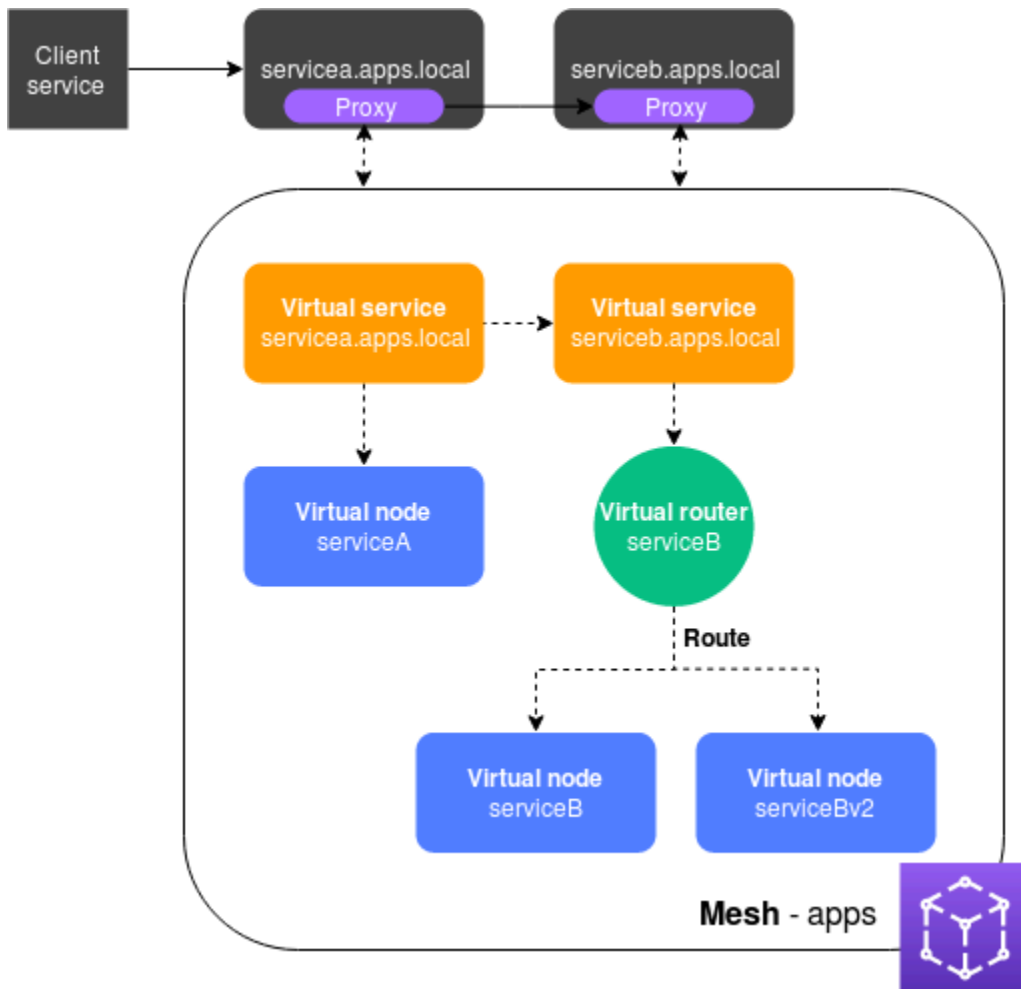
Ajouter App Mesh à un exemple d'application

Prenons l'exemple d'application simple suivant qui n'utilise pas App Mesh. Les deux services peuvent être exécutés sur AWS Fargate Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), Kubernetes sur des instances Amazon Elastic Compute Cloud (Amazon EC2) ou sur des instances Amazon EC2 avec Docker.



Dans cette illustration, les deux `serviceA` et `serviceB` sont détectables via l'espace de noms `apps.local`. Supposons, par exemple, que vous décidiez de déployer une nouvelle version de `serviceb.apps.local` nommée `servicebv2.apps.local`. Ensuite, vous souhaitez diriger un pourcentage du trafic de `servicea.apps.local` vers `serviceb.apps.local` et un pourcentage vers `servicebv2.apps.local`. Lorsque vous êtes sûr que `servicebv2` que cela fonctionne bien, vous souhaitez y envoyer 100 % du trafic.

App Mesh peut vous aider à le faire sans modifier le code de l'application ou les noms de service enregistrés. Si vous utilisez App Mesh avec cet exemple d'application, votre maillage peut ressembler à l'illustration suivante.



Dans cette configuration, les services ne communiquent plus directement entre eux. Au lieu de cela, ils communiquent entre eux via un proxy. Le proxy déployé avec `servicea.apps.local` lit la configuration App Mesh et envoie le trafic vers `serviceb.apps.local` ou `servicebv2.apps.local` en fonction de la configuration.

Composants de l'App Mesh

App Mesh est constitué des composants suivants, illustrés dans l'exemple précédent :

- **Service Mesh** : un maillage de service est une limite logique pour le trafic réseau entre les services qui résident dans celui-ci. Dans cet exemple, le maillage est nommé `apps` et contient toutes les autres ressources du maillage. Pour plus d'informations, veuillez consulter [Mailles de service](#).
- **Services virtuels** : un service virtuel est une abstraction d'un service réel qui est fournie directement ou indirectement par un nœud virtuel, via un routeur virtuel. Dans l'illustration, deux services virtuels représentent les deux services réels. Les noms des services virtuels sont les noms

détectables des services réels. Lorsqu'un service virtuel et un service réel portent le même nom, plusieurs services peuvent communiquer entre eux en utilisant les mêmes noms que ceux qu'ils utilisaient avant la mise en œuvre d'App Mesh. Pour plus d'informations, veuillez consulter [Services virtuels](#).

- **Nœuds virtuels** : un nœud virtuel agit comme pointeur logique vers un service détectable, comme Amazon ECS ou Kubernetes. Pour chaque service virtuel, vous aurez au moins un nœud virtuel. Dans l'illustration, le service `servicea.apps.local` virtuel obtient des informations de configuration pour le nœud virtuel nommé `serviceA`. Le nœud `serviceA` virtuel est configuré avec le `servicea.apps.local` nom pour la découverte de services. Le service `serviceb.apps.local` virtuel est configuré pour acheminer le trafic vers les nœuds `serviceBv2` virtuels `serviceB` et via un routeur virtuel nommé `serviceB`. Pour plus d'informations, veuillez consulter [Nœuds virtuels](#).
- **Routeurs et itinéraires virtuels** : les routeurs virtuels gèrent le trafic pour un ou plusieurs services virtuels au sein de votre mesh. Un itinéraire est associé à un routeur virtuel. La route est utilisée pour faire correspondre les demandes destinées au routeur virtuel et pour distribuer le trafic à ses nœuds virtuels associés. Dans l'illustration précédente, le routeur `serviceB` virtuel possède une route qui dirige un pourcentage du trafic vers le nœud `serviceB` virtuel et un pourcentage du trafic vers le nœud `serviceBv2` virtuel. Vous pouvez définir le pourcentage de trafic acheminé vers un nœud virtuel particulier et le modifier au fil du temps. Vous pouvez acheminer le trafic en fonction de critères tels que les en-têtes HTTP, les chemins d'URL ou les noms de services et de méthodes gRPC. Vous pouvez configurer des politiques de nouvelle tentative pour réessayer de vous connecter en cas d'erreur dans la réponse. Par exemple, dans l'illustration, la politique de nouvelle tentative pour l'itinéraire peut spécifier qu'une connexion à `serviceb.apps.local` est réessayée cinq fois, avec un intervalle de dix secondes entre les tentatives, si elle `serviceb.apps.local` renvoie des types d'erreurs spécifiques. Pour plus d'informations, consultez [Routeurs virtuels](#) et [Acheminements](#).
- **Proxy** : vous configurez vos services pour utiliser le proxy après avoir créé votre maillage et ses ressources. Le proxy lit la configuration App Mesh et dirige le trafic de manière appropriée. Dans l'illustration, toutes les communications entre `servicea.apps.local` et `serviceb.apps.local` passent par le proxy déployé avec chaque service. Les services communiquent entre eux en utilisant les mêmes noms de découverte de services que ceux qu'ils utilisaient avant le lancement d'App Mesh. Comme le proxy lit la configuration App Mesh, vous pouvez contrôler la façon dont les deux services communiquent entre eux. Lorsque vous souhaitez modifier la configuration de l'App Mesh, vous n'avez pas besoin de modifier ou de redéployer les services eux-mêmes ou les proxys. Pour plus d'informations, veuillez consulter [Image de l'envoyé](#).

Comment démarrer

Pour utiliser App MeshAWS Fargate, vous devez disposer d'un service existant exécuté sur Amazon ECS, Amazon EKS, Kubernetes sur Amazon EC2 ou Amazon EC2 avec Docker.

Pour commencer à utiliser App Mesh, consultez l'un des guides suivants :

- [Démarrer avec App Mesh et Amazon ECS](#)
- [Démarrer avec App Mesh et Kubernetes](#)
- [Démarrer avec App Mesh et Amazon EC2](#)

Accès à App Mesh

Vous pouvez utiliser les App Mesh comme suit :

AWS Management Console

La console est une interface basée sur un navigateur que vous pouvez utiliser pour gérer les ressources App Mesh. Vous pouvez ouvrir la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).

AWS CLI

Fournit des commandes pour une large gamme de produits AWS et est prise en charge par Windows, Mac et Linux/UNIX. Consultez [AWS Command Line Interface Guide de l'utilisateur](#) pour démarrer. Pour de plus amples informations sur les commandes pour App Mesh, voir [appMesh](#) dans la [Référence](#) de l'AWS CLIApp Mesh.

AWS Tools for Windows PowerShell

Fournit des commandes pour une large gamme deAWS produits à ceux qui écrivent des scripts dans l' PowerShell environnement. Consultez le [AWS Tools for Windows PowerShell Guide de l'utilisateur](#) pour démarrer. Pour plus d'informations sur les applets de commande pour App Mesh, consultez [App Mesh](#) dans le manuel de [référenceAWS des outils pour les PowerShell cmdlets](#).

AWS CloudFormation

Vous permet de créer un modèle qui décrit toutes lesAWS ressources que vous souhaitez utiliser. À l'aide du modèle,AWS CloudFormation provisionnez et configurez les ressources pour vous. Consultez le [Guide de l'utilisateur AWS CloudFormation](#) pour démarrer. Pour plus d'informations

sur les types de ressources App Mesh, consultez la section [Référence des types de ressources App Mesh](#) dans la [référence des AWS CloudFormation modèles](#).

Kits de développement logiciel (SDK) AWS

Nous fournissons également des kits SDK qui vous permettent d'accéder à App Mesh à partir de différents langages de programmation. Les SDK prennent automatiquement en charge des tâches telles que :

- Signature cryptographique des requêtes de service
- Nouvelles tentatives de requête
- Gestion des réponses d'erreur

Pour plus d'informations sur les kits SDK disponibles, consultez [Outils pour Amazon Web Services](#).

Pour de plus amples informations sur les API Mesh API Mesh, veuillez consulter la [Référence AWS App Mesh d'API Mesh](#).

Débuter avec App Mesh

Vous pouvez utiliser App Mesh avec des applications que vous déployez sur Amazon ECS, Kubernetes (que vous déployez sur vos propres instances Amazon EC2 ou exécutées sur Amazon EKS) et Amazon EC2. Pour commencer à utiliser App Mesh, sélectionnez l'un des services sur lesquels vous avez déployé des applications et que vous souhaitez utiliser avec App Mesh. Vous pouvez toujours activer les applications des autres services pour qu'elles fonctionnent également avec App Mesh après avoir terminé l'un des guides de démarrage.

Rubriques

- [Commencer à utiliser Amazon ECS AWS App Mesh et à utiliser Amazon ECS](#)
- [Premiers pas avec AWS App Mesh Kubernetes](#)
- [Premiers pas avec AWS App Mesh Amazon EC2](#)
- [App Mesh](#)
- [App Mesh](#)

Commencer à utiliser Amazon ECS AWS App Mesh et à utiliser Amazon ECS

Cette rubrique vous aide AWS App Mesh à utiliser un service réel qui s'exécute sur Amazon ECS. Ce didacticiel couvre les fonctionnalités de base de plusieurs types de ressources App Mesh.

Scénario

Pour illustrer l'utilisation d'App Mesh, supposons que vous disposez d'une application présentant les caractéristiques suivantes :

- Se compose de deux services nommés `serviceA` et `serviceB`.
- Les deux services sont enregistrés dans un espace de noms nommé `apps.local`.
- `ServiceA` communique avec `serviceB` via HTTP/2, port 80.
- Vous avez déjà déployé la version 2 de `serviceB` et l'avez enregistrée avec le nom `serviceBv2` dans l'espace de noms `apps.local`.

Les exigences requises sont les suivantes :

- Vous devez envoyer 75 % du trafic de `serviceA` vers `serviceB` et 25 % du trafic `serviceBv2` vers pour vérifier qu'il `serviceBv2` n'y a pas de bogue avant d'envoyer 100 % du trafic en provenance `serviceA` de ce site.
- Vous voulez pouvoir ajuster facilement la pondération du trafic afin que 100 % du trafic soit acheminé vers `serviceBv2` une fois que sa fiabilité a été prouvée. Une fois que tout le trafic est envoyé à `serviceBv2`, vous souhaitez l'arrêter `serviceB`.
- Vous ne souhaitez pas avoir à modifier le code d'application existant ni à vous inscrire à la découverte de services pour que vos services actuels répondent aux exigences précédentes.

Pour répondre à vos besoins, vous décidez de créer un maillage de services App Mesh avec des services virtuels, des nœuds virtuels, un routeur virtuel et un itinéraire. Après avoir implémenté votre maillage, vous mettez à jour vos services pour utiliser le proxy Envoy. Une fois mis à jour, vos services communiquent entre eux via le proxy Envoy plutôt que directement entre eux.

Prérequis

- Compréhension existante des concepts de l'App Mesh. Pour plus d'informations, consultez [Qu'est-ce qu'AWS App Mesh ?](#).
- Compréhension existante des concepts d'Amazon ECS. Pour plus d'informations, consultez la section [Qu'est-ce qu'Amazon ECS](#) dans le guide du développeur Amazon Elastic Container Service.
- App Mesh prend en charge les services Linux enregistrés auprès du DNS AWS Cloud Map, ou des deux. Pour utiliser ce guide de démarrage, nous vous recommandons d'avoir trois services existants enregistrés avec DNS. Les procédures décrites dans cette rubrique supposent que les services existants sont nommés `serviceA` `serviceBv2` et que tous les services sont détectables via un espace de noms nommé `serviceB apps.local`

Vous pouvez créer un maillage de service et ses ressources même si les services n'existent pas, mais vous ne pouvez pas utiliser le maillage tant que vous n'avez pas déployé des services réels. Pour plus d'informations sur la découverte de services sur Amazon ECS, consultez [Service Discovery](#). Pour créer un service Amazon ECS avec service discovery, consultez [Tutoriel : Création d'un service à l'aide de Service Discovery](#). Si aucun service n'est déjà en cours d'exécution, vous pouvez [créer un service Amazon ECS avec service discovery](#).

Étape 1 : Créer un maillage et un service virtuel

Un maillage de service est une limite logique pour le trafic réseau entre les services qui résident dans celui-ci. Pour plus d'informations, consultez [Mailles de service](#). Un service virtuel est une abstraction d'un service réel. Pour plus d'informations, consultez [Services virtuels](#).

Créez les ressources suivantes :

- Un maillage nommé `apps`, puisque tous les services du scénario sont enregistrés dans l'espace de noms `apps.local`.
- Un service virtuel nommé `serviceb.apps.local`, car le service virtuel représente un service détectable avec ce nom et vous ne souhaitez pas modifier votre code pour référencer un autre nom. Un service virtuel nommé `servicea.apps.local` est ajouté dans une étape ultérieure.

Vous pouvez utiliser la AWS Management Console AWS CLI version 1.18.116 ou supérieure ou 2.0.38 ou supérieure pour effectuer les étapes suivantes. Si vous utilisez le AWS CLI, utilisez la `aws --version` commande pour vérifier la AWS CLI version installée. Si la version 1.18.116 ou supérieure ou 2.0.38 ou supérieure n'est pas installée, vous devez [installer](#) ou mettre à jour le. AWS CLI Sélectionnez l'onglet correspondant à l'outil que vous souhaitez utiliser.

AWS Management Console

1. Ouvrez l'assistant de première exécution de la console App Mesh à l'adresse <https://console.aws.amazon.com/appmesh/get-started>.
2. Pour Mesh name (Nom de maillage), entrez **apps**.
3. Pour Virtual service name (Nom du service virtuel), entrez **serviceb.apps.local**.
4. Pour continuer, choisissez Suivant.

AWS CLI

1. Créez un maillage avec la commande [create-mesh](#).

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

2. Créez un service virtuel avec la commande [create-virtual-service](#).

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

Étape 2 : Créer un nœud virtuel

Un nœud virtuel tient lieu de pointeur logique vers un service réel. Pour plus d'informations, consultez [Nœuds virtuels](#).

Créez un nœud virtuel nommé `serviceB`, car l'un des nœuds virtuels représente le service réel nommé `serviceB`. Le service réel représenté par le nœud virtuel est détectable via DNS avec le nom d'hôte `serviceb.apps.local`. Vous pouvez également découvrir les services réels en utilisant AWS Cloud Map. Le nœud virtuel écoutera le trafic en utilisant le protocole HTTP/2 sur le port 80. D'autres protocoles sont également pris en charge, tout comme les vérifications de l'état. Vous allez créer des nœuds virtuels pour `serviceA` et `serviceBv2` dans une étape ultérieure.

AWS Management Console

1. Pour Virtual node name (Nom du nœud virtuel), entrez **serviceB**.
2. Pour Service discovery method (Méthode de découverte de service), choisissez DNS et entrez **serviceb.apps.local** comme DNS hostname (Nom d'hôte DNS).
3. Sous Configuration de l'écouteur, sélectionnez `http2` comme Protocole et entrez **80** comme Port.
4. Pour continuer, choisissez Suivant.

AWS CLI

1. Créez un fichier nommé `create-virtual-node-serviceb.json` avec le contenu suivant :

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

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

2. Créez le nœud virtuel avec la commande [create-virtual-node](#) en utilisant le fichier JSON en entrée.

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

Étape 3 : Créer un routeur virtuel et un routage

Les routeurs virtuels gèrent le trafic d'un ou de plusieurs services virtuels au sein de votre maillage. Pour plus d'informations, consultez [Routeurs virtuels](#) et [Acheminements](#).

Créez les ressources suivantes :

- Routeur virtuel nommé `serviceB`, car le service virtuel `serviceB.apps.local` n'initie pas de communication sortante avec un autre service. N'oubliez pas que le service virtuel que vous avez créé précédemment est une abstraction de votre service `serviceb.apps.local` réel. Le service virtuel envoie du trafic au routeur virtuel. Le routeur virtuel écoute le trafic à l'aide du protocole HTTP/2 sur le port 80. D'autres protocoles sont également pris en charge.
- Un itinéraire nommé `serviceB`. Il achemine 100 % de son trafic vers le nœud `serviceB` virtuel. Le poids sera déterminé ultérieurement une fois que vous aurez ajouté le nœud `serviceBv2` virtuel. Bien que cet aspect ne soit pas couvert dans ce guide, vous pouvez ajouter des critères de filtre supplémentaires pour l'itinéraire et ajouter une stratégie de nouvelle tentative pour que le proxy Envoy fasse plusieurs tentatives pour envoyer du trafic vers un nœud virtuel lorsqu'il rencontre un problème de communication.

AWS Management Console

1. Pour Virtual router name (Nom du routeur virtuel), entrez **serviceB**.
2. Sous Configuration de l'écouteur, sélectionnez http2 comme Protocole et entrez **80** comme Port.
3. Pour Route name (Nom de l'itinéraire), entrez **serviceB**.
4. Pour Route type (Type d'itinéraire), choisissez http2.
5. Pour le nom du nœud virtuel sous Configuration cible, sélectionnez serviceB et entrez **100** le poids.
6. Sous Configuration de correspondance, choisissez une méthode.
7. Pour continuer, choisissez Suivant.

AWS CLI

1. Créez un routeur virtuel.
 - a. Créez un fichier nommé `create-virtual-router.json` avec le contenu suivant :

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

- b. Créez le routeur virtuel avec la commande [create-virtual-router](#) en utilisant le fichier JSON en entrée.

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

2. Créez un itinéraire.

- a. Créez un fichier nommé `create-route.json` avec le contenu suivant :

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

- b. Créez l'itinéraire avec la commande [create-route](#) en utilisant le fichier JSON en entrée.

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

Étape 4 : vérifier et créer

Vérifiez les paramètres par rapport aux instructions précédentes.

AWS Management Console

Choisissez Edit (Modifier) si vous devez apporter des modifications dans une section. Une fois que vous êtes satisfait des paramètres, choisissez Créer un maillage.

L'écran État affiche toutes les ressources de maillage créées. Vous pouvez voir les ressources créées dans la console en sélectionnant Afficher le maillage.

AWS CLI

Passez en revue les paramètres du maillage que vous avez créé à l'aide de la commande [describe-mesh](#).

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

Passez en revue les paramètres du service virtuel que vous avez créé avec la commande [describe-virtual-service](#).

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

Passez en revue les paramètres du nœud virtuel que vous avez créé avec la commande [describe-virtual-node](#).

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

Passez en revue les paramètres du routeur virtuel que vous avez créé avec la commande [describe-virtual-router](#).

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

Passez en revue les paramètres de l'itinéraire que vous avez créé avec la commande [describe-route](#).

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

Étape 5 : Créer des ressources supplémentaires

Pour terminer le scénario, vous devez :

- Créer un nœud virtuel nommé `serviceBv2` et un autre nommé `serviceA`. Les deux nœuds virtuels écoutent les demandes sur le port HTTP/2 80. Pour le nœud `serviceA` virtuel, configurez un backend `deserviceb.apps.local`. Tout le trafic sortant du nœud `serviceA` virtuel est envoyé au service virtuel nommé `serviceb.apps.local`. Bien que cela ne soit pas couvert

dans ce guide, vous pouvez également spécifier un chemin d'accès de fichier vers lequel écrire les journaux d'accès pour un nœud virtuel.

- Créez un service virtuel supplémentaire nommé `servicea.apps.local`, qui envoie tout le trafic directement au nœud `serviceA` virtuel.
- Mettez à jour l'itinéraire `serviceB` que vous avez créé à l'étape précédente pour envoyer 75 % de son trafic vers le nœud virtuel `serviceB` et 25 % de son trafic vers le nœud virtuel `serviceBv2`. Au fil du temps, vous pouvez continuer à modifier les poids jusqu'à ce que `serviceBv2` reçoive 100 % du trafic. Une fois que tout le trafic est envoyé `serviceBv2`, vous pouvez arrêter et interrompre le nœud `serviceB` virtuel et le service réel. Lorsque vous modifiez les pondérations, votre code ne nécessite aucune modification, car les noms de service `serviceb.apps.local` virtuels et réels ne changent pas. Rappelez-vous que le service virtuel `serviceb.apps.local` envoie du trafic au routeur virtuel, qui achemine le trafic vers les nœuds virtuels. Les noms de découverte de service pour les nœuds virtuels peuvent être modifiés à tout moment.

AWS Management Console

1. Dans le panneau de navigation, sélectionnez Meshes (Maillages).
2. Sélectionnez le maillage `apps` que vous avez créé à l'étape précédente.
3. Dans le panneau de navigation de gauche, sélectionnez Virtual nodes (Nœuds virtuels).
4. Choisissez Create virtual node (Créer un nœud virtuel).
5. Pour Virtual node name (Nom de nœud virtuel), entrez **serviceBv2**, pour Service discovery method (Méthode de découverte de service), choisissez DNS, et pour DNS hostname (Nom d'hôte DNS), entrez **servicebv2.apps.local**.
6. Pour la Configuration de l'écouteur, sélectionnez http2 comme Protocole et entrez **80** comme Port.
7. Choisissez Create virtual node (Créer un nœud virtuel).
8. Choisissez Create virtual node (Créer un nœud virtuel). Entrez **serviceA** comme Nom de nœud virtuel. Pour Méthode de découverte de service, choisissez DNS, et comme Nom d'hôte DNS, entrez **servicea.apps.local**.
9. Dans la zone Entrez un nom de service virtuel sous Nouveau backend, entrez **serviceb.apps.local**.
10. Sous Configuration de l'écouteur, choisissez http2 comme Protocole, entrez **80** comme Port, puis choisissez Créer un nœud virtuel.

11. Dans le panneau de navigation de gauche, sélectionnez Virtual routers (Routeurs virtuels), puis choisissez le routeur virtuel `serviceB` dans la liste.
12. Sous Routes (Itinéraires), sélectionnez l'itinéraire `ServiceB` que vous avez créé à l'étape précédente, puis choisissez Modifier.
13. Sous Cibles, Virtual node name (Nom du nœud virtuel), modifiez la valeur de Weight (Pondération) de `serviceB` en **75**.
14. Choisissez Ajouter un objectif, choisissez `serviceBv2` dans la liste déroulante et définissez la valeur du poids sur **25**.
15. Choisissez Enregistrer.
16. Dans le panneau de navigation de gauche, sélectionnez Services virtuels puis choisissez Créer un service virtuel.
17. Entrez **`servicea.apps.local`** pour Nom du service virtuel, sélectionnez Nœud virtuel comme Fournisseur, sélectionnez `serviceA` comme Nœud virtuel, puis choisissez Créer un service virtuel.

AWS CLI

1. Créez le nœud virtuel `serviceBv2`.
 - a. Créez un fichier nommé `create-virtual-node-servicebv2.json` avec le contenu suivant :

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



```
  },  
  "virtualNodeName": "serviceBv2"  
}
```

- b. Créez le nœud virtuel.

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

2. Créez le nœud virtuel serviceA.

- a. Créez un fichier nommé `create-virtual-node-servicea.json` avec le contenu suivant :

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

- b. Créez le nœud virtuel.

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

3. Mettez à jour le service virtuel `serviceb.apps.local` que vous avez créé lors d'une étape précédente pour envoyer son trafic vers le routeur virtuel `serviceB`. Lorsque le service virtuel a été créé à l'origine, il n'envoyait de trafic nulle part, car le routeur virtuel `serviceB` n'avait pas encore été créé.
 - a. Créez un fichier nommé `update-virtual-service.json` avec le contenu suivant :

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

- b. Mettez à jour le service virtuel avec la commande [update-virtual-service](#).

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

4. Mettez à jour l'itinéraire `serviceB` que vous avez créé lors d'une étape précédente.
 - a. Créez un fichier nommé `update-route.json` avec le contenu suivant :

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

```

        },
        {
            "virtualNode" : "serviceBv2",
            "weight" : 25
        }
    ]
},
"match" : {
    "prefix" : "/"
}
}
},
"virtualRouterName" : "serviceB"
}

```

- b. Mettez à jour l'itinéraire avec la commande [update-route](#).

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

5. Créez le service virtuel serviceA.

- a. Créez un fichier nommé `create-virtual-servicea.json` avec le contenu suivant :

```

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

```

- b. Créez le service virtuel.

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

Résumé du maillage

Avant de créer le maillage de service, vous aviez trois services réels nommés `servicea.apps.local`, `serviceb.apps.local` et `servicebv2.apps.local`. En plus des services réels, vous disposez désormais d'un maillage de service qui contient les ressources suivantes représentant les services réels :

- Deux services virtuels. Le proxy envoie tout le trafic du service virtuel `servicea.apps.local` vers le service virtuel `serviceb.apps.local` via un routeur virtuel.
- Trois nœuds virtuels nommés `serviceA`, `serviceB` et `serviceBv2`. Le proxy Envoy utilise les informations de découverte de service configurées pour les nœuds virtuels pour rechercher les adresses IP des services réels.
- Un routeur virtuel avec un itinéraire qui indique au proxy Envoy d'acheminer 75 % du trafic entrant vers le nœud virtuel `serviceB` et 25 % du trafic vers le nœud virtuel `serviceBv2`.

Étape 6 : Mettre à jour les services

Après avoir créé votre maillage, vous devez effectuer les tâches suivantes :

- Autorisez le proxy Envoy que vous déployez avec chaque tâche Amazon ECS à lire la configuration d'un ou de plusieurs nœuds virtuels. Pour plus d'informations sur l'autorisation du proxy, consultez [Autorisation du proxy](#).
- Mettez à jour chacune de vos définitions de tâches Amazon ECS existantes pour utiliser le proxy Envoy.

Informations d'identification

Le conteneur Envoy nécessite des AWS Identity and Access Management informations d'identification pour signer les demandes envoyées au service App Mesh. Pour les tâches Amazon ECS déployées avec le type de lancement Amazon EC2, les informations d'identification peuvent provenir du rôle d'[instance ou d'un rôle IAM de tâche](#). Les tâches Amazon ECS déployées avec Fargate sur les conteneurs Linux n'ont pas accès au serveur de métadonnées Amazon EC2 qui fournit les informations d'identification du profil IAM de l'instance. Pour fournir les informations d'identification, vous devez associer un rôle de tâche IAM à toutes les tâches déployées avec le type de conteneurs Fargate sur Linux.

Si une tâche est déployée avec le type de lancement Amazon EC2 et que l'accès au serveur de métadonnées Amazon EC2 est bloqué, comme décrit dans l'annotation importante dans le rôle [IAM pour les tâches, un rôle IAM de tâche](#) doit également être associé à la tâche. Le rôle que

vous attribuez à l'instance ou à la tâche doit être associé à une politique IAM, comme décrit dans [Autorisation du proxy](#).

Pour mettre à jour votre définition de tâche à l'aide du AWS CLI

Vous utilisez la AWS CLI commande Amazon ECS [register-task-definition](#). L'exemple de définition de tâche ci-dessous montre comment configurer App Mesh pour votre service.

Note

La configuration d'App Mesh pour Amazon ECS via la console n'est pas disponible.

Définition de tâche json

Configuration du proxy

Pour configurer votre service Amazon ECS afin d'utiliser App Mesh, la définition de tâche de votre service doit comporter la section de configuration du proxy suivante. Définissez la configuration de proxy type à APPMESH et le `containerName` à `envoy`. Définissez les valeurs de propriété suivantes en conséquence.

IgnoredUID

Le proxy Envoy n'achemine pas le trafic provenant de processus qui utilisent cet ID utilisateur. Vous pouvez choisir n'importe quel ID utilisateur souhaité pour cette valeur de propriété, mais cet ID doit être identique à celui du conteneur Envoy `user` dans la définition de tâche. Cette mise en correspondance permet à Envoy d'ignorer son propre trafic sans utiliser le proxy. Nos exemples utilisent `1337` à des fins historiques.

ProxyIngressPort

Il s'agit du port entrant pour le conteneur proxy Envoy. Définissez cette valeur à `15000`.

ProxyEgressPort

Il s'agit du port sortant du conteneur proxy Envoy. Définissez cette valeur à `15001`.

AppPorts

Spécifiez les ports entrants que vos conteneurs d'applications écoutent. Dans cet exemple, le conteneur d'application écoute le port `9080`. Le port que vous spécifiez doit correspondre au port configuré sur l'écouteur du nœud virtuel.

EgressIgnoredIPs

Envoy ne proxy pas de trafic vers ces adresses IP. Définissez cette valeur sur `169.254.170.2,169.254.169.254`, ce qui ignore le serveur de métadonnées Amazon EC2 et le point de terminaison des métadonnées des tâches Amazon ECS. Le point de terminaison des métadonnées fournit des rôles IAM pour les informations d'identification des tâches. Vous pouvez ajouter des adresses supplémentaires.

EgressIgnoredPorts

Vous pouvez ajouter une liste de ports séparés par des virgules. Envoy n'attribue pas de proxy pour le trafic vers ces ports. Même si vous ne mentionnez aucun port sur la liste, le port 22 est ignoré.

Note

Le nombre maximum de ports sortants pouvant être ignorés est de 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"
}
]
```

Dépendances Envoy de conteneur d'application

Les conteneurs d'application dans vos définitions de tâches doivent attendre que le proxy Envoy amorce et démarre avant de pouvoir démarrer. Pour vous assurer que cela se produise, vous définissez une `dependsOn` section dans chaque définition de conteneur d'application pour attendre que le conteneur Envoy soit signalé sous le nom de `HEALTHY`. Le bloc de code suivant illustre un exemple de définition de conteneur d'application avec cette dépendance. Toutes les propriétés de l'exemple suivant sont obligatoires. Certaines valeurs de propriété sont également requises, mais d'autres sont *remplaçables*.

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

Définition de conteneur Envoy

Vos définitions de tâches Amazon ECS doivent contenir une image de conteneur App Mesh Envoy.

Toutes les régions [prises en charge](#) autres que `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1`, `etaf-south-1`. Vous pouvez remplacer le *code de région* par n'importe quelle région autre que `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1`, et `af-south-1`.

Standard

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

me-south-1

Standard

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

ap-east-1

Standard

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

ap-southeast-3

Standard

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

eu-south-1

Standard


```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

il-central-1

Standard

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

af-south-1

Standard

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

Public repository

Standard

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.5.0-prod
```

Conforme à la norme FIPS

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

⚠ Important

Seule la version v1.9.0.0-prod ou ultérieure est prise en charge pour une utilisation avec App Mesh.

Vous devez utiliser l'image du conteneur App Mesh Envoy jusqu'à ce que l'équipe du projet Envoy fusionne les modifications compatibles avec App Mesh. Pour plus de détails, consultez le [numéro de la GitHub feuille de route](#).

Toutes les propriétés de l'exemple suivant sont obligatoires. Certaines valeurs de propriété sont également requises, mais d'autres sont *remplaçables*.

📘 Note

- La définition de conteneur Envoy doit être marquée comme `essential`.
- Nous recommandons d'allouer des unités de 512 processeur et au moins des 64 MiB de mémoire au conteneur Envoy. Sur Fargate, le minimum que vous pourrez définir est le 1024 MiB de mémoire.
- Le nom du nœud virtuel pour le service Amazon ECS doit être défini sur la valeur de la `APPMESH_RESOURCE_ARN` propriété. Cette propriété nécessite une version `1.15.0` ou une version ultérieure de l'image Envoy. Pour plus d'informations, consultez [Envoy](#).
- La valeur du paramètre `user` doit correspondre à la valeur `IgnoreDUID` de la configuration du proxy de définition de tâche. Dans cet exemple, nous utilisons `1337`.
- Le test de santé présenté ici attend que le conteneur Envoy démarre correctement avant de signaler à Amazon ECS que le conteneur Envoy est sain et prêt pour le démarrage des conteneurs d'applications.
- Par défaut, App Mesh utilise le nom de la ressource que vous avez spécifiée dans `APPMESH_RESOURCE_ARN` lorsque Envoy fait référence à lui-même dans les métriques et les traces. Vous pouvez remplacer ce comportement en définissant la variable d'environnement `APPMESH_RESOURCE_CLUSTER` avec votre propre nom. Cette propriété nécessite une version `1.15.0` ou une version ultérieure de l'image Envoy. Pour plus d'informations, consultez [Envoy](#).

Le bloc de code suivant présente un exemple de définition de conteneur Envoy.

```
{
  "name": "envoy",
  "image": "840364872350.dkr.ecr.us-west-2.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-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"
}
```

Exemples de définitions de tâches

Les exemples de définitions de tâches Amazon ECS suivants montrent comment fusionner les exemples ci-dessus dans une définition de tâche pour `taskB`. Des exemples sont fournis pour créer des tâches pour les deux types de lancement Amazon ECS avec ou sans utilisation AWS X-Ray. Modifiez les valeurs *remplaçables* le cas échéant, pour créer des définitions de tâche pour les tâches nommées `taskBv2` et `taskA` à partir du scénario. Substituez votre nom de maillage et le nom de nœud virtuel à la valeur `APPMESH_RESOURCE_ARN` et une liste des ports que votre application écoute pour la valeur de configuration de proxy `AppPorts`. Par défaut, App Mesh utilise le nom de la ressource que vous avez spécifiée dans `APPMESH_RESOURCE_ARN` lorsque Envoy fait référence à lui-même dans les métriques et les traces. Vous pouvez remplacer ce comportement en définissant la variable d'environnement `APPMESH_RESOURCE_CLUSTER` avec votre propre nom. Toutes les propriétés des exemples suivants sont obligatoires. Certaines valeurs de propriété sont également requises, mais d'autres sont *remplaçables*.

Si vous exécutez une tâche Amazon ECS comme décrit dans la section Informations d'identification, vous devez ajouter un [rôle IAM de tâche](#) existant aux exemples.

⚠ Important

Fargate doit utiliser une valeur de port supérieure à 1024.

Exemple Définition de tâche JSON pour Amazon ECS - Fargate sur les conteneurs 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:v1.29.5.0-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"
}
```

Exemple Définition de tâches JSON pour Amazon ECS avec AWS X-Ray - Fargate sur des conteneurs Linux

X-Ray vous permet de collecter des données sur les demandes traitées par une application et fournit des outils que vous pouvez utiliser pour visualiser le flux de trafic. L'utilisation du pilote X-Ray pour Envoy permet à Envoy de communiquer des informations de suivi à X-Ray. Vous pouvez activer le suivi X-Ray à l'aide de la [configuration Envoy](#). Sur la base de la configuration, Envoy envoie des données de suivi au daemon X-Ray qui fonctionne comme [un conteneur annexe](#) et le daemon transmet les traces au service X-Ray. Une fois les traces publiées sur X-Ray, vous pouvez utiliser la console X-Ray pour visualiser le graphique des appels de service et demander les détails du suivi. Le JSON suivant représente une définition de tâche pour activer l'intégration de 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:v1.29.5.0-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"
}

```

Exemple Définition de tâche JSON pour Amazon ECS - type de lancement 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:v1.29.5.0-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": "awsipc"
}

```

Exemple Définition de tâche JSON pour Amazon ECS avec le AWS X-Ray type de lancement 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:v1.29.5.0-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"  
}
```

Rubriques avancées

Déploiements Canary à l'aide d'App Mesh

Les déploiements et les versions de Canary vous aident à transférer le trafic entre une ancienne version d'une application et une version récemment déployée. Il surveille également l'état de santé de la nouvelle version déployée. En cas de problème avec la nouvelle version, le déploiement de Canary peut automatiquement faire revenir le trafic à l'ancienne version. Les déploiements Canary vous permettent de transférer le trafic entre les versions de l'application avec un meilleur contrôle.

Pour plus d'informations sur la mise en œuvre de déploiements Canary pour Amazon ECS à l'aide d'App Mesh, consultez [Créer un pipeline avec des déploiements Canary pour Amazon ECS à l'aide d'App Mesh](#)

Note

Pour plus d'exemples et de procédures pas à pas pour App Mesh, consultez le référentiel d'[exemples d'App Mesh](#).

Premiers pas avec AWS App Mesh Kubernetes

Lorsque vous intégrez AWS App Mesh Kubernetes à l'aide du contrôleur App Mesh pour Kubernetes, vous gérez les ressources App Mesh, telles que les maillages, les services virtuels, les nœuds virtuels, les routeurs virtuels et les itinéraires via Kubernetes. Vous ajoutez également automatiquement les images du conteneur App Mesh aux spécifications du pod Kubernetes. Ce didacticiel vous guide tout au long de l'installation du contrôleur App Mesh pour Kubernetes afin de permettre cette intégration.

Le contrôleur est accompagné par le déploiement des définitions de ressources personnalisées Kubernetes suivantes : `meshes`, `virtual services`, `virtual nodes` et `virtual routers`. Le contrôleur surveille la création, la modification et la suppression des ressources personnalisées et apporte des modifications aux ressources App Mesh [the section called “Maillages”](#), [the section called “Services virtuels”](#), [the section called “Nœuds virtuels”](#), [the section called “Passerelle”](#) [the section called “itinéraires de passerelle”](#), [the section called “Routeurs virtuels”](#) (y compris [the section](#)

called “Acheminements”) correspondantes via l'API App Mesh. Pour en savoir plus ou contribuer au contrôleur, consultez le [GitHubprojet](#).

Le contrôleur installe également un webhook qui injecte les conteneurs suivants dans les pods Kubernetes étiquetés avec un nom que vous spécifiez.

- Proxy App Mesh Envoy — Envoy utilise la configuration définie dans le plan de contrôle App Mesh pour déterminer où envoyer le trafic de votre application.
- Gestionnaire de routes proxy App Mesh : met à jour iptables les règles de l'espace de noms réseau d'un pod qui acheminent le trafic entrant et sortant via Envoy. Ce conteneur fonctionne comme un conteneur init Kubernetes à l'intérieur du pod.

Prérequis

- Compréhension existante des concepts de l'App Mesh. Pour plus d'informations, consultez [Qu'est-ce qu'AWS App Mesh ?](#).
- Connaissances des concepts Kubernetes. Pour plus d'informations, consultez [Qu'est-ce que Kubernetes dans la documentation Kubernetes](#).
- Un cluster Kubernetes existant. Si vous n'avez pas de cluster existant, consultez [Getting Started with Amazon EKS](#) dans le guide de l'utilisateur Amazon EKS. Si vous utilisez votre propre cluster Kubernetes sur Amazon EC2, assurez-vous que Docker est authentifié auprès du référentiel Amazon ECR dans lequel se trouve l'image Envoy. Pour plus d'informations, consultez [l'image d'Envoy, l'authentification du registre](#) dans le guide de l'utilisateur d'Amazon Elastic Container Registry [et l'extraction d'une image d'un registre privé](#) dans la documentation de Kubernetes.
- App Mesh prend en charge les services Linux enregistrés auprès du DNS AWS Cloud Map, ou des deux. Pour utiliser ce guide de démarrage, nous vous recommandons d'avoir trois services existants enregistrés avec DNS. Les procédures décrites dans cette rubrique supposent que les services existants sont nommés `serviceA` `serviceBv2` et que tous les services sont détectables via un espace de noms nommé `serviceB apps.local`

Vous pouvez créer un maillage de service et ses ressources même si les services n'existent pas, mais vous ne pouvez pas utiliser le maillage tant que vous n'avez pas déployé des services réels.

- La AWS CLI version 1.18.116 ou ultérieure ou 2.0.38 ou ultérieure est installée. Pour installer ou mettre à niveau le AWS CLI, reportez-vous à la section [Installation du AWS CLI](#).

- Un client `kubectl` configuré pour communiquer avec votre cluster Kubernetes. Si vous utilisez Amazon Elastic Kubernetes Service, vous pouvez suivre les instructions d'installation et de configuration d'un fichier. [kubectl kubeconfig](#)
- Helm version 3.0 ou ultérieure installée. Si Helm n'est pas installé, consultez la section [Utilisation de Helm avec Amazon EKS](#) dans le guide de l'utilisateur Amazon EKS.
- Amazon EKS ne prend actuellement en charge IPv6_ONLY que IPv4_ONLY les préférences IP, car Amazon EKS ne prend actuellement en charge que les pods capables de traiter uniquement du IPv4 trafic ou uniquement IPv6 du trafic.

Les étapes restantes supposent que les services réels sont nommés `serviceA`, `serviceB` et `serviceBv2`, et que tous les services peuvent être détectés via un espace de noms nommé `apps.local`.

Étape 1 : Installer les composants d'intégration

Installez les composants d'intégration une fois sur chaque cluster hébergeant les pods que vous souhaitez utiliser avec App Mesh.

Pour installer les composants d'intégration

1. Les étapes restantes de cette procédure nécessitent un cluster sans qu'une version préalable du contrôleur soit installée. Si vous avez installé une version préliminaire, ou si vous n'êtes pas certain de l'avoir fait, vous pouvez télécharger et exécuter un script qui vérifie si une version préliminaire est installée sur votre 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
```

Si le script retourne `Your cluster is ready for upgrade. Please proceed to the installation instructions`, vous pouvez passer à l'étape suivante. Si un autre message est renvoyé, vous devrez effectuer les étapes de mise à niveau avant de continuer. Pour plus d'informations sur la mise à niveau d'une version préliminaire, consultez la section [Mise à niveau](#) sur GitHub.

2. Ajoutez le référentiel `eks-charts` à Helm.

```
helm repo add eks https://aws.github.io/eks-charts
```

3. Installez les définitions de ressources personnalisées (CRD) App Mesh Kubernetes.

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

4. Créez un espace de noms Kubernetes pour le contrôleur.

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

5. Définissez les variables suivantes à utiliser pour les étapes ultérieures. Remplacez *cluster-name* et *Region-code* par les valeurs de votre cluster existant.

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

6. (Facultatif) Si vous souhaitez exécuter le contrôleur sur Fargate, vous devez créer un profil Fargate. Si ce n'est pas le cas, consultez la section [Installation ou mise à niveau eksctl](#) dans le guide de l'utilisateur Amazon EKS. `eksctl` Si vous préférez créer le profil à l'aide de la console, consultez la section [Création d'un profil Fargate](#) dans le guide de l'utilisateur Amazon EKS.

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

7. Créez un fournisseur d'identité OpenID Connect (OIDC) pour votre cluster. Si ce n'est pas le cas, vous pouvez l'installer en suivant les instructions de la section [Installation ou mise à niveau eksctl](#) du guide de l'utilisateur Amazon EKS. `eksctl` Si vous préférez créer le fournisseur à l'aide de la console, consultez la section [Activation des rôles IAM pour les comptes de service de votre cluster](#) dans le guide de l'utilisateur Amazon EKS.

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

8. Créez un rôle IAM, associez-y les politiques [AWSCloudMapFullAccess](#) AWS gérées [AWSAppMeshFullAccess](#) et liez-le au compte de service `appmesh-controller` Kubernetes. Le rôle permet au contrôleur d'ajouter, de supprimer et de modifier des ressources App Mesh.

Note

La commande crée un rôle AWS IAM avec un nom généré automatiquement. Vous ne pouvez pas spécifier le nom de rôle IAM créé.

```
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
```

Si vous préférez créer le compte de service à l'aide du AWS Management Console ou AWS CLI, consultez la section [Création d'un rôle et d'une politique IAM pour votre compte de service](#) dans le guide de l'utilisateur Amazon EKS. Si vous utilisez le AWS Management Console ou AWS CLI pour créer le compte, vous devez également associer le rôle à un compte de service Kubernetes. Pour plus d'informations, consultez la section [Spécification d'un rôle IAM pour votre compte de service](#) dans le guide de l'utilisateur Amazon EKS.

9. Déployez le contrôleur App Mesh. Pour obtenir la liste de toutes les options de configuration, voir [Configuration](#) activée GitHub.
 1. Pour déployer le contrôleur App Mesh pour un cluster privé, vous devez d'abord activer les points de terminaison Amazon VPC App Mesh et Service Discovery sur le sous-réseau privé lié. Vous devez également définir le `accountId`.

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

Pour activer le suivi X-Ray dans un cluster privé, activez les points de terminaison Amazon VPC X-Ray et Amazon ECR. Le contrôleur l'utilise `public.ecr.aws/xray/aws-xray-daemon:latest` par défaut, alors extrayez cette image en local et [insérez-la dans votre référentiel ECR personnel](#).

Note

Les [points de terminaison Amazon VPC](#) ne sont actuellement pas compatibles avec les référentiels publics Amazon ECR.

L'exemple suivant montre le déploiement du contrôleur avec des configurations pour 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
```

Vérifiez si le daemon X-Ray est correctement injecté lorsque vous liez le déploiement de l'application à votre nœud virtuel ou à votre passerelle.

Pour plus d'informations, consultez la section [Clusters privés](#) dans le guide de l'utilisateur Amazon EKS.

2. Déployez le contrôleur App Mesh pour d'autres clusters. Pour obtenir la liste de toutes les options de configuration, voir [Configuration](#) activée 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

Si votre famille de clusters Amazon EKS l'est IPv6, veuillez définir le nom du cluster lors du déploiement du contrôleur App Mesh en ajoutant l'option suivante à la commande précédente `--set clusterName=$CLUSTER_NAME`.

Important

Si votre cluster se trouve dans les régions `af-south-1`, `me-south-1`, `ap-east-1`, `ap-southeast-3`, `eu-south-1`, `il-central-1`,
Remplacez *account-id* et *Region-code* par l'un des ensembles de valeurs appropriés.

- Pour l'image du sidecar :

- ```
--set image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/amazon/appmesh-controller
```
- `772975370895.dkr.ecr.me-south-1.amazonaws.com /aws-appmesh-envoy:v1.29.5.0-prod`
- `856666278305.dkr.ecr.ap-east-1.amazonaws.com /aws-appmesh-envoy:v1.29.5.0-prod`
- `909464085924.dkr.ecr.ap-southeast-3.amazonaws.com /aws-appmesh-envoy:v1.29.5.0-prod`
- `422531588944.dkr.ecr.eu-south-1.amazonaws.com /aws-appmesh-envoy:v1.29.5.0-prod`
- `564877687649.dkr.ecr.il-central-1.amazonaws.com /aws-appmesh-envoy:v1.29.5.0-prod`
- `924023996002.dkr.ecr.af-south-1.amazonaws.com /aws-appmesh-envoy:v1.29.5.0-prod`
- Les anciens URI des images se trouvent dans le [journal des modifications](#). GitHub Les AWS comptes sur lesquels les images sont présentes ont changé de version `v1.5.0`. Les anciennes versions des images sont hébergées sur des AWS comptes figurant

dans les registres d'images de conteneurs Amazon Elastic Kubernetes [Service](#) Amazon.

- Pour l'image du contrôleur :

- ```
--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:v1.13.0
- 856666278305.dkr.ecr.ap-east-1.amazonaws.com /amazon/appmesh-controller:v1.13.0
- 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com /amazon/appmesh-controller:v1.13.0
- 422531588944.dkr.ecr.eu-south-1.amazonaws.com /amazon/appmesh-controller:v1.13.0
- 564877687649.dkr.ecr.il-central-1.amazonaws.com /amazon/appmesh-controller:v1.13.0
- 924023996002.dkr.ecr.af-south-1.amazonaws.com /amazon/appmesh-controller:v1.13.0

- Pour l'image d'initialisation du sidecar :

- ```
--set sidecar.image.repository=account-id.dkr.ecr.Region-code.amazonaws.com/aws-appmesh-envoy
```
- 772975370895.dkr.ecr.me-south-1.amazonaws.com /aws-appmesh-proxy-route-manager:v7-prod
- 856666278305.dkr.ecr.ap-east-1.amazonaws.com /aws-appmesh-proxy-route-manager:v7-prod
- 909464085924.dkr.ecr.ap-southeast-3.amazonaws.com /aws-appmesh-proxy-route-manager:v7-prod
- 422531588944.dkr.ecr.eu-south-1.amazonaws.com /aws-appmesh-proxy-route-manager:v7-prod
- 564877687649.dkr.ecr.il-central-1.amazonaws.com /aws-appmesh-proxy-route-manager:v7-prod

- 924023996002.dkr.ecr.af-south-1.amazonaws.com /aws-appmesh-proxy-route-manager:v7-prod

### Important

Seule la version v1.9.0.0-prod ou ultérieure est prise en charge pour une utilisation avec App Mesh.

10. Vérifiez que la version du contrôleur est v1.4.0 ou une version ultérieure. Vous pouvez consulter le [journal des modifications](#) GitHub.

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

### Note

Si vous affichez le journal du conteneur en cours d'exécution, une ligne contenant le texte suivant peut s'afficher. Vous pouvez l'ignorer en toute sécurité.

```
Neither -kubeconfig nor -master was specified. Using the inClusterConfig.
This might not work.
```

## Étape 2 : Déployer les ressources App Mesh

Lorsque vous déployez une application dans Kubernetes, vous créez également les ressources personnalisées Kubernetes afin que le contrôleur puisse créer les ressources App Mesh correspondantes. La procédure suivante vous aide à déployer des ressources App Mesh avec certaines de leurs fonctionnalités. Vous pouvez trouver des exemples de manifestes pour le déploiement d'autres fonctionnalités des ressources App Mesh dans les v1beta2 sous-dossiers de nombreux dossiers de fonctionnalités répertoriés dans les procédures pas à pas d'[App Mesh sur GitHub](#).

**⚠ Important**

Une fois que le contrôleur a créé une ressource App Mesh, nous vous recommandons de ne modifier ou de supprimer la ressource App Mesh qu'à l'aide du contrôleur. Si vous modifiez ou supprimez la ressource à l'aide d'App Mesh, le contrôleur ne modifiera ni ne recréera la ressource App Mesh modifiée ou supprimée pendant dix heures, par défaut. Vous pouvez configurer cette durée afin qu'elle soit inférieure. Pour plus d'informations, consultez [Configuration](#) sur GitHub.

## Pour déployer des ressources App Mesh

1. Créez un espace de noms Kubernetes dans lequel déployer les ressources App Mesh.
  - a. Enregistrez le contenu suivant dans un fichier nommé `namespace.yaml` sur votre ordinateur.

```
apiVersion: v1
kind: Namespace
metadata:
 name: my-apps
labels:
 mesh: my-mesh
 appmesh.k8s.aws/sidecarInjectorWebhook: enabled
```

- b. Créez l'espace de noms.

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

2. Créez un maillage de service App Mesh.
  - a. Enregistrez le contenu suivant dans un fichier nommé `mesh.yaml` sur votre ordinateur. Le fichier est utilisé pour créer une ressource de maillage nommée *my-mesh*. Un maillage de service est une limite logique pour le trafic réseau entre les services qui résident dans celui-ci.

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

```
spec:
 namespaceSelector:
 matchLabels:
 mesh: my-mesh
```

- b. Créez le maillage.

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

- c. Affichez les détails de la ressource de maillage Kubernetes créée.

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

## Sortie

```
Name: my-mesh
Namespace:
Labels: <none>
Annotations: kubect1.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-j11k11111m711
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. Consultez les détails du maillage de service App Mesh créé par le contrôleur.

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

### Sortie

```
{
 "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-j11k11111m711",
 "version": 1
 },
 "spec": {},
 "status": {
 "status": "ACTIVE"
 }
 }
}
```

3. Créez un nœud virtuel App Mesh. Un nœud virtuel agit comme un pointeur logique vers un déploiement Kubernetes.
- a. Enregistrez le contenu suivant dans un fichier nommé `virtual-node.yaml` sur votre ordinateur. Le fichier est utilisé pour créer un nœud virtuel App Mesh nommé *my-service-a* dans l'espace de *my-apps* noms. Le nœud virtuel représente un service Kubernetes créé dans une étape ultérieure. La valeur de `hostname` est le nom d'hôte DNS complet du service réel que ce nœud virtuel représente.

```
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
```

Les nœuds virtuels possèdent des fonctionnalités, telles que le end-to-end chiffrement et les contrôles de santé, qui ne sont pas abordées dans ce didacticiel. Pour plus d'informations, consultez [the section called "Nœuds virtuels"](#). Pour afficher tous les paramètres disponibles pour un nœud virtuel que vous pouvez définir dans la spécification précédente, exécutez la commande suivante.

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

- b. Déployez le nœud virtuel.

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

- c. Affichez les détails de la ressource de nœud virtuel Kubernetes créée.

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

## Sortie

```
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-j11k11111m711
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-j11k11111m711
 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. Affichez les détails du nœud virtuel créé par le contrôleur dans App Mesh.

#### Note

Même si le nom du nœud virtuel créé dans Kubernetes est *my-service-a*, le nom du nœud virtuel créé dans App Mesh est *my-service-a\_my-apps*. Le contrôleur ajoute le nom de l'espace de noms Kubernetes au nom du nœud virtuel App Mesh lorsqu'il crée la ressource App Mesh. Le nom de l'espace de noms est ajouté car dans Kubernetes, vous pouvez créer des nœuds virtuels portant le même nom dans

différents espaces de noms, mais dans App Mesh, le nom d'un nœud virtuel doit être unique au sein d'un maillage.

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

## Sortie

```
{
 "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-j11k11111m711",
 "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. Créez un routeur virtuel App Mesh. Les routeurs virtuels gèrent le trafic d'un ou de plusieurs services virtuels au sein de votre mesh.
  - a. Enregistrez le contenu suivant dans un fichier nommé `virtual-router.yaml` sur votre ordinateur. Le fichier est utilisé pour créer un routeur virtuel afin d'acheminer le trafic vers le nœud virtuel nommé `my-service-a` qui a été créé à l'étape précédente. Le contrôleur crée le routeur virtuel App Mesh et achemine les ressources. Vous pouvez spécifier beaucoup d'autres fonctionnalités pour vos routages et utiliser des protocoles autres que `http`. Pour plus d'informations, consultez [the section called "Routeurs virtuels"](#) et [the section called "Acheminements"](#). Notez que le nom du nœud virtuel référencé est le nom du nœud virtuel Kubernetes, et non le nom du nœud virtuel App Mesh créé dans App Mesh par le contrôleur.

```
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
```

(Facultatif) Pour voir tous les paramètres disponibles pour un routeur virtuel que vous pouvez définir dans la spécification précédente, exécutez la commande suivante.

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

Pour voir tous les paramètres disponibles pour un itinéraire que vous pouvez définir dans la spécification précédente, exécutez la commande suivante.

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

- b. Déployez le routeur virtuel.

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

- c. Affichez la ressource de routeur virtuel Kubernetes qui a été créée.

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

### Sortie abrégée

```
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-j11k11111m711
 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. Affichez la ressource de routeur virtuel créée par le contrôleur dans App Mesh. Vous spécifiez `my-service-a-virtual-router_my-apps` pour `name`, car lorsque le contrôleur a créé le routeur virtuel dans App Mesh, il a ajouté le nom de l'espace de noms Kubernetes au nom du routeur virtuel.

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

## Sortie

```

{
 "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-j11k11111m711",
 "version": 1
 },
 "spec": {
 "listeners": [
 {

```

```

 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
],
 "status": {
 "status": "ACTIVE"
 },
 "virtualRouterName": "my-service-a-virtual-router_my-apps"
}
}

```

- e. Affichez la ressource d'itinéraire créée par le contrôleur dans App Mesh. Une ressource de route n'a pas été créée dans Kubernetes car la route fait partie de la configuration du routeur virtuel dans Kubernetes. Les informations de route ont été affichées dans le détail de la ressource Kubernetes dans la sous-étape c. Le contrôleur n'a pas ajouté le nom de l'espace de noms Kubernetes au nom de l'itinéraire App Mesh lorsqu'il a créé l'itinéraire dans App Mesh, car les noms des itinéraires sont uniques à un routeur virtuel.

```

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

```

## Sortie

```

{
 "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-j11k11111m711",
 "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. Créez un service virtuel App Mesh. Un service virtuel est une abstraction d'un service réel qui est fournie directement ou indirectement par un nœud virtuel, via un routeur virtuel. Les services dépendants appellent votre service virtuel par son nom. Bien que le nom n'ait pas d'importance pour App Mesh, nous recommandons de nommer le service virtuel comme le nom de domaine complet du service réel que le service virtuel représente. En nommant vos services virtuels de cette façon, vous n'avez pas besoin de modifier votre code d'application pour référencer un autre nom. Ces demandes sont acheminées vers le nœud virtuel ou le routeur virtuel qui est spécifié en tant que fournisseur pour le service virtuel.
  - a. Enregistrez le contenu suivant dans un fichier nommé `virtual-service.yaml` sur votre ordinateur. Le fichier est utilisé pour créer un service virtuel qui utilise un fournisseur de routeur virtuel pour acheminer le trafic vers le nœud virtuel nommé `my-service-a` qui a été créé à l'étape précédente. La valeur de `awsName` dans le `spec` correspond au nom de domaine complet (FQDN) du service Kubernetes réel que ce service virtuel résume. Le service Kubernetes est créé dans [the section called "Étape 3 : Créer ou mettre à jour des services"](#). Pour plus d'informations, consultez [the section called "Services virtuels"](#).

```
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

```

Pour afficher tous les paramètres disponibles pour un nœud virtuel que vous pouvez définir dans la spécification précédente, exécutez la commande suivante.

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

- b. Créez le service virtuel.

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

- c. Affichez les détails de la ressource de service virtuel Kubernetes créée.

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

## Sortie

```

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-j11k11111m711
Spec:
 Aws Name: my-service-a.my-apps.svc.cluster.local
 Mesh Ref:
 Name: my-mesh
 UID: 111a11b1-c11d-1e1f-gh1i-j11k11111m711
 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. Affichez les détails de la ressource de service virtuel créée par le contrôleur dans App Mesh. Le contrôleur Kubernetes n'a pas ajouté le nom de l'espace de noms Kubernetes au nom du service virtuel App Mesh lorsqu'il a créé le service virtuel dans App Mesh, car le nom du service virtuel est un FQDN unique.

```

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

```

## Sortie

```

{
 "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-j11k11111m711",
 "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"
 }
}
```

Bien que cela ne soit pas abordé dans ce didacticiel, le contrôleur peut également déployer App Mesh [the section called “Passerelle”](#) et [the section called “itinéraires de passerelle”](#). Pour une présentation détaillée du déploiement de ces ressources avec le contrôleur, consultez la [section Configuration de la passerelle entrante](#) ou un [exemple de manifeste](#) incluant les ressources sur GitHub.

## Étape 3 : Créer ou mettre à jour des services

Les conteneurs annexes App Mesh doivent être ajoutés à tous les pods que vous souhaitez utiliser avec App Mesh. L'injecteur ajoute automatiquement les conteneurs sidecar à n'importe quel pod déployé dans un espace de noms avec l'étiquette que vous spécifiez.

1. Activez l'autorisation proxy. Nous vous recommandons d'activer chaque déploiement Kubernetes pour diffuser uniquement la configuration de son propre nœud virtuel App Mesh.
  - a. Enregistrez le contenu suivant dans un fichier nommé `proxy-auth.json` sur votre ordinateur. Assurez-vous de remplacer les *valeurs d'autre couleur* par les vôtres.

```
{
 "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. Créez la politique.

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

- c. Créez un rôle IAM, associez-y la politique que vous avez créée à l'étape précédente, créez un compte de service Kubernetes et liez la politique au compte de service Kubernetes. Le rôle permet au contrôleur d'ajouter, de supprimer et de modifier des ressources 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
```

Si vous préférez créer le compte de service à l'aide du AWS Management Console ou AWS CLI, consultez la section [Création d'un rôle et d'une politique IAM pour votre compte de service](#) dans le guide de l'utilisateur Amazon EKS. Si vous utilisez le AWS Management Console ou AWS CLI pour créer le compte, vous devez également associer le rôle à un compte de service Kubernetes. Pour plus d'informations, consultez la section [Spécification d'un rôle IAM pour votre compte de service](#) dans le guide de l'utilisateur Amazon EKS.

2. (Facultatif) Si vous souhaitez déployer votre déploiement sur des pods Fargate, vous devez créer un profil Fargate. Si ce n'est pas le cas, vous pouvez l'installer en suivant les instructions de la section [Installation ou mise eksctl à niveau](#) du guide de l'utilisateur Amazon EKS. `eksctl` Si vous préférez créer le profil à l'aide de la console, consultez la section [Création d'un profil Fargate](#) dans le guide de l'utilisateur Amazon EKS.

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

3. Créez un service et un déploiement Kubernetes. Si vous avez un déploiement existant que vous souhaitez utiliser avec App Mesh, vous devez déployer un nœud virtuel, comme vous l'avez fait à la sous-étape 3 de [the section called “Étape 2 : Déployer les ressources App Mesh”](#). Mettez à jour votre déploiement pour vous assurer que son étiquette correspond à celle que vous avez définie sur le nœud virtuel, afin que les conteneurs annexes soient automatiquement ajoutés aux pods et que les pods soient redéployés.
  - a. Enregistrez le contenu suivant dans un fichier nommé `example-service.yaml` sur votre ordinateur. Si vous modifiez le nom de l'espace de noms et que vous utilisez des pods Fargate, assurez-vous que le nom de l'espace de noms correspond à celui défini dans votre profil 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

La valeur de la spécification `app matchLabels selector` dans la spécification doit correspondre à la valeur que vous avez spécifiée lors de la création du nœud virtuel dans la sous-étape 3 de [the section called “Étape 2 : Déployer les ressources App Mesh”](#), sinon les conteneurs sidecar ne seront pas injectés dans le conteneur. Dans l'exemple précédent, la valeur de l'étiquette est `my-app-1`. Si vous déployez une passerelle virtuelle plutôt qu'un nœud virtuel, le Deployment manifeste ne doit inclure que le conteneur Envoy. Pour plus d'informations sur l'image à utiliser, consultez [Envoy](#). Pour un exemple de manifeste, consultez [l'exemple de déploiement](#) sur GitHub.

- b. Déployez le service.

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

- c. Consultez le service et le déploiement.

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

### Sortie

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

- d. Consultez les détails de l'un des pods qui a été déployé.

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

## Sortie abrégée

```
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

```

Dans la sortie précédente, vous pouvez voir que les conteneurs proxyinit et envoy ont été ajoutés au pod. Si vous avez déployé l'exemple de service sur Fargate, envoy le conteneur a été ajouté au pod par le contrôleur, mais pas proxyinit le conteneur.

- (Facultatif) Installez des modules complémentaires tels que Prometheus, Grafana, Jaeger et AWS X-Ray Datadog. Pour plus d'informations, consultez les [modules complémentaires App Mesh](#) GitHub et la section [Observabilité](#) du guide de l'utilisateur d'App Mesh.

#### Note

Pour plus d'exemples et de procédures pas à pas pour App Mesh, consultez le référentiel [d'exemples d'App Mesh](#).

## Étape 4 : Nettoyer

Supprimez toutes les ressources d'exemple créées dans ce didacticiel. Le contrôleur supprime également les ressources créées dans le maillage du service `my-mesh` App Mesh.

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

Si vous avez créé un profil Fargate pour le service d'exemple, supprimez-le.

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

Supprimez le maillage.

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

(Facultatif) Vous pouvez supprimer les composants d'intégration Kubernetes.

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

(Facultatif) Si vous avez déployé les composants d'intégration Kubernetes sur Fargate, supprimez le profil Fargate.

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

# Premiers pas avec AWS App Mesh Amazon EC2

Cette rubrique vous aide à utiliser AWS App Mesh un service réel qui s'exécute sur Amazon EC2. Ce didacticiel couvre les fonctionnalités de base de plusieurs types de ressources App Mesh.

## Scénario

Pour illustrer l'utilisation d'App Mesh, supposons que vous disposez d'une application présentant les caractéristiques suivantes :

- Se compose de deux services nommés `serviceA` et `serviceB`.
- Les deux services sont enregistrés dans un espace de noms nommé `apps.local`.
- `ServiceA` communique avec `serviceB` via HTTP/2, port 80.
- Vous avez déjà déployé la version 2 de `serviceB` et l'avez enregistrée avec le nom `serviceBv2` dans l'espace de noms `apps.local`.

Les exigences requises sont les suivantes :

- Vous devez envoyer 75 % du trafic de `serviceA` vers `serviceB` et 25 % du trafic `serviceBv2` vers pour vérifier qu'il `serviceBv2` n'y a pas de bogue avant d'envoyer 100 % du trafic en provenance `serviceA` de ce site.
- Vous voulez pouvoir ajuster facilement la pondération du trafic afin que 100 % du trafic soit acheminé vers `serviceBv2` une fois que sa fiabilité a été prouvée. Une fois que tout le trafic est envoyé à `serviceBv2`, vous souhaitez l'arrêter `serviceB`.
- Vous ne souhaitez pas avoir à modifier le code d'application existant ni à vous inscrire à la découverte de services pour que vos services actuels répondent aux exigences précédentes.

Pour répondre à vos besoins, vous décidez de créer un maillage de services App Mesh avec des services virtuels, des nœuds virtuels, un routeur virtuel et un itinéraire. Après avoir implémenté votre maillage, vous mettez à jour vos services pour utiliser le proxy Envoy. Une fois mis à jour, vos services communiquent entre eux via le proxy Envoy plutôt que directement entre eux.

## Prérequis

App Mesh prend en charge les services Linux enregistrés auprès du DNS AWS Cloud Map, ou des deux. Pour utiliser ce guide de démarrage, nous vous recommandons d'avoir trois services

existants enregistrés avec DNS. Vous pouvez créer un maillage de service et ses ressources même si les services n'existent pas, mais vous ne pouvez pas utiliser le maillage tant que vous n'avez pas déployé des services réels.

Si aucun service n'est déjà en cours d'exécution, vous pouvez lancer des instances Amazon EC2 et y déployer des applications. Pour plus d'informations, consultez [Tutoriel : Démarrage avec les instances Linux Amazon EC2](#) dans le guide de l'utilisateur Amazon EC2. Les étapes restantes supposent que les services réels sont nommés `serviceA`, `serviceB` et `serviceBv2`, et que tous les services peuvent être détectés via un espace de noms nommé `apps.local`.

## Étape 1 : Créer un maillage et un service virtuel

Un maillage de service est une limite logique pour le trafic réseau entre les services qui résident dans celui-ci. Pour plus d'informations, consultez [Mailles de service](#). Un service virtuel est une abstraction d'un service réel. Pour plus d'informations, consultez [Services virtuels](#).

Créez les ressources suivantes :

- Un maillage nommé `apps`, puisque tous les services du scénario sont enregistrés dans l'espace de noms `apps.local`.
- Un service virtuel nommé `serviceb.apps.local`, car le service virtuel représente un service détectable avec ce nom et vous ne souhaitez pas modifier votre code pour référencer un autre nom. Un service virtuel nommé `servicea.apps.local` est ajouté dans une étape ultérieure.

Vous pouvez utiliser la AWS Management Console AWS CLI version 1.18.116 ou supérieure ou 2.0.38 ou supérieure pour effectuer les étapes suivantes. Si vous utilisez le AWS CLI, utilisez la `aws --version` commande pour vérifier la AWS CLI version installée. Si la version 1.18.116 ou supérieure ou 2.0.38 ou supérieure n'est pas installée, vous devez [installer](#) ou mettre à jour le. AWS CLI Sélectionnez l'onglet correspondant à l'outil que vous souhaitez utiliser.

### AWS Management Console

1. Ouvrez l'assistant de première exécution de la console App Mesh à l'adresse <https://console.aws.amazon.com/appmesh/get-started>.
2. Pour Mesh name (Nom de maillage), entrez **apps**.
3. Pour Virtual service name (Nom du service virtuel), entrez **serviceb.apps.local**.
4. Pour continuer, choisissez Suivant.

## AWS CLI

1. Créez un maillage avec la commande [create-mesh](#).

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

2. Créez un service virtuel avec la commande [create-virtual-service](#).

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

## Étape 2 : Créer un nœud virtuel

Un nœud virtuel tient lieu de pointeur logique vers un service réel. Pour plus d'informations, consultez [Nœuds virtuels](#).

Créez un nœud virtuel nommé `serviceB`, car l'un des nœuds virtuels représente le service réel nommé `serviceB`. Le service réel représenté par le nœud virtuel est détectable via DNS avec le nom d'hôte `serviceb.apps.local`. Vous pouvez également découvrir les services réels en utilisant AWS Cloud Map. Le nœud virtuel écoute le trafic à l'aide du protocole HTTP/2 sur le port 80. D'autres protocoles sont également pris en charge, tout comme les vérifications de l'état. Vous créez des nœuds virtuels pour `serviceA` et `serviceBv2` lors d'une étape ultérieure.

### AWS Management Console

1. Pour Virtual node name (Nom du nœud virtuel), entrez **serviceB**.
2. Pour Service discovery method (Méthode de découverte de service), choisissez DNS et entrez **serviceb.apps.local** comme DNS hostname (Nom d'hôte DNS).
3. Sous Configuration de l'écouteur, sélectionnez http2 comme Protocole et entrez **80** comme Port.
4. Pour continuer, choisissez Suivant.

## AWS CLI

1. Créez un fichier nommé `create-virtual-node-serviceb.json` avec le contenu suivant :

```
{
```

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

2. Créez le nœud virtuel avec la commande [create-virtual-node](#) en utilisant le fichier JSON en entrée.

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

## Étape 3 : Créer un routeur virtuel et un routage

Les routeurs virtuels gèrent le trafic d'un ou de plusieurs services virtuels au sein de votre maillage. Pour plus d'informations, consultez [Routeurs virtuels](#) et [Acheminements](#).

Créez les ressources suivantes :

- Routeur virtuel nommé `serviceB`, car le service virtuel `serviceB.apps.local` n'initie pas de communication sortante avec un autre service. N'oubliez pas que le service virtuel que vous avez créé précédemment est une abstraction de votre service `serviceb.apps.local` réel. Le service virtuel envoie du trafic au routeur virtuel. Le routeur virtuel écoute le trafic à l'aide du protocole HTTP/2 sur le port 80. D'autres protocoles sont également pris en charge.
- Un itinéraire nommé `serviceB`. Il achemine 100 % de son trafic vers le nœud `serviceB` virtuel. Le poids sera déterminé ultérieurement une fois que vous aurez ajouté le nœud `serviceBv2` virtuel. Bien que cet aspect ne soit pas couvert dans ce guide, vous pouvez ajouter des critères

de filtre supplémentaires pour l'itinéraire et ajouter une stratégie de nouvelle tentative pour que le proxy Envoy fasse plusieurs tentatives pour envoyer du trafic vers un nœud virtuel lorsqu'il rencontre un problème de communication.

## AWS Management Console

1. Pour Virtual router name (Nom du routeur virtuel), entrez **serviceB**.
2. Sous Configuration de l'écouteur, sélectionnez http2 comme Protocole et entrez **80** comme Port.
3. Pour Route name (Nom de l'itinéraire), entrez **serviceB**.
4. Pour Route type (Type d'itinéraire), choisissez http2.
5. Pour le nom du nœud virtuel sous Configuration cible, sélectionnez serviceB et entrez **100** le poids.
6. Sous Configuration de correspondance, choisissez une méthode.
7. Pour continuer, choisissez Suivant.

## AWS CLI

1. Créez un routeur virtuel.
  - a. Créez un fichier nommé `create-virtual-router.json` avec le contenu suivant :

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

- b. Créez le routeur virtuel avec la commande [create-virtual-router](#) en utilisant le fichier JSON en entrée.

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

## 2. Créez un itinéraire.

- a. Créez un fichier nommé `create-route.json` avec le contenu suivant :

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

- b. Créez l'itinéraire avec la commande [create-route](#) en utilisant le fichier JSON en entrée.

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

## Étape 4 : vérifier et créer

Vérifiez les paramètres par rapport aux instructions précédentes.



## AWS Management Console

Choisissez Edit (Modifier) si vous devez apporter des modifications dans une section. Une fois que vous êtes satisfait des paramètres, choisissez Créer un maillage.

L'écran État affiche toutes les ressources de maillage créées. Vous pouvez voir les ressources créées dans la console en sélectionnant Afficher le maillage.

## AWS CLI

Passez en revue les paramètres du maillage que vous avez créé à l'aide de la commande [describe-mesh](#).

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

Passez en revue les paramètres du service virtuel que vous avez créé avec la commande [describe-virtual-service](#).

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

Passez en revue les paramètres du nœud virtuel que vous avez créé avec la commande [describe-virtual-node](#).

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

Passez en revue les paramètres du routeur virtuel que vous avez créé avec la commande [describe-virtual-router](#).

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

Passez en revue les paramètres de l'itinéraire que vous avez créé avec la commande [describe-route](#).

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

## Étape 5 : Créer des ressources supplémentaires

Pour terminer le scénario, vous devez :

- Créer un nœud virtuel nommé `serviceBv2` et un autre nommé `serviceA`. Les deux nœuds virtuels écoutent les demandes sur le port HTTP/2 80. Pour le nœud `serviceA` virtuel, configurez un backend `deserviceb.apps.local`. Tout le trafic sortant du nœud `serviceA` virtuel est envoyé au service virtuel nommé `serviceb.apps.local`. Bien que cela ne soit pas couvert dans ce guide, vous pouvez également spécifier un chemin d'accès de fichier vers lequel écrire les journaux d'accès pour un nœud virtuel.
- Créez un service virtuel supplémentaire nommé `servicea.apps.local`, qui envoie tout le trafic directement au nœud `serviceA` virtuel.
- Mettez à jour l'itinéraire `serviceB` que vous avez créé à l'étape précédente pour envoyer 75 % de son trafic vers le nœud virtuel `serviceB` et 25 % de son trafic vers le nœud virtuel `serviceBv2`. Au fil du temps, vous pouvez continuer à modifier les poids jusqu'à ce que `serviceBv2` reçoive 100 % du trafic. Une fois que tout le trafic est envoyé à `serviceBv2`, vous pouvez arrêter et interrompre le nœud `serviceB` virtuel et le service réel. Lorsque vous modifiez les pondérations, votre code ne nécessite aucune modification, car les noms de service `serviceb.apps.local` virtuels et réels ne changent pas. Rappelez-vous que le service virtuel `serviceb.apps.local` envoie du trafic au routeur virtuel, qui achemine le trafic vers les nœuds virtuels. Les noms de découverte de service pour les nœuds virtuels peuvent être modifiés à tout moment.

## AWS Management Console

1. Dans le panneau de navigation, sélectionnez Meshes (Maillages).
2. Sélectionnez le maillage `apps` que vous avez créé à l'étape précédente.
3. Dans le panneau de navigation de gauche, sélectionnez Virtual nodes (Nœuds virtuels).
4. Choisissez Create virtual node (Créer un nœud virtuel).
5. Pour Virtual node name (Nom de nœud virtuel), entrez **serviceBv2**, pour Service discovery method (Méthode de découverte de service), choisissez DNS, et pour DNS hostname (Nom d'hôte DNS), entrez **servicebv2.apps.local**.
6. Pour la Configuration de l'écouteur, sélectionnez http2 comme Protocole et entrez **80** comme Port.
7. Choisissez Create virtual node (Créer un nœud virtuel).
8. Choisissez Create virtual node (Créer un nœud virtuel). Entrez **serviceA** comme Nom de nœud virtuel. Pour Méthode de découverte de service, choisissez DNS, et comme Nom d'hôte DNS, entrez **servicea.apps.local**.

9. Dans la zone Entrez un nom de service virtuel sous Nouveau backend, entrez **serviceb.apps.local**.
10. Sous Configuration de l'écouteur, choisissez http2 comme Protocole, entrez **80** comme Port, puis choisissez Créer un nœud virtuel.
11. Dans le panneau de navigation de gauche, sélectionnez Virtual routers (Routeurs virtuels), puis choisissez le routeur virtuel serviceB dans la liste.
12. Sous Routes (Itinéraires), sélectionnez l'itinéraire ServiceB que vous avez créé à l'étape précédente, puis choisissez Modifier.
13. Sous Cibles, Virtual node name (Nom du nœud virtuel), modifiez la valeur de Weight (Pondération) de serviceB en **75**.
14. Choisissez Ajouter un objectif, choisissez serviceBv2 dans la liste déroulante et définissez la valeur du poids sur **25**.
15. Choisissez Enregistrer.
16. Dans le panneau de navigation de gauche, sélectionnez Services virtuels puis choisissez Créer un service virtuel.
17. Entrez **servicea.apps.local** pour Nom du service virtuel, sélectionnez Nœud virtuel comme Fournisseur, sélectionnez serviceA comme Nœud virtuel, puis choisissez Créer un service virtuel.

## AWS CLI

1. Créez le nœud virtuel serviceBv2.
  - a. Créez un fichier nommé `create-virtual-node-servicebv2.json` avec le contenu suivant :

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

```
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceBv2.apps.local"
 }
 },
 "virtualNodeName": "serviceBv2"
 }
}
```

- b. Créez le nœud virtuel.

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

2. Créez le nœud virtuel serviceA.

- a. Créez un fichier nommé `create-virtual-node-servicea.json` avec le contenu suivant :

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

```
"virtualNodeName" : "serviceA"
}
```

- b. Créez le nœud virtuel.

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

3. Mettez à jour le service virtuel `serviceb.apps.local` que vous avez créé lors d'une étape précédente pour envoyer son trafic vers le routeur virtuel `serviceB`. Lorsque le service virtuel a été créé à l'origine, il n'envoyait de trafic nulle part, car le routeur virtuel `serviceB` n'avait pas encore été créé.

- a. Créez un fichier nommé `update-virtual-service.json` avec le contenu suivant :

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

- b. Mettez à jour le service virtuel avec la commande [update-virtual-service](#).

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

4. Mettez à jour l'itinéraire `serviceB` que vous avez créé lors d'une étape précédente.

- a. Créez un fichier nommé `update-route.json` avec le contenu suivant :

```
{
 "meshName" : "apps",
 "routeName" : "serviceB",
 "spec" : {
 "http2Route" : {
 "action" : {
```

```

 "weightedTargets" : [
 {
 "virtualNode" : "serviceB",
 "weight" : 75
 },
 {
 "virtualNode" : "serviceBv2",
 "weight" : 25
 }
],
 "match" : {
 "prefix" : "/"
 }
 },
 "virtualRouterName" : "serviceB"
}

```

- b. Mettez à jour l'itinéraire avec la commande [update-route](#).

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

## 5. Créez le service virtuel serviceA.

- a. Créez un fichier nommé `create-virtual-servicea.json` avec le contenu suivant :

```

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

```

- b. Créez le service virtuel.

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

## Résumé du maillage

Avant de créer le maillage de service, vous aviez trois services réels nommés `servicea.apps.local`, `serviceb.apps.local` et `servicebv2.apps.local`. En plus des services réels, vous disposez désormais d'un maillage de service qui contient les ressources suivantes représentant les services réels :

- Deux services virtuels. Le proxy envoie tout le trafic du service virtuel `servicea.apps.local` vers le service virtuel `serviceb.apps.local` via un routeur virtuel.
- Trois nœuds virtuels nommés `serviceA`, `serviceB` et `serviceBv2`. Le proxy Envoy utilise les informations de découverte de service configurées pour les nœuds virtuels pour rechercher les adresses IP des services réels.
- Un routeur virtuel avec un itinéraire qui indique au proxy Envoy d'acheminer 75 % du trafic entrant vers le nœud virtuel `serviceB` et 25 % du trafic vers le nœud virtuel `serviceBv2`.

## Étape 6 : Mettre à jour les services

Après avoir créé votre maillage, vous devez effectuer les tâches suivantes :

- Autorisez le proxy Envoy que vous déployez avec chaque service à lire la configuration d'un ou de plusieurs nœuds virtuels. Pour plus d'informations sur la façon d'autoriser le proxy, consultez [Automatisation du proxy Envoy](#).
- Pour mettre à jour votre service existant, suivez les étapes ci-dessous.

Pour configurer une instance Amazon EC2 en tant que membre d'un nœud virtuel

1. Créez un rôle IAM.
  - a. Créez un fichier nommé `ec2-trust-relationship.json` avec les contenus suivants.

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

```
 "Action": "sts:AssumeRole"
 }
]
}
```

- b. Créez un rôle IAM à l'aide de la commande suivante.

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

2. Associez au rôle des politiques IAM qui lui permettent de lire depuis Amazon ECR et uniquement la configuration d'un nœud virtuel App Mesh spécifique.

- a. Créez un fichier nommé `virtual-node-policy.json` avec le contenu suivant. `apps` est le nom du maillage que vous avez créé dans [the section called “Étape 1 : Créer un maillage et un service virtuel”](#) et `serviceB` le nom du nœud virtuel que vous avez créé dans [the section called “Étape 2 : Créer un nœud virtuel”](#). Remplacez `111122223333` par votre identifiant de compte et `us-west-2` par la région dans laquelle vous *avez créé votre maillage*.

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

- b. Créez la politique à l'aide de la commande suivante.

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

- c. Attachez la politique que vous avez créée à l'étape précédente au rôle afin que le rôle puisse lire la configuration uniquement pour le nœud `serviceB` virtuel depuis 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. Associez la politique AmazonEC2ContainerRegistryReadOnly gérée au rôle afin qu'il puisse extraire l'image du conteneur Envoy depuis Amazon ECR.

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

3. [Lancez une instance Amazon EC2 avec le rôle IAM](#) que vous avez créé.
4. Connectez-vous à votre instance via SSH.
5. Installez Docker et le AWS CLI sur votre instance conformément à la documentation de votre système d'exploitation.
6. Authentifiez-vous auprès du référentiel Envoy Amazon ECR de la région à partir de laquelle vous souhaitez que votre client Docker extrait l'image.
  - Toutes les régions sauf me-south-1,ap-east-1,ap-southeast-3,eu-south-1,il-central-1,etaf-south-1. Vous pouvez remplacer *us-west-2* par [n'importe quelle région prise en charge](#), à l'exception de ap-east-1,, etap-southeast-3. eu-south-1 il-central-1 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
```

- Région 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
```

- Région 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. Exécutez l'une des commandes suivantes pour démarrer le conteneur App Mesh Envoy sur votre instance, en fonction de la région d'où vous souhaitez extraire l'image. Les valeurs *apps* et *serviceB* sont les noms de maillage et de nœud virtuel définis dans le scénario. Ces informations indiquent au proxy la configuration du nœud virtuel à lire dans App Mesh. Pour terminer le scénario, vous devez également suivre ces étapes pour les instances Amazon EC2 qui hébergent les services représentés par les nœuds *serviceA* virtuels *serviceBv2* et. Pour votre propre application, remplacez ces valeurs par les vôtres.
- Toutes les régions sauf *me-south-1*, *ap-east-1*, *ap-southeast-3*, *eu-south-1*, *il-central-1*, *etaf-south-1*. Vous pouvez remplacer le *code de région* par n'importe quelle [région prise en charge](#) *me-south-1*, à l'exception des régions *ap-east-1*, *ap-southeast-3*, *eu-south-1*, *il-central-1*, *etaf-south-1*. Vous pouvez remplacer *1337* par n'importe quelle valeur comprise entre 0 et 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:v1.29.5.0-prod
```

- Région *me-south-1*. Vous pouvez remplacer *1337* par n'importe quelle valeur comprise entre 0 et 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:v1.29.5.0-prod
```

- Région *ap-east-1*. Vous pouvez remplacer *1337* par n'importe quelle valeur comprise entre 0 et 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:v1.29.5.0-prod
```

**Note**

La APPMESH\_RESOURCE\_ARN propriété nécessite une version 1.15.0 ou une version ultérieure de l'image Envoy. Pour plus d'informations, consultez [Envoy](#).

**Important**

Seule la version v1.9.0.0-prod ou ultérieure est prise en charge pour une utilisation avec App Mesh.

- Sélectionnez `Show more` ci-dessous. Créez un fichier nommé `envoy-networking.sh` sur votre instance avec le contenu suivant. Remplacez `8000` par le port utilisé par le code de votre application pour le trafic entrant. Vous pouvez modifier la valeur de `APPMESH_IGNORE_UID`, mais la valeur doit être la même que celle que vous avez spécifiée à l'étape précédente, par exemple `1337`. Vous pouvez ajouter des adresses supplémentaires `APPMESH_EGRESS_IGNORED_IP` si nécessaire. Ne modifiez pas d'autres lignes.

```
#!/bin/bash -e

#
Start of configurable options
#

#APPMESH_START_ENABLED=""
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=""

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

```
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
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. Pour configurer les règles iptables afin d'acheminer le trafic d'application vers le proxy Envoy, exécutez le script que vous avez créé à l'étape précédente.

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

10. Démarrez votre code d'application de nœud virtuel.

### Note

Pour plus d'exemples et de procédures pas à pas pour App Mesh, consultez le référentiel [d'exemples d'App Mesh](#).

## App Mesh

Il s'agit de la feuille de route publique expérimentale pour AWS App Mesh. La feuille de route permet aux clients de connaître nos prochains produits et priorités, ce qui les aide à planifier leur utilisation future d'App Mesh. Elle contient des informations sur les travaux et permet à tous les AWS clients de donner des commentaires directs.

[Feuille de route App Mesh](#)

## App Mesh

Vous trouverez des end-to-end procédures pas-à-pas AWS App Mesh en action et des exemples de code pour l'intégration à différents AWS services dans le référentiel suivant :

[Exemples de maillage d'applications](#)

# App Mesh

App Mesh est composé des concepts suivants.

- [Mailles de service](#)
- [Services virtuels](#)
- [Passerelle](#)
- [Nœuds virtuels](#)
- [Routeurs virtuels](#)

## Mailles de service

Un maillage de service est une limite logique pour le trafic réseau entre les services qui résident dans celui-ci. Une fois que vous avez créé votre maillage de service, vous pouvez créer des services virtuels, des nœuds virtuels, des routeurs virtuels et des routes afin de répartir le trafic entre les applications de votre maillage.

## Créer un mesh de services

### Note


Lorsque vous créez un maillage, vous devez ajouter un sélecteur d'espace de noms. Si le sélecteur d'espace de noms est vide, il sélectionne tous les espaces de noms. Pour restreindre les espaces de noms, utilisez une étiquette pour associer les ressources App Mesh au maillage créé.

## AWS Management Console

Pour créer un mesh de services à l'aide de AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Sélectionnez Create mesh (Créer un maillage).
3. Pour Mesh name (Nom de maillage), spécifiez un nom pour votre maillage de service.
4. (Facultatif) Choisissez Autoriser le trafic externe. Par défaut, les proxys du maillage transfèrent uniquement le trafic entre eux. Si vous autorisez le trafic externe, les proxys


du maillage transfèrent également le trafic TCP directement aux services qui ne sont pas déployés avec un proxy défini dans le maillage.

 Note

Si vous spécifiez des backends sur un nœud virtuel lorsque vous utilisez `ALLOW_ALL`, vous devez spécifier toutes les sorties de ce nœud virtuel comme des backends. Dans le cas contraire, `ALLOW_ALL` ne fonctionnera plus pour ce nœud virtuel.

## 5. Version préféré

Contrôlez la version IP à utiliser pour le trafic au sein du maillage en activant l'option Ignorer le comportement de la version IP par défaut. Par défaut, App Mesh utilise différentes versions IP.

 Note

Le maillage applique la préférence IP à tous les nœuds virtuels et passerelles virtuelles au sein d'un maillage. Ce comportement peut être modifié sur un nœud virtuel individuel en définissant la préférence IP lorsque vous créez ou modifiez le nœud. La préférence IP ne peut pas être remplacée sur une passerelle virtuelle, car la configuration des passerelles virtuelles qui leur permet d'écouter le trafic IPv4 et IPv6 est la même quelle que soit la préférence définie sur le maillage.

- Par défaut
  - Le résolveur DNS d'Envoy préfère IPv6 et revient à IPv4.
  - Nous utilisons l'IPv4 adresse renvoyée AWS Cloud Map si elle est disponible et nous recommandons à utiliser l'IPv6 adresse.
  - Le point de terminaison créé pour l'application locale utilise une IPv4 adresse.
  - Les auditeurs Envoy se connectent à toutes les IPv4 adresses.
- IPv6 préféré
  - Le résolveur DNS d'Envoy préfère IPv6 et revient à IPv4.
  - L'IPv6 adresse renvoyée par AWS Cloud Map est utilisée si elle est disponible et revient à utiliser l'IPv4 adresse

- Le point de terminaison créé pour l'application locale utilise une IPv6 adresse.
  - Les auditeurs de The Envoy s'IPv4 IPv6 adressent à tous.
  - IPv4 préféré
    - Le résolveur DNS d'Envoy préfère IPv4 et revient à IPv6.
    - Nous utilisons l'IPv4 adresse renvoyée AWS Cloud Map si elle est disponible et nous recommandons à utiliser l'IPv6 adresse.
    - Le point de terminaison créé pour l'application locale utilise une IPv4 adresse.
    - Les auditeurs de The Envoy s'IPv4 IPv6 adressent à tous.
  - IPv6 uniquement
    - Le résolveur DNS d'Envoy utilise uniquement IPv6.
    - Seule l'IPv6 adresse renvoyée par AWS Cloud Map est utilisée. Si une IPv4 adresse est AWS Cloud Map renvoyée, aucune adresse IP n'est utilisée et des résultats vides sont renvoyés à l'Envoyé.
    - Le point de terminaison créé pour l'application locale utilise une IPv6 adresse.
    - Les auditeurs de The Envoy s'IPv4 IPv6 adressent à tous.
  - IPv4 uniquement
    - Le résolveur DNS d'Envoy utilise uniquement IPv4.
    - Seule l'IPv4 adresse renvoyée par AWS Cloud Map est utilisée. Si une IPv6 adresse est AWS Cloud Map renvoyée, aucune adresse IP n'est utilisée et des résultats vides sont renvoyés à l'Envoyé.
    - Le point de terminaison créé pour l'application locale utilise une IPv4 adresse.
    - Les auditeurs de The Envoy s'IPv4 IPv6 adressent à tous.
6. Sélectionnez Create mesh (Créer un maillage) pour terminer.
  7. (Facultatif) Partager le mesh avec d'autres comptes. Un maillage partagé permet aux ressources créées par différents comptes de communiquer entre elles dans le même maillage. Pour plus d'informations, veuillez consulter [Utilisation de maillages partagés](#).

## AWS CLI

Pour créer un maillage à l'aide du AWS CLI.

Créez un maillage de services à l'aide de la commande suivante (remplacez les valeurs *rouges* par les vôtres) :

1. `aws appmesh create-mesh --mesh-name meshName`

2. Exemple de sortie :

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

Pour plus d'informations sur la création d'un maillage à l'aide de l'AWS CLI for App Mesh, consultez la commande [create-mesh](#) dans la AWS CLI référence.

## Supprimer un maillage

### AWS Management Console

Pour supprimer une passerelle virtuelle à l'aide de l'AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh que vous voulez supprimer. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Dans la zone de confirmation, tapez **delete** puis cliquez sur Supprimer.

## AWS CLI

Pour supprimer un maillage à l'aide du AWS CLI

1. Utilisez la commande suivante pour supprimer votre maillage (remplacez les valeurs *rouges* par les vôtres) :

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

2. Exemple de sortie :

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

Pour plus d'informations sur la suppression d'un maillage à l'aide du AWS CLI for App Mesh, consultez la commande [delete-mesh](#) dans la AWS CLI référence.

## Services virtuels

Un service virtuel est une abstraction d'un service réel qui est fournie directement ou indirectement par un nœud virtuel, via un routeur virtuel. Les services dépendants appellent votre service virtuel via son `virtualServiceName`. Ces demandes sont acheminées vers le nœud virtuel ou le routeur virtuel qui est spécifié en tant que fournisseur pour le service virtuel.

# Création d'un service virtuel.

## AWS Management Console

Pour créer un service virtuel à l'aide de l'AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh dans lequel vous souhaitez créer le service virtuel. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual services (Services virtuels) dans le panneau de navigation gauche.
4. Choisissez Create virtual service (Créer un service virtuel).
5. Pour Virtual service name (Nom de service virtuel), choisissez un nom pour votre service virtuel. Vous pouvez choisir n'importe quel nom, mais le nom de découverte du service réel que vous ciblez, par exemple `my-service.default.svc.cluster.local`, est recommandé pour faciliter la corrélation entre vos services virtuels et les services réels. De cette façon, vous n'avez pas besoin de modifier votre code pour référencer un nom différent de celui auquel votre code référence actuellement. Le nom que vous spécifiez doit correspondre à une adresse IP sans bouclage, car le conteneur de l'application doit être en mesure de résoudre le nom avec succès avant que la demande ne soit envoyée au proxy Envoy. Vous pouvez utiliser n'importe quelle adresse IP sans boucle, car ni l'application ni les conteneurs proxy ne communiquent avec cette adresse IP. Le proxy communique avec les autres services virtuels par le biais des noms que vous leur avez configurés dans App Mesh, et non via des adresses IP auxquelles ces noms se réfèrent.
6. Pour Provider (Fournisseur), choisissez le type de fournisseur pour votre service virtuel :
  - Si vous souhaitez que le service virtuel répartisse le trafic sur plusieurs nœuds virtuels, sélectionnez Virtual routeur (Routeur virtuel), puis choisissez le routeur virtuel à utiliser dans le menu déroulant.
  - Si vous souhaitez que le service virtuel atteigne directement un nœud virtuel sans routeur virtuel, sélectionnez Nœud virtuel, puis choisissez le nœud virtuel à utiliser dans le menu déroulant.

### Note

App Mesh peut créer automatiquement une politique de réessai d'itinéraire Envoy par défaut pour chaque fournisseur de nœuds virtuels que vous définissez à compter du 29 juillet 2020, même si vous ne pouvez pas définir une telle politique



via l'API App Mesh. Pour plus d'informations, veuillez consulter [Politique de nouvelle tentative d'itinéraire par défaut](#).

- Si vous ne voulez pas que le service virtuel achemine le trafic pour l'instant (par exemple, si vos nœuds virtuels ou votre routeur virtuel n'existent pas encore), choisissez None (Aucun). Vous pourrez mettre à jour le fournisseur pour ce service virtuel plus tard.

7. Choisissez Create virtual service (Créer un service virtuel) pour terminer.

## AWS CLI

Pour créer un service virtuel à l'aide du AWS CLI.

Créez un service virtuel avec un fournisseur de nœuds virtuels à l'aide de la commande suivante et d'un fichier JSON d'entrée (remplacez les valeurs *rouges* par les vôtres) :

1. 

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

2. Contenu de l'exemple create-virtual-service-virtual-node.json :

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

3. Exemple de sortie :

```
{
 "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"
}
```

Pour plus d'informations sur la création d'un service virtuel à l'aide du AWS CLI for App Mesh, consultez la [create-virtual-service](#) commande dans la AWS CLI référence.

## Supprimer un service virtuel

### Note

Vous ne pouvez pas supprimer un service virtuel référencé par une route de passerelle. Vous devez d'abord supprimer la route de passerelle.

### AWS Management Console

Pour supprimer un service virtuel avec AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh dont vous souhaitez supprimer un service virtuel. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual services (Services virtuels) dans le panneau de navigation gauche.

4. Choisissez le service virtuel que vous souhaitez supprimer et cliquez sur Supprimer dans le coin supérieur droit. Vous pouvez uniquement supprimer une passerelle virtuelle lorsque votre compte est répertorié en tant que propriétaire de la ressource.
5. Dans la zone de confirmation, tapez **delete** puis cliquez sur Supprimer.

## AWS CLI

Pour supprimer un service virtuel avec AWS CLI

1. Utilisez la commande suivante pour supprimer votre service virtuel (remplacez les valeurs *rouges* par les vôtres) :

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

2. Exemple de sortie :

```
{
 "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"
 }
}
```

Pour plus d'informations sur la suppression d'un service virtuel à l'aide du AWS CLI for App Mesh, consultez la [delete-virtual-service](#) commande dans la AWS CLI référence.

## Passerelle

Une passerelle virtuelle permet aux ressources qui se trouvent à l'extérieur de votre maillage de communiquer avec les ressources qui se trouvent à l'intérieur de votre maillage. Une passerelle virtuelle est un proxy Envoy s'exécutant dans un service Amazon ECS, dans un service Kubernetes ou sur une instance Amazon EC2. Contrairement à un nœud virtuelle, qui représente Envoy s'exécutant avec une application, une passerelle virtuelle est Envoy déployé seul.

Les ressources externes doivent être en mesure de convertir un nom DNS en une adresse IP attribuée au service ou à l'instance qui exécute Envoy. Envoy peut ensuite accéder à l'ensemble de la configuration App Mesh pour les ressources qui se trouvent à l'intérieur du maillage. La configuration de gestion des demandes entrantes sur la passerelle virtuelle est spécifiée à l'aide de [Gateway Routes](#).

### Important

Une passerelle virtuelle dotée d'un écouteur HTTP ou HTTP2 réécrit le nom d'hôte de la demande entrante dans le nom du service virtuel cible de Gateway Route, et le préfixe correspondant de la Gateway Route est réécrit/, par défaut. Par exemple, si vous avez configuré la route Gateway, faites correspondre le préfixe à et/`chapter`, si la demande entrante l'est/`chapter/1`, la demande sera réécrite dans/`1`. Pour configurer les réécritures, reportez-vous à la section [Création d'un itinéraire de passerelle](#) à partir de Itinéraires de passerelle.

Lors de la création d'une passerelle virtuelle, `proxyConfiguration` elle ne `user` doit pas être configurée.

Pour effectuer une end-to-end procédure pas à pas, consultez [la section Configuration de la passerelle entrante](#).

## Création d'une passerelle virtuelle

### Note


Lorsque vous créez une passerelle virtuelle, vous devez ajouter un sélecteur d'espace de noms avec une étiquette pour identifier la liste des espaces de noms auxquels associer des itinéraires de passerelle à la passerelle virtuelle créée.

### AWS Management Console

Pour créer une passerelle virtuelle à l'aide de laAWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le maillage dans lequel vous voulez créer la passerelle virtuelle. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Passerelles virtuelles dans le menu de navigation de gauche.
4. Choisissez Créer une passerelle virtuelle.
5. Pour Nom de passerelle virtuelle, entrez un nom pour votre passerelle virtuelle.
6. (Facultatif, mais recommandé) Configurez les paramètres par défaut des politiques du client.
  - a. (Facultatif) Sélectionnez Appliquer le protocole TLS si vous souhaitez que la passerelle communique uniquement avec les services virtuels à l'aide du protocole TLS (Transport Layer Security).
  - b. (Facultatif) Dans Ports, spécifiez un ou plusieurs ports sur lesquels vous souhaitez appliquer la communication TLS avec les services virtuels.
  - c. Pour Méthode de validation, sélectionnez l'une des options suivantes. Le certificat que vous spécifiez doit déjà exister et répondre à des exigences spécifiques. Pour plus d'informations, veuillez consulter [Exigences du certificat](#).
    - AWS Private Certificate Authorityhébergement : sélectionnez un ou plusieurs certificats existants.
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupère à l'aide du Secret Discovery Service.
    - Hébergement de fichiers locaux : spécifiez le chemin d'accès au fichier de chaîne de certificats sur le système de fichiers sur lequel Envoy est déployé.

- d. (Facultatif) Saisissez un autre nom pour le message. Pour ajouter des réseaux SAN supplémentaires, sélectionnez Ajouter un réseau SAN. Les réseaux SAN doivent être au format FQDN ou URI.
  - e. (Facultatif) Sélectionnez Fournir un certificat client et l'une des options ci-dessous pour fournir un certificat client lorsqu'un serveur le demande et activer l'authentification TLS mutuelle. Pour en savoir plus sur le protocole TLS mutuel, consultez la documentation App Mesh [Mutual TLS Authentication](#).
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupère à l'aide du Secret Discovery Service.
    - Hébergement de fichiers locaux : spécifiez le chemin d'accès au fichier de chaîne de certificats, ainsi que la clé privée, sur le système de fichiers sur lequel Envoy est déployé. Pour une présentation complète du déploiement d'un maillage end-to-end à l'aide d'un exemple d'application utilisant le chiffrement de fichiers locaux, voir [Configuration du protocole TLS avec des certificats TLS fournis par des fichiers](#) activés GitHub.
7. (Facultatif) Pour configurer la journalisation, sélectionnez Journalisation. Entrez le chemin des journaux d'accès HTTP que vous souhaitez qu'Envoy utilise. Nous vous recommandons le `/dev/stdout` chemin afin que vous puissiez utiliser les pilotes de journal Docker pour exporter vos journaux Envoy vers un service tel qu'Amazon CloudWatch Logs.

 Note


Les journaux doivent toujours être ingérés par un agent dans votre application et envoyés à une destination. Ce chemin d'accès au fichier indique à Envoy où envoyer les journaux.

8. Configurez le récepteur.
  - a. Sélectionnez un protocole et spécifiez le port sur lequel Envoy écoute le trafic. L'écouteur HTTP permet la transition de connexion vers des websockets. Vous pouvez cliquer sur Ajouter un écouteur pour ajouter plusieurs écouteurs. Le bouton Supprimer supprimera cet écouteur.
  - b. (Facultatif) Activer le pool de connexions

Une passerelle virtuelle est un regroupement de connexions est limité le nombre de connexions que l'Envoy de passerelle virtuelle peut établir simultanément. Il est conçu

pour protéger votre instance Envoy contre le débordement de connexions et vous permet d'ajuster la gestion du trafic en fonction des besoins de vos applications.

Vous pouvez configurer les paramètres du pool de connexions côté destination pour un écouteur de passerelle virtuelle. App Mesh définit les paramètres du pool de connexions côté client sur une valeur infinie par défaut, ce qui simplifie la configuration du maillage.

 Note

Les protocoles `connectionPool` et `connectionPool PortMapping` doivent être identiques. Si votre protocole d'écoute est `grpc` ou `http2`, spécifiez `maxRequests` uniquement. Si votre protocole d'écoute l'est `http`, vous pouvez spécifier à la fois `maxConnections` et `maxPendingRequests`.

- Pour Nombre maximal de connexions, spécifiez le nombre maximum de connexions sortantes.
  - Pour le champ Nombre maximal de demandes, spécifiez le nombre maximal de demandes parallel pouvant être établies avec l'Envoy de passerelle virtuelle.
  - (Facultatif) Pour le nombre maximal de demandes en attente, spécifiez le nombre de demandes débordantes après le nombre maximal de connexions mises en file d'attente par un Envoy. La valeur par défaut est 2147483647.
- c. (Facultatif) Si vous souhaitez configurer un bilan de santé pour votre écouteur, sélectionnez Activer le contrôle de santé.

Une politique de bilan de Health est facultative, mais si vous spécifiez des valeurs pour une politique de Health, vous devez spécifier des valeurs pour le seuil sain, l'intervalle de contrôle de santé, le protocole de contrôle de santé, le délai d'expiration et le seuil d'insalubrité.

- Pour Protocole de vérification de l'état, choisissez un protocole. Si vous sélectionnez GRPC, votre service doit être conforme au [protocole de vérification de l'Health GRPC](#).
- Pour Health check port (Port de vérification de l'état), spécifiez le port sur lequel la vérification de l'état doit s'exécuter.
- Pour Healthy threshold (Seuil de santé), spécifiez le nombre de réussites consécutives de vérification de l'état qui doivent se produire avant de déclarer l'écouteur sain.

- Pour Health check interval (Intervalle de vérification de l'état), spécifiez la période en millisecondes entre chaque exécution de vérification de l'état.
  - Pour Path (Chemin) : spécifiez le chemin de destination pour la demande de vérification de l'état. Cette valeur n'est utilisée que si le protocole de bilan de Health est `http` ou `https`. La valeur est ignorée pour les autres protocoles.
  - Pour Délai d'attente, spécifiez le temps d'attente avant de recevoir une réponse de la vérification de l'état en millisecondes.
  - Pour Unhealthy threshold (Seuil non sain), spécifiez le nombre d'échecs consécutifs de vérification de l'état qui doivent se produire avant de déclarer l'écouteur non sain.
- d. (Facultatif) Si vous souhaitez spécifier si les clients communiquent avec cette passerelle virtuelle via TLS, sélectionnez Activer la terminaison TLS.
- Dans Mode, sélectionnez le mode pour lequel vous souhaitez configurer le protocole TLS sur l'écouteur.
  - Pour Méthode de certificat, sélectionnez l'une des options suivantes. Le certificat doit répondre à des exigences spécifiques. Pour plus d'informations, veuillez consulter [Exigences du certificat](#).
    - AWS Certificate Manager hébergement — Sélectionnez un certificat existant.
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupère à l'aide du Secret Discovery Service.
    - Hébergement de fichiers locaux : spécifiez le chemin d'accès à la chaîne de certificats et aux fichiers de clé privée sur le système de fichiers sur lequel Envoy est déployé.
  - (Facultatif) Sélectionnez Exiger un certificat client et l'une des options ci-dessous pour activer l'authentification TLS mutuelle si le client fournit un certificat. Pour en savoir plus sur le protocole TLS mutuel, consultez la documentation App Mesh [Mutual TLS Authentication](#).
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupère à l'aide du Secret Discovery Service.
    - Hébergement de fichiers locaux : spécifiez le chemin d'accès au fichier de chaîne de certificats sur le système de fichiers sur lequel Envoy est déployé.
  - (Facultatif) Saisissez un autre nom pour le message. Pour ajouter des réseaux SAN supplémentaires, sélectionnez Ajouter un réseau SAN. Les réseaux SAN doivent être au format FQDN ou URI.



9. Choisissez Créer une passerelle virtuelle pour terminer.

## AWS CLI

Pour créer une passerelle virtuelle à l'aide du AWS CLI.

Créez une passerelle virtuelle à l'aide de la commande suivante et saisissez JSON (remplacez les valeurs *rouges* par les vôtres) :

1. 

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

2. Contenu de l'exemple create-virtual-gateway .json :

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

3. Exemple de sortie :

```
{
 "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"
}
```

Pour plus d'informations sur la création d'une passerelle virtuelle à l'aide de l'AWS CLI for App Mesh, consultez la [create-virtual-gateway](#) commande dans la AWS CLI référence.

## Déploiement une passerelle virtuelle

Déployez un service Amazon ECS ou Kubernetes contenant uniquement le [conteneur Envoy](#). Vous pouvez également déployer le conteneur Envoy sur une instance Amazon EC2. Pour de plus amples informations, veuillez consulter [Mise en route avec App Mesh et Amazon EC2](#). Pour plus d'informations sur la façon de déployer sur Amazon ECS, consultez la section [Mise en route avec App Mesh et Amazon ECS](#) ou [Prise en main d'AWS App Mesh et Kubernetes](#) pour le déploiement sur Kubernetes. Vous devez définir la variable d'environnement `APPMESH_RESOURCE_ARN` sur `mesh/mesh-name/virtualGateway/virtual-gateway-name` et vous ne devez pas spécifier de configuration du proxy afin que le trafic du proxy ne soit pas redirigé vers lui-même. Par défaut, App Mesh utilise le nom de la ressource que vous avez spécifiée dans `APPMESH_RESOURCE_ARN` lorsque Envoy fait référence à lui-même dans les métriques et les traces. Vous pouvez remplacer ce comportement en définissant la variable d'environnement `APPMESH_RESOURCE_CLUSTER` avec votre propre nom.

Nous vous recommandons de déployer plusieurs instances du conteneur et de configurer un Network Load Balancer pour équilibrer la charge du trafic vers les instances. Le nom de découverte

de services de l'équilibreur de charge est le nom que vous souhaitez que les services externes utilisent pour accéder aux ressources du maillage, telles que *myapp.example.com*. Pour plus d'informations, consultez [Création d'un Network Load Balancer](#) (Amazon ECS), [Création d'un Load Balancer externe](#) (Kubernetes) ou [Tutoriel : Augmenter la disponibilité de votre application sur Amazon EC2](#). Vous pouvez également trouver d'autres exemples et procédures pas à pas dans nos [exemples d'App Mesh](#).

Activez l'autorisation par proxy pour Envoy. Pour plus d'informations, veuillez consulter [Automatisation du proxy Envoy](#).

## Supprimer une passerelle virtuelle

### AWS Management Console

Pour supprimer une passerelle virtuelle à l'aide de laAWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le maillage dont vous souhaitez supprimer une passerelle virtuelle. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Passerelles virtuelles dans le menu de navigation de gauche.
4. Choisissez la passerelle virtuelle que vous souhaitez supprimer et sélectionnez Supprimer. Vous ne pouvez pas supprimer une passerelle virtuelle si des routes de passerelle lui sont associées. Vous devez d'abord supprimer toutes les routes de passerelle associées. Vous pouvez uniquement supprimer une passerelle virtuelle lorsque votre compte est répertorié en tant que propriétaire de la ressource.
5. Dans la zone de confirmation, tapez **delete** puis sélectionnez Supprimer.

### AWS CLI

Pour supprimer une passerelle virtuelle à l'aide de laAWS CLI

1. Utilisez la commande suivante pour supprimer votre passerelle virtuelle (remplacez les valeurs *rouges* par les vôtres) :

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

## 2. Exemple de sortie :

```
{
 "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"
 }
}
```

Pour plus d'informations sur la suppression d'une passerelle virtuelle à l'aide du AWS CLI for App Mesh, consultez la [delete-virtual-gateway](#) commande dans la AWS CLI référence.

## itinéraires de passerelle

Une route de passerelle est attachée à une passerelle virtuelle et achemine le trafic vers un service virtuel existant. Si un itinéraire correspond à une demande, il peut distribuer le trafic vers un service virtuel cible. Cette rubrique vous allez devoir utiliser les passerelle dans un maillage de service.


## Création d'un passerelle

### AWS Management Console


Pour créer un passerelle à l'aide deAWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le maillage dans lequel vous souhaitez créer l'itinéraire de passerelle. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Passerelles virtuelles dans le menu de navigation de gauche.
4. Choisissez la passerelle virtuelle à laquelle vous souhaitez associer une nouvelle route de passerelle. Si aucun n'est répertorié, vous devez d'abord [créer un passerelle virtuelle](#). Vous ne pouvez créer une route de passerelle que pour une passerelle virtuelle dont votre compte est répertorié en tant que propriétaire de la ressource.
5. Dans le tableau des itinéraires de passerelle, choisissez Créer un itinéraire de passerelle.
6. Dans Nom de la route de passerelle, spécifiez le nom à utiliser pour votre route de passerelle.
7. Pour le type de route Gateway, choisissez http, http2 ou grpc.
8. Sélectionnez un nom de service virtuel existant. Si aucun n'est répertorié, vous devez d'abord créer un [service virtuel](#).
9. Choisissez le port qui correspond à la cible pour le port du fournisseur de services virtuels. Le port du fournisseur de services virtuels est requis lorsque le fournisseur (routeur ou nœud) du service virtuel sélectionné possède plusieurs écouteurs.
10. (Facultatif) Dans Priorité, spécifiez la priorité de cette route de passerelle.
11. Pour la configuration Match, spécifiez :
  - Si http/http2 est le type sélectionné :
    - (Facultatif) Méthode - Spécifie l'en-tête de méthode à mettre en correspondance dans les requêtes http/http2 entrantes.
    - (Facultatif) Correspondance des ports - Faites correspondre le port pour le trafic entrant. La correspondance des ports est requise si cette passerelle virtuelle possède plusieurs écouteurs.
    - (Facultatif) Nom d'hôte exact/suffixe - Spécifie le nom d'hôte qui doit correspondre à la demande entrante d'acheminement vers le service virtuel cible.
    - (Facultatif) Prefix/Exact/Regex path - Méthode permettant de faire correspondre le chemin de l'URL.

- Correspondance du préfixe - Une demande correspondante par une route de passerelle est réécrite dans le nom du service virtuel cible et le préfixe correspondant est réécrit/, par défaut. Selon la manière dont vous configurez votre service virtuel, celui-ci peut utiliser un routeur virtuel pour acheminer la demande vers différents nœuds virtuels, en fonction de préfixes ou d'en-têtes spécifiques.

 Important

- Vous ne pouvez spécifier ni l'une `/aws-appmesh*` ni `/aws-app-mesh*` l'autre correspondance entre les préfixes. Ces préfixes sont réservés à une utilisation interne future d'App Mesh.
- Si plusieurs itinéraires de passerelle sont définis, une demande est mise en correspondance avec l'itinéraire dont le préfixe est le plus long. Par exemple, s'il existait deux routes de passerelle, l'une ayant le préfixe de `/chapter` et l'autre le préfixe de `/`, alors la demande de `www.example.com/chapter/` serait mise en correspondance avec la route de passerelle portant le `/chapter` préfixe.

 Note

Si vous activez la correspondance basée sur le chemin et le préfixe, App Mesh active la normalisation des chemins ([normalize\\_path](#) et [merge\\_slashes](#)) afin de minimiser la probabilité de vulnérabilités de confusion de chemin.

Des vulnérabilités de confusion de chemin se produisent lorsque les parties participant à la demande utilisent des représentations de chemin différentes.

- Correspondance exacte - Le paramètre exact désactive la correspondance partielle pour un itinéraire et garantit qu'il ne renvoie l'itinéraire que si le chemin correspond EXACTEMENT à l'URL actuelle.
- Regex match - Utilisé pour décrire les modèles dans lesquels plusieurs URL peuvent en fait identifier une seule page du site Web.
- (Facultatif) Paramètres de requête - Ce champ vous permet de faire correspondre les paramètres de la requête.
- (Facultatif) En-têtes - Spécifie les en-têtes pour http et http2. Elle doit correspondre à la demande entrante à router vers le service virtuel cible.

- Si `grpc` est le type sélectionné :
  - Type de correspondance du nom d'hôte et (facultatif) correspondance exacte/suffixe - Spécifie le nom d'hôte qui doit être mis en correspondance sur la demande entrante de routage vers le service virtuel cible.
  - nom du service `grpc` - Le service `grpc` agit comme une API pour votre application et est défini avec `ProtoBuf`.

 Important

Vous ne pouvez pas spécifier `/aws . app-mesh*` ou `aws . appmesh` pour le nom du service. Ces noms de service sont réservés à une utilisation interne future d'App Mesh.

- (Facultatif) Métadonnées - Spécifie les métadonnées pour `grpc`. Elle doit correspondre à la demande entrante de routage vers le service virtuel cible.

## 12. (Facultatif) Pour la configuration de réécriture :

- Si `http/http2` est le type sélectionné :
  - Si `Prefix` est le type de correspondance sélectionné :
    - Ignorer la réécriture automatique du nom d'hôte - Par défaut, le nom d'hôte est réécrit selon le nom du service virtuel cible.
    - Ignorer la réécriture automatique du préfixe - Lorsque cette option est activée, la réécriture du préfixe spécifie la valeur du préfixe réécrit.
  - Si `Exact Path` est le type de correspondance sélectionné :
    - Ignorez la réécriture automatique du nom d'hôte - par défaut, le nom d'hôte est réécrit selon le nom du service virtuel cible.
    - Réécriture du chemin - Spécifie la valeur du chemin réécrit. Aucun chemin par défaut.
  - Si `Regex Path` est le type de correspondance sélectionné :
    - Ignorez la réécriture automatique du nom d'hôte - par défaut, le nom d'hôte est réécrit selon le nom du service virtuel cible.
    - Réécriture du chemin - Spécifie la valeur du chemin réécrit. Aucun chemin par défaut.
- Si `grpc` est le type sélectionné :
  - Ignorer la réécriture automatique du nom d'hôte - Par défaut, le nom d'hôte est réécrit selon le nom du service virtuel cible.

### 13. Choisissez Créer un itinéraire de passerelle pour terminer.

## AWS CLI

Pour créer un itinéraire de passerelle à l'aide du AWS CLI.

Créez une route de passerelle à l'aide de la commande suivante et saisissez JSON (remplacez les valeurs *rouges* par les vôtres) :

```
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. Contenu de l'exemple create-gateway-route .json :

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

3. Exemple de sortie :

```
{
 "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"
}
```

Pour plus d'informations sur la création d'une route de passerelle à l'aide de l'AWS CLI for App Mesh, consultez la [create-gateway-route](#) commande dans la AWS CLI référence.

## Supprimer un passerelle

### AWS Management Console

Pour supprimer un itinéraire de passerelle à l'aide de l'AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).

2. Choisissez le maillage dont vous souhaitez supprimer un itinéraire de passerelle. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Passerelles virtuelles dans le menu de navigation de gauche.
4. Choisissez la passerelle virtuelle à partir de laquelle vous souhaitez supprimer une route de passerelle.
5. Dans le tableau des itinéraires de passerelle, choisissez l'itinéraire de passerelle que vous souhaitez supprimer et sélectionnez Supprimer. Vous ne pouvez supprimer une route de passerelle que si votre compte est répertorié en tant que propriétaire de la ressource.
6. Dans la zone de confirmation, tapez **delete** puis cliquez sur Supprimer.

## AWS CLI

Pour supprimer un itinéraire de passerelle à l'aide de AWS CLI

1. Utilisez la commande suivante pour supprimer votre itinéraire de passerelle (remplacez les valeurs *rouges* par les vôtres) :

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

2. Exemple de sortie :

```
{
 "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"
}
}
```

Pour plus d'informations sur la suppression d'une route de passerelle à l'aide du AWS CLI for App Mesh, consultez la [delete-gateway-route](#) commande dans la AWS CLI référence.

## Nœuds virtuels

Un nœud virtuel agit comme pointeur logique vers un groupe de tâches particulier, comme Amazon ECS service ou un déploiement Kubernetes. Lorsque vous créez un nœud virtuel, vous devez spécifier une méthode de découverte de services pour votre groupe de tâches. Tout trafic entrant attendu par votre nœud virtuel est spécifié en tant qu'écouteur. Tout service virtuel auquel un nœud virtuel envoie du trafic sortant est spécifié en tant que backend.

Les métadonnées de réponse pour votre nouveau nœud virtuel contiennent le nom de ressource Amazon (ARN) associé au nœud virtuel. Définissez cette valeur comme variable d'APP\_MESH\_RESOURCE\_ARN environnement pour le conteneur proxy Envoy de votre groupe de tâches dans votre définition de tâche Amazon ECS ou dans la spécification de votre pod Kubernetes. Par exemple, la valeur peut être `arn:aws:appmesh:us-west-2:111122223333:mesh/myMesh/virtualNode/myVirtualNode`. Cette valeur est ensuite mappée aux paramètres `node.id` et `node.cluster` Envoy. Vous devez utiliser l'image Envoy 1.15.0 ou une version ultérieure lorsque vous définissez cette variable. Pour plus d'informations sur les variables App Mesh Envoy, consultez [Envoy](#).

**Note**

Par défaut, App Mesh utilise le nom de la ressource que vous avez spécifiée dans `APPMESH_RESOURCE_ARN` lorsque Envoy fait référence à lui-même dans les métriques et les traces. Vous pouvez remplacer ce comportement en définissant la variable d'environnement `APPMESH_RESOURCE_CLUSTER` avec votre propre nom.

## Création d'un nœud virtuel

### AWS Management Console

Pour créer un nœud virtuel à l'aide du AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le maillage dans lequel vous souhaitez créer le nœud virtuel. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual nodes (Nœuds virtuels) dans le panneau de navigation gauche.
4. Choisissez Créer un nœud virtuel, puis spécifiez les paramètres de votre nœud virtuel.
5. Dans Nom du nœud virtuel, entrez le nom de votre nœud virtuel.
6. Pour la méthode de découverte des services, choisissez l'une des options suivantes :
  - DNS — Spécifiez le nom d'hôte DNS du service réel représenté par le nœud virtuel. Le proxy Envoy est déployé dans un Amazon VPC. Le proxy envoie des demandes de résolution de noms au serveur DNS configuré pour le VPC. Si le nom d'hôte est résolu, le serveur DNS renvoie une ou plusieurs adresses IP. Pour plus d'informations sur les paramètres DNS du VPC, consultez la section [Utilisation du DNS avec votre VPC](#). Pour le type de réponse DNS (facultatif), spécifiez les types de points de terminaison renvoyés par le résolveur DNS. Load Balancer signifie que le résolveur DNS renvoie un ensemble de points de terminaison équilibrés en charge. Les points de terminaison signifient que le résolveur DNS renvoie tous les points de terminaison. Par défaut, le type de réponse est supposé être Load Balancer.

**Note**

Si vous utilisez Route53, vous devez utiliser Load Balancer.

- **AWS Cloud Map**— Spécifiez un nom de service et un espace de noms HTTP existants. Facultativement, vous pouvez également spécifier les attributs qu'App Mesh peut AWS Cloud Map rechercher en sélectionnant Ajouter une ligne et en spécifiant une clé et une valeur. Seules les instances qui correspondent à toutes les paires clé/valeur spécifiées sont renvoyées. Pour pouvoir être utilisé AWS Cloud Map, votre compte doit avoir le rôle `AWSServiceRoleForAppMesh` [lié au service](#). Pour plus d'informations sur AWS Cloud Map, consultez le [Manuel du développeur AWS Cloud Map](#).
- **Aucun** : sélectionnez cette option si votre nœud virtuel n'attend aucun trafic entrant.

## 7. préférence de version IP


Contrôlez quelle version IP doit être utilisée pour le trafic au sein du maillage en activant l'option Annuler le comportement de la version IP par défaut. Par défaut, App Mesh utilise différentes versions IP.

### Note

La définition de la préférence IP sur le nœud virtuel ne remplace que la préférence IP définie pour le maillage sur ce nœud spécifique.

- **Par défaut**
  - Le résolveur DNS d'Envoy préfère IPv6 et revient à IPv4.
  - Nous utilisons l'IPv4adresse renvoyée AWS Cloud Map si elle est disponible et nous nous contentons de l'IPv6utiliser.
  - Le point de terminaison créé pour l'application locale utilise une IPv4 adresse.
  - Les écouteurs Envoy se lient à toutes les IPv4 adresses.
- **IPv6 préféré**
  - Le résolveur DNS d'Envoy préfère IPv6 et revient à IPv4.
  - L'IPv6adresse renvoyée par AWS Cloud Map est utilisée si elle est disponible et revient à l'utilisation de l'IPv4adresse
  - Le point de terminaison créé pour l'application locale utilise une IPv6 adresse.
  - Les écouteurs Envoy se lient à toutes les IPv6 adresses IPv4 et adresses.
- **IPv4 préféré**
  - Le résolveur DNS d'Envoy préfère IPv4 et revient à IPv6.

- Nous utilisons l'IPv4adresse renvoyée AWS Cloud Map si elle est disponible et nous nous contentons de l'IPv6utiliser.
  - Le point de terminaison créé pour l'application locale utilise une IPv4 adresse.
  - Les écouteurs Envoy se lient à toutes les IPv6 adresses IPv4 et adresses.
- IPv6 uniquement
    - Le résolveur DNS d'Envoy n'utilise IPv6 que.
    - Seule l'IPv6adresse renvoyée par AWS Cloud Map est utilisée. Si elle AWS Cloud Map renvoie une IPv4 adresse, aucune adresse IP n'est utilisée et les résultats vides sont renvoyés à l'Envoy.
    - Le point de terminaison créé pour l'application locale utilise une IPv6 adresse.
    - Les écouteurs Envoy se lient à toutes les IPv6 adresses IPv4 et adresses.
  - IPv4 uniquement
    - Le résolveur DNS d'Envoy n'utilise IPv4 que.
    - Seule l'IPv4adresse renvoyée par AWS Cloud Map est utilisée. Si elle AWS Cloud Map renvoie une IPv6 adresse, aucune adresse IP n'est utilisée et les résultats vides sont renvoyés à l'Envoy.
    - Le point de terminaison créé pour l'application locale utilise une IPv4 adresse.
    - Les écouteurs Envoy se lient à toutes les IPv6 adresses IPv4 et adresses.
8. (Facultatif) Paramètres de politique client par défaut : configurez les exigences par défaut lors de la communication avec les services virtuels principaux.

 Note

- Si vous souhaitez activer le protocole TLS (Transport Layer Security) pour un nœud virtuel existant, nous vous recommandons de créer un nouveau nœud virtuel, qui représente le même service que le nœud virtuel existant, sur lequel vous souhaitez activer le protocole TLS. Transférez ensuite progressivement le trafic vers le nouveau nœud virtuel à l'aide d'un routeur et d'un itinéraire virtuels. Pour plus d'informations sur la création d'un itinéraire et l'ajustement des pondérations pour la transition, consultez [Acheminements](#). Si vous mettez à jour un nœud virtuel existant desservant le trafic avec le protocole TLS, il est possible que les proxys Envoy du client en aval reçoivent le contexte de validation TLS avant que le proxy

Envoy du nœud virtuel que vous avez mis à jour ne reçoive le certificat. Cela peut provoquer des erreurs de négociation TLS sur les proxys Envoy en aval.

- [L'autorisation du proxy](#) doit être activée pour le proxy Envoy déployé avec l'application représentée par les nœuds virtuels du service principal. Lorsque vous activez l'autorisation du proxy, nous vous recommandons de limiter l'accès aux seuls nœuds virtuels avec lesquels ce nœud virtuel communique.

- (Facultatif) Sélectionnez Appliquer le protocole TLS si vous souhaitez que le nœud virtuel communique avec tous les backends à l'aide du protocole TLS (Transport Layer Security).
- (Facultatif) Si vous souhaitez uniquement exiger l'utilisation du protocole TLS pour un ou plusieurs ports spécifiques, entrez un numéro dans Ports. Pour ajouter des ports supplémentaires, sélectionnez Ajouter un port. Si vous ne spécifiez aucun port, le protocole TLS est appliqué à tous les ports.
- Pour Méthode de validation, sélectionnez l'une des options suivantes. Le certificat que vous spécifiez doit déjà exister et répondre à des exigences spécifiques. Pour plus d'informations, veuillez consulter [Exigences du certificat](#).
  - AWS Private Certificate Authority hébergement : sélectionnez un ou plusieurs certificats existants. Pour une description complète du déploiement d'un maillage avec un exemple d'application utilisant le chiffrement à l'aide d'un certificat ACM, voir [Configuration du protocole TLS avec AWS Certificate Manager activé](#). end-to-end GitHub
  - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupérera à l'aide du Secret Discovery Service.
  - Hébergement de fichiers local — Spécifiez le chemin d'accès au fichier de chaîne de certificats sur le système de fichiers sur lequel Envoy est déployé. Pour une description complète du déploiement d'un maillage avec un exemple d'application utilisant le chiffrement avec des fichiers locaux, voir [Configuration du protocole TLS avec des certificats TLS fournis par fichier activé](#). end-to-end GitHub
- (Facultatif) Entrez un autre nom de sujet. Pour ajouter des SAN supplémentaires, sélectionnez Ajouter un SAN. Les SAN doivent être au format FQDN ou URI.
- (Facultatif) Sélectionnez Fournir un certificat client et l'une des options ci-dessous pour fournir un certificat client lorsqu'un serveur le demande et activer l'authentification TLS mutuelle. Pour en savoir plus sur le protocole TLS mutuel, consultez la documentation relative à [l'authentification TLS mutuelle](#) App Mesh.

- Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupérera à l'aide du Secret Discovery Service.
  - Hébergement de fichiers local — Spécifiez le chemin d'accès au fichier de chaîne de certificats, ainsi que la clé privée, sur le système de fichiers sur lequel Envoy est déployé.
9. (Facultatif) Backends de service : spécifiez le service virtuel App Mesh avec lequel le nœud virtuel communiquera.
- Entrez un nom de service virtuel App Mesh ou un nom de ressource Amazon (ARN) complet pour le service virtuel avec lequel votre nœud virtuel communique.
  - (Facultatif) Si vous souhaitez définir des paramètres TLS uniques pour un backend, sélectionnez les paramètres TLS, puis sélectionnez Remplacer les paramètres par défaut.
  - (Facultatif) Sélectionnez Appliquer le protocole TLS si vous souhaitez que le nœud virtuel communique avec tous les backends à l'aide du protocole TLS.
  - (Facultatif) Si vous souhaitez uniquement exiger l'utilisation du protocole TLS pour un ou plusieurs ports spécifiques, entrez un numéro dans Ports. Pour ajouter des ports supplémentaires, sélectionnez Ajouter un port. Si vous ne spécifiez aucun port, le protocole TLS est appliqué à tous les ports.
  - Pour Méthode de validation, sélectionnez l'une des options suivantes. Le certificat que vous spécifiez doit déjà exister et répondre à des exigences spécifiques. Pour plus d'informations, veuillez consulter [Exigences du certificat](#).
    - AWS Private Certificate Authorityhébergement : sélectionnez un ou plusieurs certificats existants.
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupérera à l'aide du Secret Discovery Service.
    - Hébergement de fichiers local — Spécifiez le chemin d'accès au fichier de chaîne de certificats sur le système de fichiers sur lequel Envoy est déployé.
  - (Facultatif) Entrez un autre nom de sujet. Pour ajouter des SAN supplémentaires, sélectionnez Ajouter un SAN. Les SAN doivent être au format FQDN ou URI.
  - (Facultatif) Sélectionnez Fournir un certificat client et l'une des options ci-dessous pour fournir un certificat client lorsqu'un serveur le demande et activer l'authentification TLS mutuelle. Pour en savoir plus sur le protocole TLS mutuel, consultez la documentation relative à [l'authentification TLS mutuelle](#) App Mesh.



- Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupérera à l'aide du Secret Discovery Service.
- Hébergement de fichiers local — Spécifiez le chemin d'accès au fichier de chaîne de certificats, ainsi que la clé privée, sur le système de fichiers sur lequel Envoy est déployé.
- Pour ajouter des backends supplémentaires, sélectionnez Ajouter un backend.

## 10. (Facultatif) Journalisation

Pour configurer la journalisation, saisissez le chemin d'accès HTTP que vous souhaitez qu'Envoy utilise. Nous vous recommandons ce `/dev/stdout` chemin afin que vous puissiez utiliser les pilotes de journal Docker pour exporter vos journaux Envoy vers un service tel qu'Amazon CloudWatch Logs.

### Note

Les journaux doivent toujours être ingérés par un agent dans votre application et envoyés à une destination. Ce chemin d'accès au fichier indique à Envoy où envoyer les journaux.

## 11. Configuration de l'écouteur

Support des auditeurs HTTP, HTTP/2GRPC, et TCP protocoles. HTTPS n'est pas pris en charge.

- a. Si votre nœud virtuel attend du trafic entrant, spécifiez un port et un protocole pour le récepteur. L'écouteur HTTP permet la transition de connexion vers les websockets. Vous pouvez cliquer sur Ajouter un écouteur pour ajouter plusieurs écouteurs. Le bouton Supprimer supprimera cet écouteur.
- b. (Facultatif) Activer le pool de connexions

Le regroupement de connexions limite le nombre de connexions qu'un Envoy peut établir simultanément avec le cluster d'applications local. Il est conçu pour empêcher votre application locale d'être submergée par les connexions et vous permet d'ajuster la configuration du trafic en fonction des besoins de vos applications.

Vous pouvez configurer les paramètres du pool de connexions côté destination pour un écouteur de nœud virtuel. App Mesh définit les paramètres du pool de connexions côté client sur infini par défaut, ce qui simplifie la configuration du maillage.

**Note**

Les protocoles `ConnectionPool` et `PortMapping` doivent être identiques. Si le protocole de votre écouteur est `TCP`, spécifiez uniquement `MaxConnections`. Si le protocole de votre écouteur est `grpc` ou `http2`, spécifiez uniquement `MaxRequests`. Si le protocole de votre écouteur est `http`, vous pouvez spécifier à la fois `MaxConnections` et `maxPendingRequests`.

- Pour Nombre maximal de connexions, spécifiez le nombre maximal de connexions sortantes.
  - (Facultatif) Pour le nombre maximal de demandes en attente, spécifiez le nombre de demandes débordantes après le nombre maximal de connexions qu'un Envoy mettra en file d'attente. La valeur par défaut est 2147483647.
- c. (Facultatif) Activer la détection des valeurs aberrantes

La détection des valeurs aberrantes appliquée au niveau du client Envoy permet aux clients de prendre des mesures quasi immédiates sur les connexions présentant des défaillances connues. Il s'agit d'une forme de mise en œuvre d'un disjoncteur qui permet de suivre l'état de santé de chaque hôte dans le service en amont.

La détection des valeurs aberrantes détermine de manière dynamique si les points de terminaison d'un cluster en amont fonctionnent différemment des autres et les supprime de l'ensemble d'équilibrage de charge sain.

**Note**

Pour configurer efficacement la détection des valeurs aberrantes pour un nœud virtuel de serveur, la méthode de découverte de service de ce nœud virtuel peut être `AWS Cloud Map` soit `DNS` avec le champ de type de réponse défini sur `ENDPOINTS`. Si vous utilisez la méthode de découverte du service `DNS` avec le type de réponse `asLOADBALANCER`, le proxy Envoy ne choisira qu'une seule adresse IP pour le routage vers le service en amont. Cela annule le comportement de détection exceptionnel qui consiste à éjecter un hôte défectueux d'un ensemble d'hôtes. Reportez-vous à la section `Méthode de`


découverte de service pour plus de détails sur le comportement du proxy Envoy par rapport au type de découverte de service.

- Pour les erreurs du serveur, spécifiez le nombre d'erreurs 5xx consécutives requises pour l'éjection.
  - Pour Intervalle de détection des valeurs aberrantes, spécifiez l'intervalle de temps et l'unité entre les analyses par balayage par éjection.
  - Pour la durée d'éjection de base, spécifiez la durée de base et l'unité pendant lesquelles un hôte est éjecté.
  - Pour le pourcentage d'éjection, spécifiez le pourcentage maximal d'hôtes pouvant être éjectés dans le pool d'équilibrage de charge.
- d. (Facultatif) Activer le contrôle de santé : configurez les paramètres d'une politique de contrôle de santé.

Une politique de bilan de santé est facultative, mais si vous spécifiez des valeurs pour une politique de santé, vous devez spécifier des valeurs pour le seuil de santé, l'intervalle entre les contrôles de santé, le protocole de contrôle de santé, le délai d'expiration et le seuil d'anomalie.

- Pour le protocole Health check, choisissez un protocole. Si vous sélectionnez `grpc`, votre service doit être conforme au protocole [GRPC Health Checking](#) Protocol.
- Pour Health check port (Port de vérification de l'état), spécifiez le port sur lequel la vérification de l'état doit s'exécuter.
- Pour Healthy threshold (Seuil de santé), spécifiez le nombre de réussites consécutives de vérification de l'état qui doivent se produire avant de déclarer l'écouteur sain.
- Pour Health check interval (Intervalle de vérification de l'état), spécifiez la période en millisecondes entre chaque exécution de vérification de l'état.
- Pour Path (Chemin) : spécifiez le chemin de destination pour la demande de vérification de l'état. Cette valeur n'est utilisée que si le protocole Health check est `http` ou `http2`. La valeur est ignorée pour les autres protocoles.
- Pour Timeout period (Délai), spécifiez le délai d'attente pour recevoir une réponse de la vérification de l'état, en millisecondes.
- Pour Unhealthy threshold (Seuil non sain), spécifiez le nombre d'échecs consécutifs de vérification de l'état qui doivent se produire avant de déclarer l'écouteur non sain.

- e. (Facultatif) Activez la terminaison TLS : configurez la manière dont les autres nœuds virtuels communiquent avec ce nœud virtuel à l'aide du protocole TLS.
- Pour Mode, sélectionnez le mode pour lequel vous souhaitez que TLS soit configuré sur l'écouteur.
  - Pour Méthode de certificat, sélectionnez l'une des options suivantes. Le certificat doit répondre à des exigences spécifiques. Pour plus d'informations, veuillez consulter [Exigences du certificat](#).
    - AWS Certificate Manager hébergement — Sélectionnez un certificat existant.
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupérera à l'aide du Secret Discovery Service.
    - Hébergement de fichiers local — Spécifiez le chemin d'accès au fichier de chaîne de certificats, ainsi que la clé privée, sur le système de fichiers sur lequel le proxy Envoy est déployé.
  - (Facultatif) Sélectionnez Exiger des certificats client et l'une des options ci-dessous pour activer l'authentification TLS mutuelle lorsqu'un client fournit un certificat. Pour en savoir plus sur le protocole TLS mutuel, consultez la documentation relative à [l'authentification TLS mutuelle](#) App Mesh.
    - Hébergement Envoy Secret Discovery Service (SDS) — Entrez le nom du secret qu'Envoy récupérera à l'aide du Secret Discovery Service.
    - Hébergement de fichiers local — Spécifiez le chemin d'accès au fichier de chaîne de certificats sur le système de fichiers sur lequel Envoy est déployé.
  - (Facultatif) Entrez un autre nom de sujet. Pour ajouter des SAN supplémentaires, sélectionnez Ajouter un SAN. Les SAN doivent être au format FQDN ou URI.
- f. Temporisations (facultatives)

 Note

Si vous spécifiez un délai d'attente supérieur à la valeur par défaut, assurez-vous de configurer un routeur virtuel et un itinéraire avec un délai d'expiration supérieur à la valeur par défaut. Toutefois, si vous réduisez le délai d'expiration à une valeur inférieure à la valeur par défaut, il est facultatif de mettre à jour les délais dans Route. Pour plus d'informations, consultez la section [Routes](#).

- Délai d'expiration de la demande : vous pouvez spécifier un délai d'attente si vous avez sélectionné grpc, http ou http2 pour le protocole de l'écouteur. La valeur par défaut est de 15 secondes. La valeur de 0 désactive le délai d'expiration.
- Durée d'inactivité : vous pouvez spécifier une durée d'inactivité pour n'importe quel protocole d'écoute. La durée par défaut est 300 secondes.

12. Choisissez Create virtual node pour terminer.

## AWS CLI

Pour créer un nœud virtuel à l'aide du AWS CLI.

Créez un nœud virtuel qui utilise le DNS pour la découverte de services à l'aide de la commande suivante et d'un fichier JSON d'entrée (remplacez les valeurs *rouges* par les vôtres) :

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

2. Contenu de l'exemple create-virtual-node-dns .json :

```
{
 "meshName": "meshName",
 "spec": {
 "listeners": [
 {
 "portMapping": {
 "port": 80,
 "protocol": "http"
 }
 }
],
 "serviceDiscovery": {
 "dns": {
 "hostname": "serviceBv1.svc.cluster.local"
 }
 }
 },
 "virtualNodeName": "nodeName"
}
```

### 3. Exemple de sortie :

```
{
 "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"
 }
}
```

Pour plus d'informations sur la création d'un nœud virtuel avec le AWS CLI for App Mesh, consultez la [create-virtual-node](#) commande dans la AWS CLI référence.

## Supprimer un nœud virtuel

### Note

Vous ne pouvez pas supprimer un nœud virtuel s'il est spécifié en tant que cible dans un [itinéraire](#) ou en tant que fournisseur dans un [service virtuel](#).

### AWS Management Console

Pour supprimer un nœud virtuel à l'aide du AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le maillage dont vous souhaitez supprimer un nœud virtuel. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual nodes (Nœuds virtuels) dans le panneau de navigation gauche.
4. Dans le tableau Nœuds virtuels, choisissez le nœud virtuel que vous souhaitez supprimer, puis sélectionnez Supprimer. Pour supprimer un nœud virtuel, votre identifiant de compte doit être répertorié dans les colonnes du propriétaire du maillage ou du propriétaire de la ressource du nœud virtuel.
5. Dans la zone de confirmation, tapez **delete** puis sélectionnez Supprimer.

### AWS CLI

Pour supprimer un nœud virtuel à l'aide du AWS CLI

1. Utilisez la commande suivante pour supprimer votre nœud virtuel (remplacez les valeurs *rouges* par les vôtres) :

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

2. Exemple de sortie :

```
{
 "virtualNode": {
 "meshName": "meshName",
```

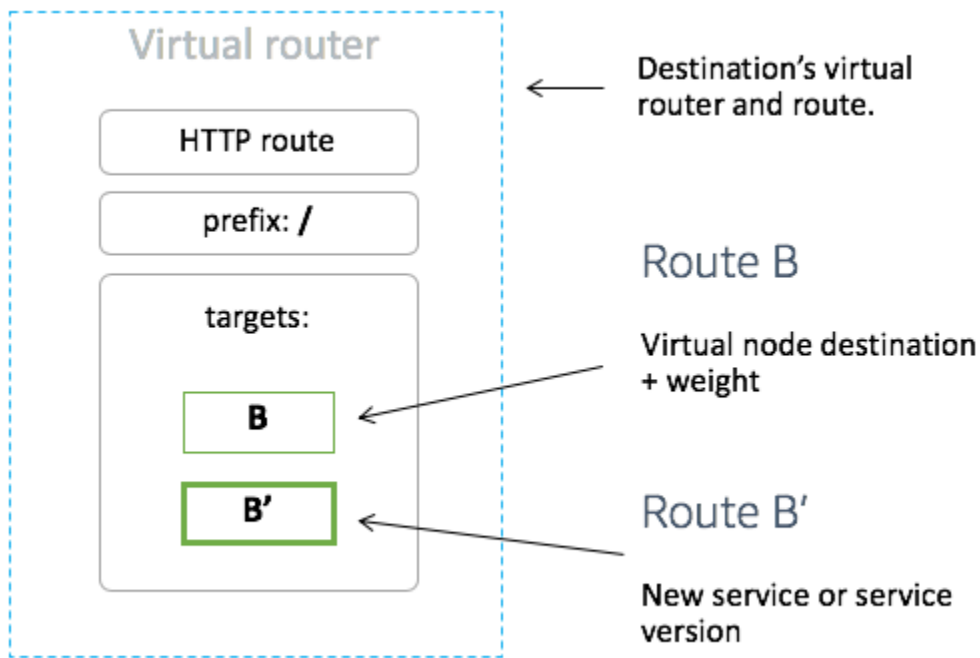
```
 "metadata": {
 "arn": "arn:aws:appmesh:us-west-2:210987654321:mesh/meshName/
virtualNode/nodeName",
 "createdAt": "2022-04-06T09:12:24.348000-05:00",
 "lastUpdatedAt": "2022-04-07T11:03:48.120000-05:00",
 "meshOwner": "123456789012",
 "resourceOwner": "210987654321",
 "uid": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",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

Pour plus d'informations sur la suppression d'un nœud virtuel avec le AWS CLI for App Mesh, consultez la [delete-virtual-node](#) commande dans la AWS CLI référence.

## Routeurs virtuels

Les routeurs virtuels gèrent le trafic d'un ou de plusieurs services virtuels au sein de votre mesh. Une fois que vous avez créé un routeur virtuel, vous pouvez créer et associer des routes pour votre routeur virtuel qui acheminent les demandes entrantes vers différents nœuds virtuels.





Tout trafic entrant attendu par votre routeur virtuel doit être spécifié en tant qu'écouteur.

## Création d'un routeur virtuel

### AWS Management Console

Pour créer un routeur virtuel à l'aide de l'AWS Management Console

#### Note

Lorsque vous créez un routeur virtuel, vous devez ajouter un sélecteur d'espace de noms avec une étiquette pour identifier la liste des espaces de noms auxquels associer des routes au routeur virtuel créé.

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh dans lequel vous souhaitez créer le routeur virtuel. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual routeurs (Routeurs virtuels) dans le panneau de navigation gauche.
4. Choisissez Create virtual router (Créer un routeur virtuel).

5. Pour Virtual router name (Nom de routeur virtuel), spécifiez un nom pour votre routeur virtuel. Il peut comporter jusqu'à 255 lettres, chiffres, points, tirets ou traits de soulignement.
6. (Facultatif) Pour la configuration du récepteur, spécifiez un port et un protocole pour votre routeur virtuel. L'httptécouteur permet la transition de connexion vers les websockets. Vous pouvez cliquer sur Ajouter un écouteur pour ajouter plusieurs écouteurs. Le bouton Supprimer supprimera cet écouteur.
7. Choisissez Create virtual router (Créer un routeur virtuel) pour terminer.

## AWS CLI

Pour créer un routeur virtuel à l'aide duAWS CLI.

Créez un routeur virtuel à l'aide de la commande suivante et saisissez JSON (remplacez les valeurs *rouges* par les vôtres) :

1. 

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

2. Contenu de l'exemple create-virtual-router .json

3. 

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

4. Exemple de sortie :

```
{
 "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"
}
```

Pour plus d'informations sur la création d'un routeur virtuel à l'aide de l'AWS CLI for App Mesh, consultez la [create-virtual-router](#) commande dans la [AWS CLI référence](#).

## Supprimer un routeur virtuel

### Note

Vous ne pouvez pas supprimer un routeur virtuel s'il possède des [routes](#) ou s'il est spécifié en tant que fournisseur pour un [service virtuel](#).

## AWS Management Console

Pour supprimer un routeur virtuel à l'aide de l'AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh dont vous souhaitez supprimer un routeur virtuel. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual routeurs (Routeurs virtuels) dans le panneau de navigation gauche.
4. Dans le tableau Routeurs virtuels, choisissez le routeur virtuel que vous souhaitez supprimer et sélectionnez Supprimer dans le coin supérieur droit. Pour supprimer un routeur virtuel, votre identifiant de compte doit être répertorié dans les colonnes Propriétaire du maillage ou Propriétaire de la ressource du routeur virtuel.
5. Dans la zone de confirmation, tapez **delete** puis cliquez sur Supprimer.

## AWS CLI

Pour supprimer un routeur virtuel à l'aide de l'AWS CLI

1. Utilisez la commande suivante pour supprimer votre routeur virtuel (remplacez les valeurs *rouges* par les vôtres) :

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

2. Exemple de sortie :

```
{
 "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"
 }
 }
```

Pour plus d'informations sur la suppression d'un routeur virtuel à l'aide de l'AWS CLI for App Mesh, consultez la [delete-virtual-router](#) commande dans la AWS CLI référence.

## Acheminements

Un itinéraire est associé à un routeur virtuel. La route est utilisée pour faire correspondre les demandes destinées au routeur virtuel et pour distribuer le trafic à ses nœuds virtuels associés. Si un itinéraire correspond à une demande, il peut distribuer le trafic vers un ou plusieurs nœuds virtuels cibles. Vous pouvez spécifier une pondération relative pour chaque nœud virtuel. Cette rubrique vous aide à utiliser les itinéraires dans un mesh de service.

### Création d'une route

#### AWS Management Console

Pour créer une route à l'aide de l'AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh dans lequel vous souhaitez créer la route. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual routeurs (Routeurs virtuels) dans le panneau de navigation gauche.

4. Choisissez le routeur virtuel auquel vous souhaitez associer une nouvelle route. Si aucun n'est répertorié, vous devez d'abord [créer un routeur virtuel](#).
5. Dans le tableau Routes, choisissez Create route (Créer une route). Pour créer un itinéraire, votre identifiant de compte doit être répertorié en tant que propriétaire de la ressource de l'itinéraire.
6. Pour Route name (Nom de route), indiquez le nom à utiliser pour votre route.
7. Dans Type d'itinéraire, choisissez le protocole que vous souhaitez router. Le protocole que vous sélectionnez doit correspondre au protocole d'écoute que vous avez sélectionné pour votre routeur virtuel et le nœud virtuel vers lequel vous acheminez le trafic.
8. (Facultatif) Pour Priorité d'itinéraire, spécifiez une priorité comprise entre 0 et 1 000 à utiliser pour votre itinéraire. La correspondance pour les routes se fait en fonction de la valeur spécifiée, où 0 est la priorité la plus haute.
9. (Facultatif) Choisissez Configuration supplémentaire. Parmi les protocoles ci-dessous, choisissez le protocole que vous avez sélectionné pour le type d'itinéraire et définissez les paramètres souhaités dans la console.
10. Pour la configuration de Target, sélectionnez le nœud virtuel App Mesh existant vers lequel acheminer le trafic et spécifiez un poids. Vous pouvez choisir Ajouter une cible pour ajouter des cibles supplémentaires. La somme des pourcentages pour toutes les cibles doit être égale à 100. Si aucun nœud virtuel n'est répertorié, vous devez d'abord en [créer](#) un. Si le nœud virtuel sélectionné possède plusieurs écouteurs, le port cible est requis.
11. Pour la configuration Match, spécifiez :

La configuration Match n'est pas disponible pour *tcp*

- Si http/http2 est le type sélectionné :
  - (Facultatif) La méthode - spécifie l'en-tête de méthode à mettre en correspondance dans les requêtes http/http2 entrantes.
  - (Facultatif) Correspondance des ports - Faites correspondre le port pour le trafic entrant. La correspondance des ports est requise si ce routeur virtuel possède plusieurs écouteurs.
  - (Facultatif) Prefix/Exact/Regex path - Méthode permettant de faire correspondre le chemin de l'URL.
    - Correspondance du préfixe - une demande correspondante par une route de passerelle est réécrite dans le nom du service virtuel cible et le préfixe correspondant est réécrit/, par défaut. Selon la manière dont vous configurez votre service virtuel,

celui-ci peut utiliser un routeur virtuel pour acheminer la demande vers différents nœuds virtuels, en fonction de préfixes ou d'en-têtes spécifiques.

 Note

Si vous activez la correspondance basée sur le chemin et le préfixe, App Mesh active la normalisation des chemins ([normalize\\_path](#) et [merge\\_slashes](#)) afin de minimiser la probabilité de vulnérabilités de confusion de chemin.

Des vulnérabilités de confusion de chemin se produisent lorsque les parties participant à la demande utilisent des représentations de chemin différentes.

- Correspondance exacte - le paramètre exact désactive la correspondance partielle pour un itinéraire et garantit qu'il ne renvoie l'itinéraire que si le chemin correspond EXACTEMENT à l'URL actuelle.
- Regex match - utilisé pour décrire des modèles dans lesquels plusieurs URL peuvent en fait identifier une seule page du site Web.
- (Facultatif) Paramètres de requête - Ce champ vous permet de faire correspondre les paramètres de la requête.
- (Facultatif) Headers - spécifie les en-têtes pour http et http2. Elle doit correspondre à la demande entrante à router vers le service virtuel cible.
- Si grpc est le type sélectionné :
  - Nom du service - le service de destination pour lequel la demande doit correspondre.
  - Nom de la méthode - la méthode de destination pour laquelle la demande doit correspondre.
  - (Facultatif) Les métadonnées - spécifient les enMatch fonction de la présence de métadonnées. Toutes doivent correspondre pour que la demande soit traitée.

12. Sélectionnez Créer un itinéraire.

## AWS CLI

Pour créer un itinéraire à l'aide du AWS CLI.

Créez une route gRPC à l'aide de la commande suivante et saisissez JSON (remplacez les valeurs *rouges* par les vôtres) :

1. 

```
aws appmesh create-route \
```

```
--cli-input-json file://create-route-grpc.json
```

## 2. Contenu de l'exemple 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. Exemple de sortie :

```
{
 "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"
}
}
```

Pour plus d'informations sur la création d'un itinéraire à l'aide de l'AWS CLI for App Mesh, consultez la commande [create-route](#) dans la [AWS CLI référence](#).

## gRPC

### (Facultatif) Match

- (Facultatif) Entrez le nom du service de destination correspondant à la demande. Si vous ne spécifiez pas de nom, les demandes vers n'importe quel service sont mises en correspondance.
- (Facultatif) Entrez le nom de la méthode de destination correspondant à la demande. Si vous ne spécifiez pas de nom, les demandes vers n'importe quelle méthode sont mises en correspondance. Si vous spécifiez un nom de méthode, vous devez spécifier un nom de service.

### (Facultatif) Métadonnées

Sélectionnez Add metadata (Ajouter des métadonnées).

- (Facultatif) Entrez le nom des métadonnées sur la base desquelles vous souhaitez acheminer, sélectionnez un type de correspondance et entrez une valeur de correspondance. Si vous sélectionnez Inverser, vous obtiendrez le résultat inverse. Par exemple, si vous spécifiez un nom de métadonnées demyMetadata, un type de correspondance Exact, une valeur de123 correspondance de et que vous sélectionnez Inverser, l'itinéraire est mis en correspondance pour toute demande dont le nom de métadonnées commence par un autre nom de123.
- (Facultatif) Sélectionnez Ajouter des métadonnées pour ajouter jusqu'à dix éléments de métadonnées.

### (Facultatif) Politique relative aux nouvelles tentatives

Une stratégie de nouvelle tentative permet aux clients de se protéger contre les défaillances intermittentes du réseau et les défaillances intermittentes côté serveur. Une politique de nouvelle tentative est facultative, mais recommandée. Les valeurs de délai d'attente de nouvelle tentative définissent le délai d'expiration de chaque nouvelle tentative (y compris la tentative initiale). Si vous ne définissez pas de politique de nouvelle tentative, App Mesh peut créer automatiquement une politique par défaut pour chacun de vos itinéraires. Pour plus d'informations, veuillez consulter [Politique de nouvelle tentative d'itinéraire par défaut](#).

- Dans le champ Délai d'expiration de la nouvelle tentative, entrez le nombre d'unités correspondant à la durée du délai d'expiration. Une valeur est requise si vous sélectionnez un événement de nouvelle tentative de protocole.
- Pour Réessayer l'unité de temporisation, sélectionnez une unité. Une valeur est requise si vous sélectionnez un événement de nouvelle tentative de protocole.
- Pour Entrées max, entrez le nombre maximal de nouvelles tentatives lorsque la demande échoue. Une valeur est requise si vous sélectionnez un événement de nouvelle tentative de protocole. Nous en recommandons une valeur d'au moins deux.
- Sélectionnez un ou plusieurs événements de nouvelle tentative HTTP. Nous vous recommandons de sélectionner au moins stream-error et gateway-error.
- Sélectionnez un événement de nouvelle tentative TCP.
- Sélectionnez un ou plusieurs événements de nouvelle tentative de gRPC. Nous vous recommandons de sélectionner au moins annulé et indisponible.

### (Facultatif) Délais d'expiration

- La valeur par défaut est 15 secondes. Si vous avez spécifié une politique de nouvelles tentatives, la durée que vous spécifiez ici doit toujours être supérieure ou égale à la durée des nouvelles tentatives multipliée par le nombre maximum de tentatives que vous avez défini dans la politique de nouvelles tentatives afin que votre politique de nouvelles tentatives puisse être exécutée. Si vous spécifiez une durée supérieure à 15 secondes, assurez-vous que le délai d'attente spécifié pour l'écouteur de tout nœud virtuel Target est également supérieur à 15 secondes. Pour plus d'informations, consultez [Virtual Nœuds](#).
- La valeur de 0 désactive le délai d'attente.
- Durée maximale pendant laquelle l'itinéraire peut être inactif.

### HTTP et HTTP/2

#### (Facultatif) Match

- Spécifiez le préfixe auquel l'itinéraire doit correspondre. Par exemple, si votre nom de service virtuel est `service-b.local` et que vous souhaitez que la route fasse correspondre des demandes à `service-b.local/metrics`, votre préfixe doit être `/metrics`. Spécifier/ les itinéraires pour tout le trafic.
- (Facultatif) Sélectionnez une méthode.
- (Facultatif) Sélectionnez un schéma. Applicable uniquement pour les routes HTTP2.

#### En-têtes (Facultatif)

- (Facultatif) Sélectionnez Ajouter un en-tête. Entrez le nom de l'en-tête sur lequel vous souhaitez effectuer le routage, sélectionnez un type de correspondance et entrez une valeur de correspondance. Si vous sélectionnez Inverser, vous obtiendrez le résultat inverse. Par exemple, si vous spécifiez un en-tête `clientRequestId` dont le préfixe est `et` et que vous sélectionnez Inverser, l'itinéraire est adapté à toute demande dont l'en-tête commence par une autre valeur que `123.123`
- (Facultatif) Sélectionnez Ajouter un en-tête. Vous pouvez ajouter jusqu'à dix en-têtes.

#### (Facultatif) Politique relative aux nouvelles tentatives

Une stratégie de nouvelle tentative permet aux clients de se protéger contre les défaillances intermittentes du réseau et les défaillances intermittentes côté serveur. Une politique de nouvelle

tentative est facultative, mais recommandée. Les valeurs de délai d'attente de nouvelle tentative définissent le délai d'expiration de chaque nouvelle tentative (y compris la tentative initiale). Si vous ne définissez pas de politique de nouvelle tentative, App Mesh peut créer automatiquement une politique par défaut pour chacun de vos itinéraires. Pour plus d'informations, veuillez consulter [Politique de nouvelle tentative d'itinéraire par défaut](#).

- Dans le champ Délai d'expiration de la nouvelle tentative, entrez le nombre d'unités correspondant à la durée du délai d'expiration. Une valeur est requise si vous sélectionnez un événement de nouvelle tentative de protocole.
- Pour Réessayer l'unité de temporisation, sélectionnez une unité. Une valeur est requise si vous sélectionnez un événement de nouvelle tentative de protocole.
- Pour Entrées max, entrez le nombre maximal de nouvelles tentatives lorsque la demande échoue. Une valeur est requise si vous sélectionnez un événement de nouvelle tentative de protocole. Nous en recommandons une valeur d'au moins deux.
- Sélectionnez un ou plusieurs événements de nouvelle tentative HTTP. Nous vous recommandons de sélectionner au moins stream-error et gateway-error.
- Sélectionnez un événement de nouvelle tentative TCP.

#### (Facultatif) Délais d'expiration

- Délai d'expiration de la demande : la valeur par défaut est de 15 secondes. Si vous avez spécifié une politique de nouvelles tentatives, la durée que vous spécifiez ici doit toujours être supérieure ou égale à la durée des nouvelles tentatives multipliée par le nombre maximum de tentatives que vous avez défini dans la politique de nouvelles tentatives afin que votre politique de nouvelles tentatives puisse être exécutée.
- Durée d'inactivité : la valeur par défaut est de 300 secondes.
- La valeur de 0 désactive le délai d'attente.

#### Note

Si vous spécifiez un délai d'attente supérieur au délai par défaut, assurez-vous que le délai spécifié pour l'écouteur pour tous les participants au nœud virtuel est également supérieur au délai par défaut. Toutefois, si vous réduisez le délai d'attente à une valeur inférieure à la valeur par défaut, il est facultatif de mettre à jour les délais d'expiration des nœuds virtuels. Pour plus d'informations, consultez [Virtual Nœuds](#).

## TCP

### (Facultatif) Délais d'expiration

- Durée d'inactivité : la valeur par défaut est de 300 secondes.
- La valeur de 0 désactive le délai d'attente.

## Suppression d'une route

### AWS Management Console

Pour supprimer un itinéraire à l'aide de AWS Management Console

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Choisissez le mesh dont vous souhaitez supprimer un itinéraire. Tous les maillages que vous possédez et qui ont été [partagés](#) avec vous sont répertoriés.
3. Choisissez Virtual routeurs (Routeurs virtuels) dans le panneau de navigation gauche.
4. Choisissez le routeur dont vous souhaitez supprimer un itinéraire.
5. Dans le tableau Itinéraires, choisissez l'itinéraire que vous souhaitez supprimer et sélectionnez Supprimer dans le coin supérieur droit.
6. Dans la zone de confirmation, tapez **delete** puis cliquez sur Supprimer.

### AWS CLI

Pour supprimer un itinéraire à l'aide de AWS CLI

1. Utilisez la commande suivante pour supprimer votre itinéraire (remplacez les valeurs *rouges* par les vôtres) :

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

2. Exemple de sortie :

```
{
 "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,
```

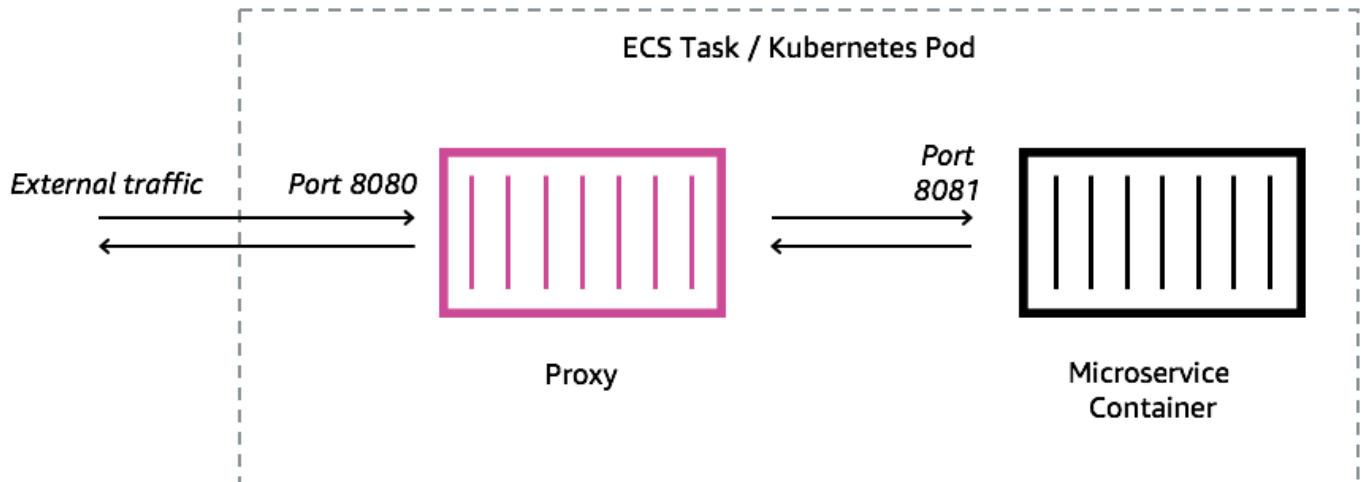
```
 "perRetryTimeout": {
 "unit": "s",
 "value": 15
 },
 "tcpRetryEvents": [
 "connection-error"
]
 },
 "priority": 100
},
"status": {
 "status": "DELETED"
},
"virtualRouterName": "routerName"
}
}
```

Pour plus d'informations sur la suppression d'un itinéraire à l'aide du AWS CLI for App Mesh, consultez la commande [delete-route](#) dans la AWS CLI référence.



# Image de l'envoyé

AWS App Mesh est un maillage de services basé sur le proxy [Envoy](#).



Vous devez ajouter un proxy Envoy à la tâche Amazon ECS, au pod Kubernetes ou à l'instance Amazon EC2 représentée par votre point de terminaison App Mesh, tel qu'un nœud virtuel ou une passerelle virtuelle. App Mesh vend une image de conteneur proxy Envoy corrigée avec les dernières mises à jour des vulnérabilités et des performances. App Mesh teste chaque nouvelle version du proxy Envoy par rapport à l'ensemble de fonctionnalités App Mesh avant de mettre une nouvelle image à votre disposition.

## Variantes d'image Envoy

App Mesh propose deux variantes de l'image du conteneur proxy Envoy. La différence entre les deux réside dans la manière dont le proxy Envoy communique avec le plan de données App Mesh et dans la manière dont les proxys Envoy communiquent entre eux. L'une est une image standard, qui communique avec les points de terminaison du service App Mesh standard. L'autre variante est conforme à la norme FIPS, qui communique avec les points de terminaison du service App Mesh FIPS et applique la cryptographie FIPS aux communications TLS entre les services App Mesh.

Vous pouvez choisir une image régionale dans la liste ci-dessous ou une image de notre [référentiel public](#) nommé `aws-appmesh-envoy`.

**⚠ Important**

- À compter du 30 juin 2023, seule l'image Envoy v1.17.2.0-prod ou une version ultérieure est compatible avec App Mesh. Pour les clients actuels qui utilisent déjà une image Envoy v1.17.2.0, même si les envoyés existants resteront compatibles, nous recommandons vivement de migrer vers la dernière version.
- En tant que bonne pratique, il est fortement recommandé de mettre régulièrement à jour la version d'Envoy vers la dernière version. Seule la dernière version d'Envoy est validée avec les derniers correctifs de sécurité, mises à jour de fonctionnalités et améliorations de performances.
- La version 1.17 était une mise à jour importante d'Envoy. Voir [Mise à jour/migration vers Envoy 1.17 pour](#) plus de détails.
- La version 1.20.0.1 ou ultérieure est ARM64 compatible.
- Pour le IPv6 support, la version Envoy 1.20 ou ultérieure est requise.

Toutes les régions [prises en charge](#) autres que me-south-1, ap-east-1, ap-southeast-3, eu-south-1, il-central-1, et af-south-1. Vous pouvez remplacer le *code de région* par n'importe quelle région autre que me-south-1, ap-east-1, ap-southeast-3, eu-south-1, il-central-1, et af-south-1

**Standard**

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

**Conforme à la norme FIPS**

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

**me-south-1****Standard**

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

**Conforme à la norme FIPS**

```
772975370895.dkr.ecr.me-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

## ap-east-1

### Standard

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

### Conforme à la norme FIPS

```
856666278305.dkr.ecr.ap-east-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

## ap-southeast-3

### Standard

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

### Conforme à la norme FIPS

```
909464085924.dkr.ecr.ap-southeast-3.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

## eu-south-1

### Standard

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

### Conforme à la norme FIPS

```
422531588944.dkr.ecr.eu-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

## il-central-1

### Standard

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

## Conforme à la norme FIPS

```
564877687649.dkr.ecr.il-central-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

## af-south-1

### Standard

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

## Conforme à la norme FIPS

```
924023996002.dkr.ecr.af-south-1.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

## Public repository

### Standard

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.5.0-prod
```

## Conforme à la norme FIPS

```
public.ecr.aws/appmesh/aws-appmesh-envoy:v1.29.5.0-prod-fips
```

### Note

Nous recommandons d'allouer 512 unités de processeur et au moins 64 MiB de mémoire au conteneur Envoy. Sur Fargate, la quantité minimale de mémoire que vous pouvez définir est de 1024 MiB de mémoire. L'allocation de ressources au conteneur Envoy peut être augmentée si les informations sur le conteneur ou d'autres indicateurs indiquent des ressources insuffisantes en raison d'une charge plus élevée.

### Note

Toutes les versions publiées à partir de `v1.22.0.0` sont créées sous forme d'`aws-appmesh-envoyimage` Docker sans distribution. Nous avons apporté cette modification

afin de réduire la taille de l'image et de réduire notre exposition aux vulnérabilités dans les packages non utilisés présents dans l'image. Si vous créez à partir d'une `aws-appmesh-envoy` image et que vous vous fiez à certains packages de base AL2 (par exemple yum) et à certaines fonctionnalités, nous vous suggérons de copier les fichiers binaires depuis l'intérieur d'une `aws-appmesh-envoy` image pour créer une nouvelle image Docker avec une base AL2.

Exécutez ce script pour générer une image docker personnalisée avec le 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 .
```

L'accès à cette image de conteneur dans Amazon ECR est contrôlé par AWS Identity and Access Management (IAM). Par conséquent, vous devez utiliser IAM pour vérifier que vous disposez d'un accès en lecture à Amazon ECR. Par exemple, lorsque vous utilisez Amazon ECS, vous pouvez attribuer un rôle d'exécution de tâche approprié à une tâche Amazon ECS. Si vous utilisez des politiques IAM qui limitent l'accès à des ressources Amazon ECR spécifiques, assurez-vous d'autoriser l'accès au nom de ressource Amazon (ARN) spécifique à la région qui identifie le `aws-appmesh-envoy` référentiel. Par exemple, dans la `us-west-2` région, vous autorisez l'accès à la ressource suivante : `arn:aws:ecr:us-west-2:840364872350:repository/aws-appmesh-envoy` Pour de plus amples informations, veuillez consulter [Politiques gérées Amazon ECR](#). Si vous utilisez Docker sur une instance Amazon EC2, authentifiez Docker auprès du référentiel. Pour plus d'informations, consultez [Authentification de registre](#).

Nous publions occasionnellement de nouvelles fonctionnalités d'App Mesh qui dépendent des modifications apportées par Envoy qui n'ont pas encore été fusionnées avec les images d'Envoy en amont. Pour utiliser ces nouvelles fonctionnalités d'App Mesh avant que les modifications d'Envoy ne soient fusionnées en amont, vous devez utiliser l'image du conteneur App Mesh-vended Envoy. Pour obtenir la liste des modifications, consultez la [GitHubfeuille de route d'App Mesh concernant les problèmes liés](#) à l'Envoy Upstreamétiquette. Nous vous recommandons d'utiliser l'image du conteneur App Mesh Envoy comme meilleure option prise en charge.

## Variables de configuration Envoy

Utilisez les variables d'environnement suivantes pour configurer les conteneurs Envoy pour les groupes de tâches de vos nœuds virtuels App Mesh.

### Note

App Mesh Envoy 1.17 ne prend pas en charge l'API xDS v2 d'Envoy. Si vous utilisez des [variables de configuration Envoy](#) qui acceptent les fichiers de configuration Envoy, elles doivent être mises à jour avec la dernière API xDS v3.

## Variables obligatoires

La variable d'environnement suivante est requise pour tous les conteneurs App Mesh Envoy. Cette variable ne peut être utilisée qu'avec une version 1.15.0 ou une version ultérieure de l'image Envoy. Si vous utilisez une version antérieure de l'image, vous devez définir la APPMESH\_VIRTUAL\_NODE\_NAME variable à la place.

### APPMESH\_RESOURCE\_ARN

Lorsque vous ajoutez le conteneur Envoy à un groupe de tâches, définissez cette variable d'environnement sur l'ARN du nœud virtuel ou de la passerelle virtuelle que le groupe de tâches représente. La liste suivante contient des exemples d'ARN :

- Nœud virtuel – *arn:aws:appmesh : Region-code : 111122223333:mesh/meshName /virtualNode/ virtual NodeName*
- Passerelle virtuelle – *arn:aws:appmesh : Code de région : 111122223333:mesh/meshName /VirtualGateway/ virtual GatewayName*

Lorsque vous utilisez le [canal App Mesh Preview](#), les ARN doivent utiliser la région *us-west-2* et utiliser, au lieu de. `appmesh-preview` `appmesh` Par exemple, l'ARN d'un nœud virtuel dans le canal de prévisualisation de l'App Mesh est `arn:aws:appmesh-preview:us-west-2:111122223333:mesh/meshName/virtualNode/virtualNodeName`.

## Variables facultatives

La variable d'environnement suivante est facultative pour les conteneurs App Mesh Envoy.

### ENVOY\_LOG\_LEVEL

Spécifie le niveau de journalisation pour le conteneur Envoy.

Valeurs valides: `trace`, `debug`, `info`, `warn`, `error`, `critical`, `off`

Par défaut : `info`

### ENVOY\_INITIAL\_FETCH\_TIMEOUT

Spécifie le temps pendant lequel Envoy attend la première réponse de configuration du serveur de gestion pendant le processus d'initialisation.

Pour plus d'informations, consultez la section [Sources de configuration](#) dans la documentation d'Envoy. Lorsqu'il est défini sur `0`, il n'y a pas de délai d'attente.

Par défaut : `0`

### ENVOY\_CONCURRENCY

Définit l'option de ligne de `--concurrency` commande lors du démarrage de l'Envoy. Ce paramètre n'est pas défini par défaut. Cette option est disponible à partir de la version Envoy `v1.24.0.0-prod` ou supérieure.

Pour plus d'informations, consultez la section [Options de ligne de commande](#) dans la documentation d'Envoy.

## Variables d'administration

Utilisez ces variables d'environnement pour configurer l'interface administrative d'Envoy.

### ENVOY\_ADMIN\_ACCESS\_PORT

Spécifiez un port d'administration personnalisé sur lequel Envoy pourra écouter. Par défaut: `9901`.

**Note**

Le port d'administration Envoy doit être différent de n'importe quel port d'écoute sur la passerelle virtuelle ou le nœud virtuel

**ENVOY\_ADMIN\_ACCESS\_LOG\_FILE**

Spécifiez un chemin personnalisé dans lequel écrire les journaux d'accès d'Envoy. Par défaut: `/tmp/envoy_admin_access.log`.

**ENVOY\_ADMIN\_ACCESS\_ENABLE\_IPV6**

Active l'interface d'administration d'Envoy pour accepter le IPv6 trafic, ce qui permet à cette interface d'accepter à la fois IPv4 le trafic. IPv6 Par défaut, cet indicateur est défini sur `false` et Envoy n'écoute que le IPv4 trafic. Cette variable ne peut être utilisée qu'avec Envoy image version 1.22.0 ou ultérieure.

**Variables de l'agent**

Utilisez ces variables d'environnement pour configurer l' AWS App Mesh agent pour Envoy. Pour plus d'informations, consultez App Mesh [Agent pour Envoy](#).

**APPNET\_ENVOY\_RESTART\_COUNT**

Spécifie le nombre de fois que l'agent redémarre le processus proxy Envoy dans une tâche ou un pod en cours d'exécution s'il s'arrête. L'agent enregistre également l'état de sortie à chaque fois qu'Envoy quitte pour faciliter le dépannage. La valeur par défaut de cette variable est `0`. Lorsque la valeur par défaut est définie, l'agent ne tente pas de redémarrer le processus.

Par défaut : `0`

Maximum : `10`

**PID\_POLL\_INTERVAL\_MS**

Spécifie l'intervalle en millisecondes pendant lequel l'état du processus du proxy Envoy est vérifié par l'agent. La valeur par défaut est `100`.

Par défaut : `100`



Minimum : 100

Maximum : 1000

#### LISTENER\_DRAIN\_WAIT\_TIME\_S

Spécifie la durée en secondes pendant laquelle le proxy Envoy attend la fermeture des connexions actives avant la fin du processus.

Par défaut : 20

Minimum : 5

Maximum : 110

#### APPNET\_AGENT\_ADMIN\_MODE

Démarre le serveur d'interface de gestion de l'Agent et le lie à une adresse TCP ou à un socket Unix.

Valeurs valides : tcp, uds

#### APPNET\_AGENT\_HTTP\_PORT

Spécifiez un port à utiliser pour lier l'interface de gestion de l'agent en tcp mode. Assurez-vous que la valeur du port est > 1024 if uid != 0. Assurez-vous que le port est inférieur à 65535.

Par défaut : 9902

#### APPNET\_AGENT\_ADMIN\_UDS\_PATH

Spécifiez le chemin du socket de domaine Unix pour l'interface de gestion de l'agent en uds mode.

Par défaut : /var/run/ecs/appnet\_admin.sock

## Variables de suivi

Vous ne pouvez configurer aucun ou l'un des pilotes de suivi suivants.

### AWS X-Ray variables

Utilisez les variables d'environnement suivantes pour configurer App Mesh avec AWS X-Ray. Pour plus d'informations, consultez le [Guide du développeur AWS X-Ray](#).

## ENABLE\_ENVOY\_XRAY\_TRACING

Active le suivi X-Ray en l'utilisant `127.0.0.1:2000` comme point de terminaison du démon par défaut. Pour l'activer, définissez la valeur sur `1`. La valeur par défaut est `0`.

## XRAY\_DAEMON\_PORT

Spécifiez une valeur de port pour remplacer le port par défaut du daemon X-Ray : `2000`

## XRAY\_SAMPLING\_RATE

Spécifiez un taux d'échantillonnage pour remplacer le taux d'échantillonnage par défaut du traceur X-Ray de `0.05` (5 %). Spécifiez la valeur sous forme décimale comprise entre `0` et `1.00` (100 %). Cette valeur est remplacée si elle `XRAY_SAMPLING_RULE_MANIFEST` est spécifiée. Cette variable est prise en charge avec les images Envoy des versions ultérieures `v1.19.1.1-prod` et ultérieures.

## XRAY\_SAMPLING\_RULE\_MANIFEST

Spécifiez un chemin de fichier dans le système de fichiers du conteneur Envoy pour configurer les règles d'échantillonnage personnalisées localisées pour le traceur X-Ray. Pour plus d'informations, consultez la section [Règles d'échantillonnage](#) dans le Guide du AWS X-Ray développeur. Cette variable est prise en charge avec les images Envoy des versions ultérieures `v1.19.1.0-prod` et ultérieures.

## XRAY\_SEGMENT\_NAME

Spécifiez un nom de segment pour les traces afin de remplacer le nom du segment X-Ray par défaut. Par défaut, cette valeur sera définie comme `mesh/resourceName`. Cette variable est prise en charge avec la version image d'Envoy `v1.23.1.0-prod` ou une version ultérieure.

## Variables de suivi Datadog

Les variables d'environnement suivantes vous aident à configurer App Mesh avec le traceur de l'agent Datadog. Pour plus d'informations, consultez la section [Configuration de l'agent](#) dans la documentation Datadog.

## ENABLE\_ENVOY\_DATADOG\_TRACING

Active la collecte de traces par Datadog en `127.0.0.1:8126` tant que point de terminaison par défaut de l'agent Datadog. Pour l'activer, définissez la valeur sur `1` (la valeur par défaut est `0`).

## DATADOG\_TRACER\_PORT

Spécifiez une valeur de port pour remplacer le port par défaut de l'agent Datadog :. 8126

## DATADOG\_TRACER\_ADDRESS

Spécifiez une adresse IP pour remplacer l'adresse par défaut de l'agent Datadog :. 127.0.0.1

## DD\_SERVICE

Spécifiez un nom de service pour les traces afin de remplacer le nom de service Datadog par défaut :/. `envoy-meshName` `virtualNodeName` Cette variable est prise en charge avec les images Envoy des versions ultérieures `v1.18.3.0-prod` et ultérieures.

## Variables de suivi Jaeger

Utilisez les variables d'environnement suivantes pour configurer App Mesh avec le suivi Jaeger. Pour plus d'informations, consultez [Getting Started](#) dans la documentation de Jaeger. Ces variables sont prises en charge avec les images Envoy des versions `1.16.1.0-prod` et ultérieures.

## ENABLE\_ENVOY\_JAEGER\_TRACING

Active la collecte de traces Jaeger en utilisant `127.0.0.1:9411` comme point de terminaison Jaeger par défaut. Pour l'activer, définissez la valeur sur 1 (la valeur par défaut est 0).

## JAEGER\_TRACER\_PORT

Spécifiez une valeur de port pour remplacer le port Jaeger par défaut :. 9411

## JAEGER\_TRACER\_ADDRESS

Spécifiez une adresse IP pour remplacer l'adresse Jaeger par défaut :. 127.0.0.1

## JAEGER\_TRACER\_VERSION

Spécifiez si le collecteur a besoin de traces au format `PROTO` codé JSON ou codé. Par défaut, cette valeur sera définie sur `PROTO`. Cette variable est prise en charge avec la version image d'Envoy `v1.23.1.0-prod` ou une version ultérieure.

## Variable de suivi Envoy

Définissez la variable d'environnement suivante pour utiliser votre propre configuration de suivi.

## ENVOY\_TRACING\_CFG\_FILE

Spécifiez un chemin de fichier dans le système de fichiers du conteneur Envoy. Pour plus d'informations, consultez [config.trace.v3.Tracing](#) la documentation d'Envoy.

### Note

Si la configuration de suivi nécessite de spécifier un cluster de suivi, assurez-vous de configurer la configuration de cluster associée `static_resources` dans le même fichier de configuration de suivi. Par exemple, Zipkin possède un [collector\\_cluster](#) champ pour le nom du cluster qui héberge les collecteurs de traces, et ce cluster doit être défini de manière statique.

## DogStatsVariables D

Utilisez les variables d'environnement suivantes pour configurer App Mesh avec DogStats D. Pour plus d'informations, consultez la documentation [DogStatsD](#).

### ENABLE\_ENVOY\_DOG\_STATSD

Active les statistiques DogStats D en utilisant `127.0.0.1:8125` comme point de terminaison du démon par défaut. Pour l'activer, définissez la valeur sur `1`.

### STATSD\_PORT

Spécifiez une valeur de port pour remplacer le port du démon DogStats D par défaut.

### STATSD\_ADDRESS

Spécifiez une valeur d'adresse IP pour remplacer l'adresse IP du démon DogStats D par défaut. Par défaut: `127.0.0.1`. Cette variable ne peut être utilisée qu'avec une version `1.15.0` ou une version ultérieure de l'image Envoy.

### STATSD\_SOCKET\_PATH

Spécifiez un socket de domaine Unix pour le DogStats démon D. Si cette variable n'est pas spécifiée et que DogStats D est activé, cette valeur correspond par défaut au port d'adresse IP du démon DogStats D de `127.0.0.1:8125`. Si la `ENVOY_STATS_SINKS_CFG_FILE` variable spécifiée contient une configuration de puits de statistiques, elle remplace toutes les variables DogStats D. Cette variable est prise en charge avec la version image d'Envoy `v1.19.1.0-prod` ou une version ultérieure.

## Variables App Mesh

Les variables suivantes vous aident à configurer App Mesh.

### APPMESH\_PREVIEW

Définissez la valeur sur `1` pour vous connecter au point de terminaison App Mesh Preview Channel. Pour plus d'informations sur l'utilisation du canal App Mesh Preview, consultez [App Mesh MeshVersion préliminaire](#).

### APPMESH\_RESOURCE\_CLUSTER

Par défaut, App Mesh utilise le nom de la ressource que vous avez spécifiée `APPMESH_RESOURCE_ARN` lorsque Envoy fait référence à lui-même dans les métriques et les traces. Vous pouvez remplacer ce comportement en définissant la variable d'environnement `APPMESH_RESOURCE_CLUSTER` avec votre propre nom. Cette variable ne peut être utilisée qu'avec une version `1.15.0` ou une version ultérieure de l'image Envoy.

### APPMESH\_METRIC\_EXTENSION\_VERSION

Définissez la valeur sur `1` pour activer l'extension des métriques App Mesh. Pour plus d'informations sur l'utilisation de l'extension de métriques App Mesh, consultez [Extension Metrics pour App Mesh](#).

### APPMESH\_DUALSTACK\_ENDPOINT

Définissez la valeur sur `1` pour vous connecter au point de terminaison App Mesh Dual Stack. Lorsque cet indicateur est activé, Envoy utilise notre domaine compatible avec le double stack. Par défaut, cet indicateur est défini sur `false` et se connecte uniquement à notre IPv4 domaine. Cette variable ne peut être utilisée qu'avec Envoy image version `1.22.0` ou ultérieure.

## Variables de statistiques Envoy

Utilisez les variables d'environnement suivantes pour configurer App Mesh avec Envoy Stats. Pour plus d'informations, consultez la documentation d'[Envoy Stats](#).

### ENABLE\_ENVOY\_STATS\_TAGS

Permet l'utilisation de balises définies par App Mesh `appmesh.mesh` et `appmesh.virtual_node`. Pour plus d'informations, consultez [config.metrics.v3.TagSpecifier](#) dans la documentation d'Envoy. Pour l'activer, définissez la valeur sur `1`.

## ENVOY\_STATS\_CONFIG\_FILE

Spécifiez un chemin de fichier dans le système de fichiers du conteneur Envoy pour remplacer le fichier de configuration des balises Stats par défaut par le vôtre. Pour plus d'informations, consultez [config.metrics.v3. StatsConfig](#).

### Note

La définition d'une configuration de statistiques personnalisée incluant des filtres de statistiques peut amener Envoy à ne plus se synchroniser correctement avec l'état mondial de l'App Mesh. Il s'agit d'un [bogue](#) dans Envoy. Notre recommandation est de ne pas filtrer les statistiques dans Envoy. Si le filtrage est absolument nécessaire, nous avons répertorié quelques solutions dans ce [numéro](#) sur notre feuille de route.

## ENVOY\_STATS\_SINKS\_CFG\_FILE

Spécifiez un chemin de fichier dans le système de fichiers du conteneur Envoy pour remplacer la configuration par défaut par la vôtre. Pour plus d'informations, consultez [config.metrics.v3. StatsSink](#) dans la documentation d'Envoy.

## Variables déconseillées

Les variables `APPMESH_VIRTUAL_NODE_NAME` d'environnement ne `APPMESH_RESOURCE_NAME` sont plus prises en charge dans la version Envoy 1.15.0 ou ultérieure. Cependant, ils sont toujours pris en charge pour les maillages existants. Au lieu d'utiliser ces variables avec la version d'Envoy 1.15.0 ou une version ultérieure, utilisez-les `APPMESH_RESOURCE_ARN` pour tous les points de terminaison App Mesh.

## Paramètres par défaut d'Envoy définis par App Mesh

Les sections suivantes fournissent des informations sur les paramètres par défaut d'Envoy pour la politique de nouvelle tentative d'itinéraire et le disjoncteur définis par App Mesh.

### Politique de nouvelle tentative d'itinéraire par défaut


Si votre compte n'avait aucun maillage avant le 29 juillet 2020, App Mesh crée automatiquement une politique de nouvelle tentative d'itinéraire Envoy par défaut pour toutes les requêtes HTTP, HTTP/2 et gRPC dans n'importe quel maillage de votre compte à compter du 29 juillet 2020. Si votre compte

comportait des maillages avant le 29 juillet 2020, aucune politique par défaut n'a été créée pour les itinéraires Envoy qui existaient avant, le 29 juillet 2020 ou après. À moins que vous n'[ouvriez un ticket auprès de AWS l'assistance](#). Une fois que le support a traité le ticket, la politique par défaut est créée pour tous les futurs itinéraires Envoy créés par App Mesh à la date de traitement du ticket ou après cette date. Pour plus d'informations sur les politiques de nouvelle tentative d'itinéraire d'Envoy, consultez [config.route.v3. RetryPolicy](#) dans la documentation d'Envoy.

App Mesh crée une route Envoy lorsque vous créez une [route](#) App Mesh ou que vous définissez un fournisseur de nœuds virtuels pour un [service virtuel](#) App Mesh. Bien que vous puissiez créer une politique de nouvelle tentative d'itinéraire App Mesh, vous ne pouvez pas créer de politique de nouvelle tentative d'App Mesh pour un fournisseur de nœuds virtuels.


La politique par défaut n'est pas visible via l'API App Mesh. La politique par défaut n'est visible que via Envoy. Pour consulter la configuration, [activez l'interface d'administration](#) et envoyez une demande à Envoy pour `unconfig_dump`. La politique par défaut inclut les paramètres suivants :

- Nombre maximum de tentatives — 2
- Événements de nouvelle tentative de gRPC — UNAVAILABLE
- Événements de nouvelle tentative HTTP : 503

 Note

Il n'est pas possible de créer une politique de nouvelle tentative d'itinéraire App Mesh qui recherche un code d'erreur HTTP spécifique. Cependant, une politique de nouvelle tentative d'itinéraire App Mesh peut rechercher `server-error` ou `gateway-error`. Dans les deux cas, il y a 503 des erreurs. Pour plus d'informations, consultez [Acheminements](#).

- événement de nouvelle tentative TCP — et `connect-failure` `refused-stream`

 Note

Il n'est pas possible de créer une politique de nouvelle tentative d'itinéraire App Mesh qui recherche l'un ou l'autre de ces événements. Cependant, une politique de nouvelle tentative d'itinéraire App Mesh peut être recherchée `connection-error`, ce qui équivaut à `connect-failure`. Pour plus d'informations, consultez [Acheminements](#).

- Réinitialisation — Envoy tente une nouvelle tentative si le serveur en amont ne répond pas du tout (délai de déconnexion/réinitialisation/lecture).

## Disjoncteur par défaut

Lorsque vous déployez un Envoy dans App Mesh, les valeurs par défaut d'Envoy sont définies pour certains paramètres du disjoncteur. Pour plus d'informations, consultez la section [cluster.CircuitBreakers.Les seuils figurent](#) dans la documentation d'Envoy. Ces paramètres ne sont pas visibles via l'API App Mesh. Les paramètres ne sont visibles que via Envoy. Pour consulter la configuration, [activez l'interface d'administration](#) et envoyez une demande à Envoy pour `unconfig_dump`.

Si votre compte n'avait aucun maillage avant le 29 juillet 2020, App Mesh désactive efficacement les disjoncteurs pour chaque Envoy que vous déployez dans un maillage créé le 29 juillet 2020 ou après cette date en modifiant les valeurs par défaut d'Envoy pour les paramètres suivants. Si vous aviez des maillages sur votre compte avant le 29 juillet 2020, les valeurs par défaut d'Envoy sont définies pour tout Envoy que vous déployez dans App Mesh le 29 juillet 2020 ou après, sauf si vous [ouvrez un ticket auprès de l' AWS assistance](#). Une fois que le support a traité le ticket, les valeurs par défaut de l'App Mesh pour les paramètres Envoy suivants sont définies par App Mesh sur tous les Envoys que vous déployez après la date de traitement du ticket :

- **max\_requests** – 2147483647
- **max\_pending\_requests** – 2147483647
- **max\_connections** – 2147483647
- **max\_retries** – 2147483647

### Note

Peu importe si vos Envoys ont les valeurs par défaut du disjoncteur Envoy ou App Mesh, vous ne pouvez pas les modifier.

## Mise à jour/migration vers Envoy 1.17

### Service de découverte secrète avec SPIRE

Si vous utilisez SPIRE (SPIFFE Runtime Environment) avec App Mesh pour distribuer des certificats de confiance à vos services, vérifiez que vous utilisez au moins une version `0.12.0` de [l'agent SPIRE](#) (publiée en décembre 2020). Il s'agit de la première version qui peut prendre en charge les versions suivantes d'Envoy `1.16`.



## Modifications des expressions régulières

À partir d'Envoy 1.17, App Mesh configure Envoy pour qu'il utilise le moteur d'expressions régulières [RE2](#) par défaut. Ce changement est évident pour la plupart des utilisateurs, mais les correspondances dans Routes ou Gateway Routes n'autorisent plus les références anticipées ou rétrospectives dans les expressions régulières.

### Perspectives positives et négatives

**Positif** - Une prévision positive est une expression entre parenthèses qui commence par : `?=`

```
(?=example)
```

Ils sont particulièrement utiles lors du remplacement de chaînes, car ils permettent de faire correspondre une chaîne sans consommer les caractères nécessaires à la correspondance. App Mesh ne prenant pas en charge le remplacement de chaînes regex, nous vous recommandons de les remplacer par des correspondances régulières.

```
(example)
```

**Négatif** : une prévision négative est une expression entre parenthèses qui commence par. `?!`

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

Les expressions entre parenthèses sont utilisées pour affirmer qu'une partie de l'expression ne correspond pas à une entrée donnée. Dans la plupart des cas, vous pouvez les remplacer par un quantificateur nul.

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

Si l'expression elle-même est une classe de caractères, vous pouvez annuler la classe entière et la marquer comme facultative en utilisant `?`.

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

En fonction de votre cas d'utilisation, vous pourrez peut-être également modifier vos itinéraires pour gérer cela.

```
{
```

```

"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"
 }
 }
]
 }
 }
 }
}

```

La première correspondance de route recherche un en-tête commençant par « préfixe » mais non suivi d'un « suffixe ». La deuxième route fait correspondre tous les autres en-têtes commençant par « préfixe », y compris ceux qui se terminent par « suffixe ». Au lieu de cela, ils peuvent également être inversés afin de supprimer les prévisions négatives.

```

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

Cet exemple inverse les itinéraires pour accorder une priorité plus élevée aux en-têtes qui se terminent par « suffixe », et tous les autres en-têtes commençant par « préfixe » sont mis en correspondance dans l'itinéraire de priorité inférieure.

## Références rétrospectives

Une référence arrière permet d'écrire des expressions plus courtes en répétant un précédent groupe entre parenthèses. Ils ont ce formulaire.

```
(group1)(group2)\1
```

Une barre oblique inversée \ suivie d'un chiffre sert d'espace réservé pour le n-ième groupe entre parenthèses dans l'expression. Dans cet exemple, \1 est utilisé comme méthode alternative pour écrire (group1) une deuxième fois.

```
(group1)(group2)(group1)
```

Ils peuvent être supprimés en remplaçant simplement la référence arrière par le groupe référencé, comme dans l'exemple.

## Agent pour Envoyé

L'agent est un gestionnaire de processus intégré à l'image Envoy vendue pour App Mesh. L'agent veille à ce qu'Envoy continue de fonctionner, à rester en bonne santé et à réduire les temps d'arrêt. Il filtre les statistiques et les données auxiliaires d'Envoy pour fournir une vue détaillée du fonctionnement du proxy Envoy dans App Mesh. Cela peut vous aider à résoudre les erreurs connexes plus rapidement.

Vous pouvez utiliser l'agent pour configurer le nombre de fois que vous souhaitez redémarrer le proxy Envoy en cas de dysfonctionnement du proxy. En cas de panne, l'agent enregistre l'état de sortie concluant lorsque Envoy se ferme. Vous pouvez l'utiliser pour résoudre la panne. L'agent facilite également l'épuisement des connexions Envoy, ce qui contribue à améliorer la résilience de vos applications en cas de panne.

Configurez l'agent pour Envoy à l'aide des variables suivantes :

- `APPNET_ENVOY_RESTART_COUNT`— Lorsque cette variable est définie sur une valeur différente de zéro, l'agent tente de redémarrer le processus proxy Envoy jusqu'à la valeur que vous avez définie lorsqu'il estime que l'état du processus proxy est défectueux lors de l'interrogation. Cela permet de réduire les temps d'arrêt en permettant un redémarrage plus rapide que le remplacement d'une tâche ou d'un pod par l'orchestrateur de conteneurs en cas d'échec du contrôle de santé du proxy.
- `PID_POLL_INTERVAL_MS`— Lors de la configuration de cette variable, la valeur par défaut est conservée `100`. Lorsque cette valeur est définie, vous pouvez détecter et redémarrer plus rapidement le processus Envoy lorsqu'il se ferme par rapport au remplacement d'une tâche ou d'un pod via des contrôles de santé de l'orchestrateur de conteneurs.

- `LISTENER_DRAIN_WAIT_TIME_S`— Lorsque vous configurez cette variable, tenez compte du délai d'expiration de l'orchestrateur de conteneurs défini pour arrêter la tâche ou le pod. Par exemple, si cette valeur est supérieure au délai d'expiration de l'orchestrateur, le proxy Envoy ne peut se vider que pendant la durée jusqu'à ce que l'orchestrateur arrête de force la tâche ou le module.
- `APPNET_AGENT_ADMIN_MODE`— Lorsque cette variable est définie sur `tcp` ou `uds`, l'agent fournit une interface de gestion locale. Cette interface de gestion sert de point de terminaison sécurisé pour interagir avec le proxy Envoy et fournit les API suivantes pour les contrôles de santé, les données de télémétrie et résume les conditions de fonctionnement du proxy.
  - `GET /status`— Demande les statistiques d'Envoy et renvoie des informations sur le serveur.
  - `POST /drain_listeners`— Épuise tous les auditeurs entrants.
  - `POST /enableLogging?level=<desired_level>`— Modifie le niveau de journalisation d'Envoy sur tous les enregistreurs.
  - `GET /stats/prometheus`— Afficher les statistiques d'Envoy au format Prometheus.
  - `GET /stats/prometheus?usedonly`— Afficher uniquement les statistiques mises à jour par Envoy.

Pour plus d'informations sur les variables de configuration Agent, veuillez consulter [Variables de configuration Envoy](#).

Le nouvel AWS App Mesh agent est inclus dans les images Envoy optimisées pour App Mesh à partir de la version `1.21.0.0` et ne nécessite aucune allocation de ressources supplémentaires dans les tâches ou les modules des clients.

# Observabilité App Mesh

L'un des avantages de l'utilisation d'App Mesh est une meilleure visibilité de vos applications de microservices. App Mesh est capable de fonctionner avec de nombreuses solutions de journalisation, de métriques et de traçage différentes.

Le proxy Envoy et App Mesh proposent les types d'outils suivants pour vous aider à avoir une vue plus claire de vos applications et de vos proxys :

- [Journalisation](#)
- [Métriques](#)
- [Traçage](#)

## Journalisation


Lorsque vous créez vos nœuds virtuels et vos passerelles virtuelles, vous avez la possibilité de configurer les journaux d'accès Envoy. Dans la console, vous pouvez accéder à la section Journalisation du nœud virtuel et de la passerelle virtuelle pour créer ou modifier des flux de travail.

### Logging

#### HTTP access logs path - *optional*

The path used to send logging information for the virtual node. App Mesh recommends using the standard out I/O stream.

```
/dev/stdout
```

 Logs must still be ingested by an agent in your application and sent to a destination. This file path only instructs Envoy where to send the logs.

L'image précédente montre le chemin de journalisation des journaux `/dev/stdout` d'accès à Envoy.

Pour le format, spécifiez l'un des deux formats possibles, `json` ou `text`, et le modèle. `json` prend des paires de clés et les transforme en structure JSON avant de les transmettre à Envoy.

Le bloc de code suivant montre la représentation JSON que vous pouvez utiliser dans 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

Assurez-vous que votre modèle de saisie est valide pour Envoy, sinon Envoy rejettera la mise à jour et enregistrera les dernières modifications dans le `error state`.

Lorsque vous envoyez des journaux d'accès à `Envoy/dev/stdout`, ils sont mélangés aux journaux du conteneur Envoy. Vous pouvez les exporter vers un service de stockage et de traitement des journaux tel que CloudWatch Logs à l'aide de pilotes de journal Docker standard tels que `awslogs`. Pour plus d'informations, consultez [Utilisation du pilote de journal `awslogs` dans le Manuel du développeur Amazon ECS](#) dans le Manuel du développeur Amazon ECS. Pour n'exporter que les journaux d'accès à Envoy (et ignorer les autres journaux des conteneurs Envoy), vous pouvez définir le paramètre `ENVoy_LOG_LEVEL` sur `off`. Vous pouvez enregistrer une demande sans chaîne de requête en incluant la chaîne de format `%REQ_WITHOUT_QUERY(X?Y):Z%`. Pour des exemples, voir [ReqWithoutQueryFormatter](#). Pour plus d'informations, consultez [journalisation des accès](#) dans la documentation Envoy.

### Activer les journaux d'accès sur Kubernetes

Lorsque vous utilisez l'[App Mesh Controller pour Kubernetes](#), vous pouvez configurer des nœuds virtuels avec journalisation des accès en ajoutant la configuration de journalisation à la spécification du nœud virtuel, comme indiqué dans l'exemple suivant.

```

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"
```

Votre cluster doit disposer d'un redirecteur de journaux pour collecter ces journaux, tel que Fluentd. Pour plus d'informations, voir [Configurer Fluentd DaemonSet pour envoyer des CloudWatch journaux à Logs](#).

Envoy écrit également divers journaux de débogage à partir de ses filtres vers `stdout`. Ces journaux sont utiles pour obtenir des informations à la fois sur les communications d'Envoy avec App Mesh et service-to-service sur le trafic. Votre niveau de journalisation spécifique peut être configuré à l'aide de la variable d'`ENVOY_LOG_LEVEL` environnement. Par exemple, le texte suivant provient d'un exemple de journal de débogage indiquant le cluster auquel Envoy a fait correspondre une requête HTTP particulière.

```
[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 et Cloudwatch

[Firelens](#) est un routeur de journalisation de conteneurs que vous pouvez utiliser pour collecter des journaux pour Amazon ECS et AWS Fargate. Vous trouverez un exemple d'utilisation de Firelens dans notre [référentiel AWS d'échantillons](#).

Vous pouvez l'utiliser CloudWatch pour recueillir des informations de journalisation ainsi que des mesures. Vous trouverez plus d'informations CloudWatch à ce sujet dans la section [Exportation des statistiques](#) de la documentation App Mesh.



## Surveillance de votre application à l'aide des métriques Envoy

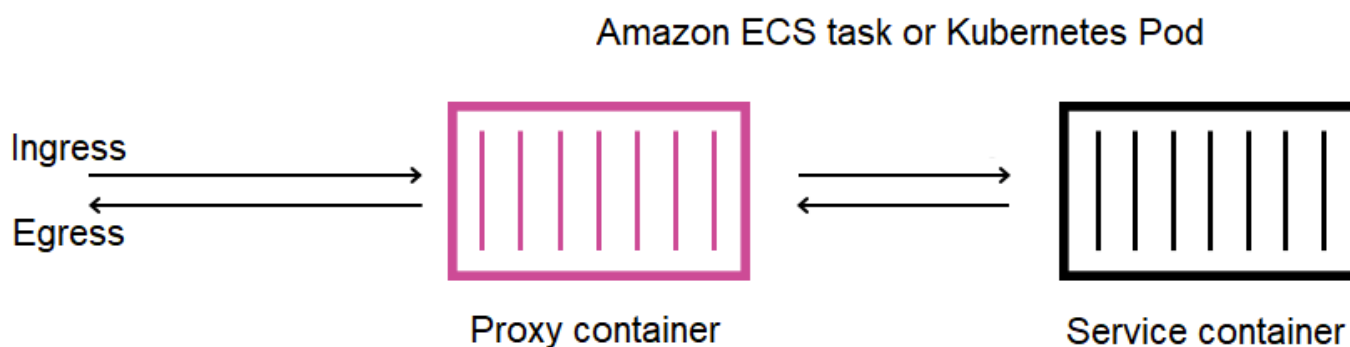
Envoy classe ses indicateurs dans les grandes catégories suivantes :

- En aval : mesures relatives aux connexions et aux demandes qui entrent dans le proxy.
- En amont : mesures relatives aux connexions sortantes et aux demandes effectuées par le proxy.
- Serveur : mesures qui décrivent l'état interne d'Envoy. Il s'agit notamment de mesures telles que la disponibilité ou la mémoire allouée.

Dans App Mesh, le proxy intercepte le trafic en amont et en aval. Par exemple, les demandes reçues de vos clients ainsi que les demandes effectuées par votre conteneur de services sont classées comme du trafic en aval par Envoy. Pour distinguer ces différents types de trafic en amont et en aval, App Mesh classe davantage les métriques Envoy en fonction de la direction du trafic par rapport à votre service :

- Entrée : mesures et ressources relatives aux connexions et aux demandes acheminées vers votre conteneur de services.
- Sortie : mesures et ressources relatives aux connexions et aux demandes qui proviennent de votre conteneur de services et, en fin de compte, de votre tâche Amazon ECS ou de votre pod Kubernetes.

L'image suivante montre la communication entre le proxy et les conteneurs de services.



Conventions de dénomination des ressources

Il est utile de comprendre comment Envoy visualise votre maillage et comment ses ressources correspondent aux ressources que vous définissez dans App Mesh. Voici les principales ressources Envoy configurées par App Mesh :

- **Récepteurs** : adresses et ports sur lesquels le proxy écoute les connexions en aval. Dans l'image précédente, App Mesh crée un écouteur d'entrée pour le trafic entrant dans votre tâche Amazon ECS ou votre pod Kubernetes et un écouteur de sortie pour le trafic sortant de votre conteneur de services.
- **Clusters** : groupe nommé de points de terminaison en amont auxquels le proxy se connecte et achemine le trafic. Dans App Mesh, votre conteneur de services est représenté sous la forme d'un cluster, ainsi que de tous les autres nœuds virtuels auxquels votre service peut se connecter.
- **Itinéraires** : ils correspondent aux itinéraires que vous définissez dans votre maillage. Ils contiennent les conditions selon lesquelles le proxy correspond à une demande ainsi que le cluster cible auquel une demande est envoyée.
- **Points de terminaison et affectations de charge des clusters** : adresses IP des clusters en amont. Lorsque vous l'utilisez AWS Cloud Map comme mécanisme de découverte de services pour des nœuds virtuels, App Mesh envoie les instances de service découvertes en tant que ressources de point de terminaison à votre proxy.
- **Secrets** : ils incluent, sans toutefois s'y limiter, vos clés de chiffrement et vos certificats TLS. Lorsque vous l'utilisez AWS Certificate Manager comme source pour les certificats client et serveur, App Mesh envoie des certificats publics et privés à votre proxy en tant que ressources secrètes.

App Mesh utilise un schéma cohérent pour nommer les ressources Envoy que vous pouvez utiliser pour établir un lien avec votre maillage.

Il est important de comprendre le schéma de dénomination des auditeurs et des clusters pour comprendre les métriques d'Envoy dans App Mesh.

## Noms des auditeurs

Les écouteurs sont nommés selon le format suivant :

```
lds_<traffic direction>_<listener IP address>_<listening port>
```

Vous verrez généralement les écouteurs suivants configurés dans Envoy :

- `lds_ingress_0.0.0.0_15000`

- `lds_egress_0.0.0.0_15001`

À l'aide d'un plugin CNI Kubernetes ou de règles de tables IP, le trafic de votre tâche Amazon ECS ou de votre pod Kubernetes est dirigé vers les ports 15000 et 15001. App Mesh configure Envoy avec ces deux écouteurs pour accepter le trafic entrant (entrant) et sortant (sortant). Si aucun écouteur n'est configuré sur votre nœud virtuel, vous ne devriez pas voir d'écouteur d'entrée.

## Noms des clusters

La plupart des clusters utilisent le format suivant :

```
cds_<traffic direction>_<mesh name>_<virtual node name>_<protocol>_<port>
```

Les nœuds virtuels avec lesquels vos services communiquent possèdent chacun leur propre cluster. Comme mentionné précédemment, App Mesh crée un cluster pour le service exécuté à côté d'Envoy afin que le proxy puisse lui envoyer du trafic entrant.

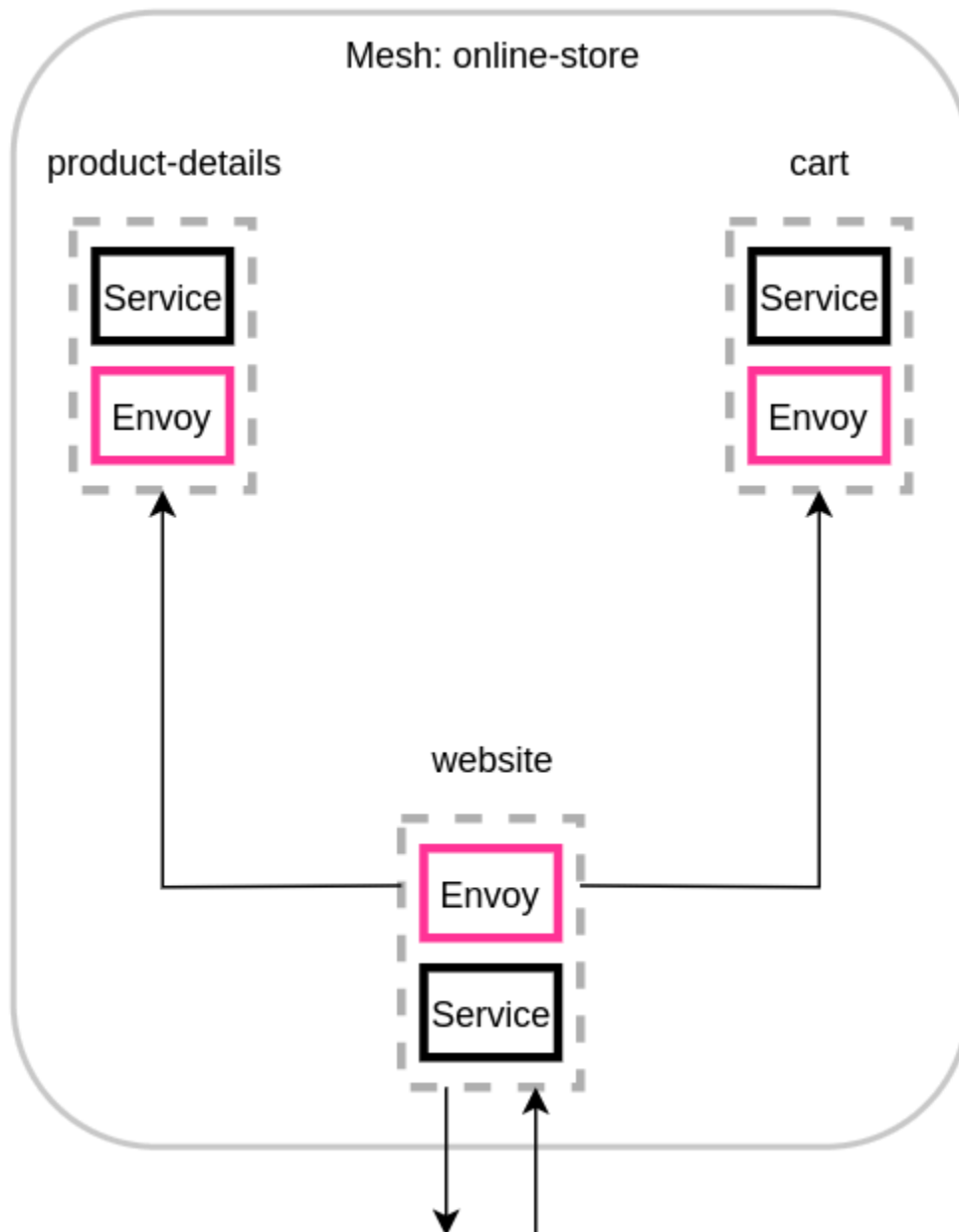
Par exemple, si vous avez nommé un nœud virtuel `my-virtual-node` qui écoute le trafic HTTP sur le port 8080 et que ce nœud virtuel se trouve dans un maillage nommé `my-mesh`, App Mesh crée un cluster nommé `cds_ingress_my-mesh_my-virtual-node_http_8080`. Ce cluster sert de destination au trafic entrant dans le conteneur `my-virtual-node` de service.

App Mesh peut également créer les types suivants de clusters spéciaux supplémentaires. Ces autres clusters ne correspondent pas nécessairement à des ressources que vous définissez explicitement dans votre maillage.

- Clusters utilisés pour accéder à d'autres AWS services. Ce type permet à votre maillage d'atteindre la plupart des services AWS par défaut : `cds_egress_<mesh name>_amazonaws`.
- Cluster utilisé pour effectuer le routage pour les passerelles virtuelles. Cela peut généralement être ignoré en toute sécurité :
  - Pour les auditeurs célibataires : `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<protocol>_<port>`
  - Pour plusieurs auditeurs : `cds_ingress_<mesh name>_<virtual gateway name>_self_redirect_<ingress_listener_port>_<protocol>_<port>`
- Le cluster dont vous pouvez définir le point de terminaison, tel que TLS, lorsque vous récupérez des secrets à l'aide du service Secret Discovery d'Envoy : `static_cluster_sds_unix_socket`.

## Exemples de métriques d'application

Pour illustrer les mesures disponibles dans Envoy, l'exemple d'application suivant comporte trois nœuds virtuels. Les services virtuels, les routeurs virtuels et les itinéraires du maillage peuvent être ignorés car ils ne sont pas reflétés dans les métriques d'Envoy. Dans cet exemple, tous les services écoutent le trafic HTTP sur le port 8080.



Nous vous recommandons d'ajouter la variable `ENABLE_ENVOY_STATS_TAGS=1` d'environnement aux conteneurs proxy Envoy exécutés dans votre maillage. Cela ajoute les dimensions métriques suivantes à toutes les métriques émises par le proxy :

- `appmesh.mesh`
- `appmesh.virtual_node`
- `appmesh.virtual_gateway`

Ces balises sont définies sur le nom du maillage, du nœud virtuel ou de la passerelle virtuelle afin de permettre le filtrage des mesures à l'aide des noms des ressources de votre maillage.

## Noms des ressources

Le proxy du nœud virtuel du site Web possède les ressources suivantes :

- Deux écouteurs pour le trafic entrant et sortant :
  - `lds_ingress_0.0.0.0_15000`
  - `lds_egress_0.0.0.0_15001`
- Deux clusters de sortie, représentant les deux nœuds principaux virtuels :
  - `cds_egress_online-store_product-details_http_8080`
  - `cds_egress_online-store_cart_http_8080`
- Un cluster d'entrée pour le conteneur de services du site Web :
  - `cds_ingress_online-store_website_http_8080`

## Exemples de mesures d'écoute

- `listener.0.0.0.0_15000.downstream_cx_active`—Nombre de connexions réseau d'entrée actives vers Envoy.
- `listener.0.0.0.0_15001.downstream_cx_active`—Nombre de connexions réseau de sortie actives vers Envoy. Les connexions effectuées par votre application à des services externes sont incluses dans ce décompte.
- `listener.0.0.0.0_15000.downstream_cx_total`—Nombre total de connexions réseau entrantes vers Envoy.
- `listener.0.0.0.0_15001.downstream_cx_total`—Nombre total de connexions réseau de sortie vers Envoy.

Pour obtenir l'ensemble complet des mesures relatives aux auditeurs, consultez la section [Statistiques](#) de la documentation Envoy.

## Exemples de mesures de cluster

- `cluster_manager.active_clusters`—Le nombre total de clusters auxquels Envoy a établi au moins une connexion.
- `cluster_manager.warming_clusters`—Le nombre total de clusters auxquels Envoy ne s'est pas encore connecté.

Les métriques de cluster suivantes utilisent le format `cluster.<cluster name>.<metric name>`. Ces noms de métriques sont propres à l'exemple d'application et sont émis par le conteneur Envoy du site Web :

- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_total`—Nombre total de connexions entre le site Web et les informations sur le produit.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_cx_connect_fail`—Nombre total d'échecs de connexion entre le site Web et les informations sur le produit.
- `cluster.cds_egress_online-store_product-details_http_8080.health_check.failure`—Nombre total de vérifications de santé infructueuses entre le site Web et les détails du produit.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_total`—Nombre total de demandes effectuées entre le site Web et les informations sur le produit.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_time`—Temps écoulé entre les demandes effectuées sur le site Web et les détails du produit.
- `cluster.cds_egress_online-store_product-details_http_8080.upstream_rq_2xx`—Nombre de réponses HTTP 2xx reçues par le site Web à partir des informations sur le produit.

Pour accéder à l'ensemble complet des métriques HTTP, consultez la section [Statistiques](#) de la documentation Envoy.

## Métriques du serveur de gestion

Envoy émet également des métriques liées à sa connexion au plan de contrôle App Mesh, qui fait office de serveur de gestion d'Envoy. Nous vous recommandons de surveiller certaines de ces mesures afin de vous avertir lorsque vos proxys sont désynchronisés du plan de contrôle pendant de longues périodes. La perte de connectivité au plan de contrôle ou l'échec des mises à jour empêchent vos proxys de recevoir de nouvelles configurations depuis App Mesh, y compris des modifications de maillage effectuées via les API App Mesh.

- `control_plane.connected_state`—Cette métrique est définie sur 1 lorsque le proxy est connecté à App Mesh, sinon elle est égale à 0.
- `*.update_rejected`—Nombre total de mises à jour de configuration rejetées par Envoy. Elles sont généralement dues à une mauvaise configuration de l'utilisateur. Par exemple, si vous configurez App Mesh pour lire un certificat TLS à partir d'un fichier qui ne peut pas être lu par Envoy, la mise à jour contenant le chemin d'accès à ce certificat est rejetée.
  - Si la mise à jour de Listener est rejetée, les statistiques seront `listener_manager.lds.update_rejected`.
  - Si la mise à jour du cluster est refusée, les statistiques seront `cluster_manager.cds.update_rejected`.
- `*.update_success`—Nombre de mises à jour de configuration réussies effectuées par App Mesh sur votre proxy. Il s'agit notamment de la charge utile de configuration initiale envoyée lors du démarrage d'un nouveau conteneur Envoy.
  - Pour que la mise à jour de Listener soit réussie, les statistiques seront `listener_manager.lds.update_success`.
  - Pour que la mise à jour du cluster soit réussie, les statistiques seront `cluster_manager.cds.update_success`.

Pour l'ensemble des mesures du serveur de gestion, consultez [Management Server](#) dans la documentation Envoy.

## Exportation des métriques

Envoy émet de nombreuses statistiques à la fois sur son propre fonctionnement et sur diverses dimensions du trafic entrant et sortant. Pour en savoir plus sur les statistiques d'Envoy, consultez la section [Statistiques](#) de la documentation Envoy. Ces mesures sont disponibles via le `/stats` point de terminaison du port d'administration du proxy, qui est généralement `9901`.

Lestat préfixe sera différent selon que vous utilisez un ou plusieurs écouteurs. Vous trouverez ci-dessous quelques exemples illustrant les différences.

### Warning

Si vous mettez à jour votre écouteur unique vers la fonctionnalité d'écoute multiple, vous risquez de subir un changement radical en raison du préfixe statistique mis à jour illustré dans le tableau suivant.

Nous vous suggérons d'utiliser `Envoy Image1.22.2.1-prod` ou une version ultérieure. Cela vous permet de voir des noms de métriques similaires dans votre point de terminaison Prometheus.

| Écouteur unique (SL) /Statistiques existantes avec préfixe d'écouteur « ingress » | Auditeurs multiples (ML) / Nouvelles statistiques avec « entrée ». <protocol>. <port>Préfixe « écouteur »                                             |  |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| <pre>http.*ingress*.rds .rds_ingress_http_ 5555.version_text</pre>                | <pre>http.*ingress.http .5555*.rds.rds_ing ress_http_5555.ver sion_text  http.*ingress.http .6666*.rds.rds_ing ress_http_6666.ver sion_text</pre>     |  |
| <pre>listener.0.0.0.0_1 5000.http.*ingress *.downstream_rq_2xx</pre>              | <pre>listener.0.0.0.0_1 5000.http.*ingress .http.5555*.downst ream_rq_2xx  listener.0.0.0.0_1 5000.http.*ingress .http.6666*.downst ream_rq_2xx</pre> |  |



|                                                                                   |                                                                                                           |  |
|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|--|
| Écouteur unique (SL) /Statistiques existantes avec préfixe d'écouteur « ingress » | Auditeurs multiples (ML) / Nouvelles statistiques avec « entrée ». <protocol>. <port>Préfixe « écouteur » |  |
| http.*ingress*.downstream_cx_length_ms                                            | http.*ingress.http.5555*.downstream_cx_length_ms<br><br>http.*ingress.http.6666*.downstream_cx_length_ms  |  |

Pour plus d'informations sur le point de terminaison des statistiques, consultez la section [Point de terminaison des statistiques](#) dans la documentation Envoy. Pour plus d'informations sur l'interface d'administration, consultez [Activer l'interface d'administration du proxy Envoy](#).

## Prometheus pour App Mesh avec Amazon EKS

Prometheus est une boîte à outils de surveillance et d'alerte open source. L'une de ses capacités est de spécifier un format d'émission de métriques pouvant être consommées par d'autres systèmes. Pour plus d'informations sur Prometheus, consultez [Présentation](#) dans la documentation Prometheus. Envoy peut émettre ses métriques via son point de terminaison de statistiques en transmettant le paramètre/stats?format=prometheus.

Pour les clients qui utilisent Envoy image build v1.22.2.1-prod, deux dimensions supplémentaires indiquent les statistiques spécifiques au récepteur d'entrée :

- appmesh.listener\_protocol
- appmesh.listener\_port

Vous trouverez ci-dessous une comparaison entre les statistiques existantes de Prometheus et les nouvelles statistiques.

- Statistiques existantes avec le préfixe d'écouteur « 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
```

- Nouvelles statistiques avec « ingress ». <protocol>. <port>"+ Appmesh Envoy Image v1.22.2.1-prod ou version ultérieure

```
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
```

- Nouvelles statistiques avec « ingress ». <protocol>. <port>« + Envoy Imagebuild personnalisé

```
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
```

Pour plusieurs auditeurs, le cluster<mesh name>\_<virtual gateway name>\_self\_redirect\_<ingress\_listener\_port>\_<protocol>\_<port> spécial sera spécifique à l'auditeur.

- Statistiques existantes avec le préfixe d'écouteur « ingress »

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-listeners-
mesh",appmesh_virtual_gateway="telligateway-vg",Mesh="multiple-listeners-
mesh",VirtualGateway="telligateway-vg",envoy_cluster_name="cds_ingress_multiple-
listeners-mesh_telligateway-vg_self_redirect_http_15001" } 0
```

- Nouvelles statistiques avec « ingress ». <protocol>. <port>»

```
envoy_cluster_assignment_stale{appmesh_mesh="multiple-
listeners-mesh",appmesh_virtual_gateway="telligateway-
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_telligateway-
vg_self_redirect_1111_http_15001" } 0
envoy_cluster_assignment_stale{appmesh_mesh="multiple-
listeners-mesh",appmesh_virtual_gateway="telligateway-
vg",envoy_cluster_name="cds_ingress_multiple-listeners-mesh_telligateway-
vg_self_redirect_2222_http_15001" } 0
```

## Installation de Prometheus

### 1. Ajoutez le référentiel EKS à Helm :

```
helm repo add eks https://aws.github.io/eks-charts
```

### 2. Installation d'App Mesh Prometheus

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system
```

## Exemple Prometheus

Voici un exemple de création d'un stockage persistant `PersistentVolumeClaim` pour Prometheus.

```
helm upgrade -i appmesh-prometheus eks/appmesh-prometheus \
--namespace appmesh-system \
--set retention=12h \
--set persistentVolumeClaim.claimName=prometheus
```

## Procédure pas à pas pour utiliser Prometheus

- [App Mesh avec EKS—Observabilité : Prometheus](#)

Pour en savoir plus sur Prometheus et Prometheus avec Amazon EKS

- [Documentation Prometheus](#)
- EKS - [Contrôlez les métriques du plan avec Prometheus](#)

## CloudWatch pour App Mesh

Transmission de statistiques Envoy à CloudWatch partir d'Amazon EKS

Vous pouvez installer l' CloudWatch agent sur votre cluster et le configurer pour collecter un sous-ensemble de mesures à partir de vos proxys. Si vous ne possédez pas encore de cluster Amazon

EKS, vous pouvez en créer un en suivant les étapes décrites dans [Walkthrough : App Mesh with Amazon EKS](#) activé GitHub. Vous pouvez installer un exemple d'application sur le cluster en suivant la même procédure pas à pas.

Pour définir les autorisations IAM appropriées pour votre cluster et installer l'agent, suivez les étapes décrites dans [Installer l' CloudWatch agent avec Prometheus Metrics Collection](#). L'installation par défaut contient une configuration Prometheus Scrape qui extrait un sous-ensemble utile de statistiques Envoy. Pour de plus amples informations, consultez [Métriques Prometheus pour App Mesh](#).

Pour créer un tableau de CloudWatch bord personnalisé App Mesh configuré pour afficher les mesures collectées par l'agent, suivez les étapes du didacticiel [Visualisation de vos métriques Prometheus](#). Vos graphiques commenceront à être remplis avec les mesures correspondantes au fur et à mesure que le trafic entre dans l'application App Mesh.

### Filtrage des métriques pour CloudWatch

L'[extension App Mesh Metrics](#) fournit un sous-ensemble de mesures utiles qui vous donnent un aperçu du comportement des ressources que vous définissez dans votre maillage. Étant donné que l' CloudWatchagent prend en charge l'extraction des métriques Prometheus, vous pouvez fournir une configuration de récupération pour sélectionner les métriques que vous souhaitez extraire d'Envoy et vers lesquelles vous souhaitez les envoyer CloudWatch.

Vous trouverez un exemple de capture de métriques à l'aide de Prometheus dans notre procédure pas à pas de [l'extension Metrics](#).

### CloudWatch Exemple

Vous trouverez un exemple de configuration CloudWatch dans notre [référentielAWS d'échantillons](#).

### Procédures pas à pas pour l'utilisation CloudWatch

- [Ajoutez des fonctionnalités de surveillance et de journalisation](#) dans notre [atelier App Mesh](#).
- [App Mesh avec EKS : observabilité : CloudWatch](#)
- [Utilisation de l'extension de statistiques d'App Mesh sur ECS](#)

## Extension Metrics pour App Mesh

Envoy génère des centaines de mesures réparties en plusieurs dimensions différentes. Les statistiques ne sont pas simples dans la mesure où elles se rapportent à App Mesh. Dans le cas des

services virtuels, il n'existe aucun mécanisme permettant de savoir avec certitude quel service virtuel communique avec un nœud virtuel ou une passerelle virtuelle donné.

L'extension App Mesh Metrics améliore les proxys Envoy exécutés dans votre maillage. Cette amélioration permet aux proxys d'émettre des mesures supplémentaires qui tiennent compte des ressources que vous définissez. Ce petit sous-ensemble de mesures supplémentaires vous permettra de mieux comprendre le comportement des ressources que vous avez définies dans App Mesh.

Pour activer l'extension App Mesh Metrics, définissez la variable `APPMESH_METRIC_EXTENSION_VERSION` d'environnement sur 1.

```
APPMESH_METRIC_EXTENSION_VERSION=1
```

Pour plus d'informations sur les variables de configuration Envoy, consultez [Variables de configuration Envoy](#).

Métriques liées au trafic entrant

- **ActiveConnectionCount**

- `envoy.appmesh.ActiveConnectionCount`— Nombre de connexions TCP actives.
- Dimensions : maille `VirtualNode`, `VirtualGateway`

- **NewConnectionCount**

- `envoy.appmesh.NewConnectionCount`— Nombre total de connexions TCP.
- Dimensions : maille `VirtualNode`, `VirtualGateway`

- **ProcessedBytes**

- `envoy.appmesh.ProcessedBytes`— Nombre total d'octets TCP envoyés et reçus de clients en aval.
- Dimensions : maille `VirtualNode`, `VirtualGateway`

- **RequestCount**

- `envoy.appmesh.RequestCount`— Le nombre de requêtes HTTP traitées.
- Dimensions : maille `VirtualNode`, `VirtualGateway`

- **GrpcRequestCount**

- `envoy.appmesh.GrpcRequestCount`— Le nombre de demandes GPRC traitées.
- Dimensions : maille `VirtualNode`, `VirtualGateway`

## Métriques relatives au trafic sortant

Vous verrez apparaître différentes dimensions sur vos mesures sortantes selon qu'elles proviennent d'un nœud virtuel ou d'une passerelle virtuelle.

### • **TargetProcessedBytes**

- `envoy . appmesh . TargetProcessedBytes`— Nombre total d'octets TCP envoyés et reçus de cibles en amont d'Envoy.
- Dimensions :
  - Dimensions des nœuds virtuels : maillage `VirtualNode`, `TargetVirtualService`, `TargetVirtualNode`
  - Dimensions de la passerelle virtuelle : maillage `VirtualGateway`, `TargetVirtualService`, `TargetVirtualNode`

### • **HTTPCode\_Target\_2XX\_Count**

- `envoy . appmesh . HTTPCode_Target_2XX_Count`— Nombre de requêtes HTTP adressées à une cible en amont d'Envoy qui ont donné lieu à une réponse HTTP 2xx.
- Dimensions :
  - Dimensions des nœuds virtuels : maillage `VirtualNode`, `TargetVirtualService`, `TargetVirtualNode`
  - Dimensions de la passerelle virtuelle : maillage `VirtualGateway`, `TargetVirtualService`, `TargetVirtualNode`

### • **HTTPCode\_Target\_3XX\_Count**

- `envoy . appmesh . HTTPCode_Target_3XX_Count`— Nombre de requêtes HTTP adressées à une cible en amont d'Envoy qui ont donné lieu à une réponse HTTP 3xx.
- Dimensions :
  - Dimensions des nœuds virtuels : maillage `VirtualNode`, `TargetVirtualService`, `TargetVirtualNode`
  - Dimensions de la passerelle virtuelle : maillage `VirtualGateway`, `TargetVirtualService`, `TargetVirtualNode`

### • **HTTPCode\_Target\_4XX\_Count**

- `envoy . appmesh . HTTPCode_Target_4XX_Count`— Nombre de requêtes HTTP adressées à une cible en amont d'Envoy qui ont donné lieu à une réponse HTTP 4xx.
- Dimensions :

- Dimensions des nœuds virtuels : maillage VirtualNode, TargetVirtualService, TargetVirtualNode
- Dimensions de la passerelle virtuelle : maillage VirtualGateway, TargetVirtualService, TargetVirtualNode
- **HTTPCode\_Target\_5XX\_Count**
  - `envoy.appmesh.HTTPCode_Target_5XX_Count`— Nombre de requêtes HTTP adressées à une cible en amont d'Envoy qui ont donné lieu à une réponse HTTP 5xx.
  - Dimensions :
    - Dimensions des nœuds virtuels : maillage VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensions de la passerelle virtuelle : maillage VirtualGateway, TargetVirtualService, TargetVirtualNode
- **RequestCountPerTarget**
  - `envoy.appmesh.RequestCountPerTarget`— Le nombre de demandes envoyées à une cible en amont d'Envoy.
  - Dimensions :
    - Dimensions des nœuds virtuels : maillage VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensions de la passerelle virtuelle : maillage VirtualGateway, TargetVirtualService, TargetVirtualNode
- **TargetResponseTime**
  - `envoy.appmesh.TargetResponseTime`— Le temps qui s'écoule entre le moment où une demande est adressée à une cible en amont d'Envoy et la réception de la réponse complète.
  - Dimensions :
    - Dimensions des nœuds virtuels : maillage VirtualNode, TargetVirtualService, TargetVirtualNode
    - Dimensions de la passerelle virtuelle : maillage VirtualGateway, TargetVirtualService, TargetVirtualNode

## Datadog pour App Mesh

Datadog est une application de surveillance et de sécurité destinée à la surveillance, aux mesures et à la journalisation de bout en bout des applications cloud. Datadog rend votre infrastructure, vos applications et les applications tierces totalement observables.

### Installation de Datadog

- EKS - Pour configurer Datadog avec EKS, procédez comme suit dans la [documentation Datadog](#).
- ECS EC2 - Pour configurer Datadog avec ECS EC2, procédez comme suit dans la [documentation Datadog](#).

### En savoir plus sur Datadog

- [Documentation Datadog](#)

## Tracing

### Important

Pour implémenter complètement le traçage, vous devez mettre à jour votre application. Pour voir toutes les données disponibles à partir du service que vous avez choisi, vous devez instrumenter votre application à l'aide des bibliothèques applicables.

## Surveillez le maillage des applications avec AWS X-Ray

AWS X-Ray est un service qui fournit des outils vous permettant de afficher, filtrer et avoir un aperçu des données collectées à partir des demandes servies par votre application. Ces informations vous aident à identifier les problèmes et les opportunités pour optimiser votre application. Vous pouvez consulter des informations détaillées sur les demandes et les réponses, ainsi que sur les appels en aval que votre application effectue vers d'autres AWS services.

X-Ray s'intègre à App Mesh pour gérer vos microservices Envoy. Les données de trace d'Envoy sont envoyées au démon X-Ray exécuté dans votre conteneur.

Implémentez X-Ray dans le code de votre application à l'aide du guide du [SDK](#) spécifique à votre langue.



## Activer le suivi X-Ray via App Mesh

- En fonction du type de service :
  - ECS - Dans la définition du conteneur proxy Envoy, définissez la variable d'ENABLE\_ENVOY\_XRAY\_TRACING environnement sur 1 et la variable d'XRAY\_DAEMON\_PORT environnement sur 2000.
  - EKS - Dans la configuration de l'App Mesh Controller, incluez `--set tracing.enabled=true` et `--set tracing.provider=x-ray`.
- Dans votre conteneur X-Ray, exposez le port 2000 et exécutez-le en tant qu'utilisateur 1337.

## Exemples X-Ray

### Définition d'un conteneur Envoy pour 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
 }
}
```

## Mise à jour du contrôleur App Mesh pour 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
```

## Procédures pas à pas pour utiliser le X-Ray

- [Moniteur avec AWS X-Ray](#)
- [App Mesh avec Amazon EKS - Observabilité : X-Ray](#)
- [Traçage distribué avec X-Ray](#) dans l'[AWS App Mesh atelier](#)

## Pour en savoir plus sur AWS X-Ray

- [AWS Documentation sur les X-Ray](#)

## Résolution des problèmes liés à AWS X-Ray avec App Mesh

- [Impossible de voir les traces de AWS X-Ray de mes applications.](#)

## Jaeger pour App Mesh avec Amazon EKS

Jaeger est un système de traçage distribué de bout en bout open source. Il peut être utilisé pour profiler des réseaux et pour la surveillance. Jaeger peut également vous aider à résoudre les problèmes d'applications cloud natives complexes.

Pour implémenter Jaeger dans le code de votre application, vous pouvez trouver le guide spécifique à votre langue dans les [bibliothèques de suivi](#) de la documentation Jaeger.

## Installation de Jaeger à l'aide de Helm

1. Ajoutez le référentiel EKS à Helm :

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Installation d'App Mesh Jaeger

```
helm upgrade -i appmesh-jaeger eks/appmesh-jaeger \
--namespace appmesh-system
```

## Exemple Jaeger

Voici un exemple de création d'un stockage persistant `PersistentVolumeClaim` pour 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
```

## Procédure pas à pas pour utiliser le Jaeger

- [App Mesh avec EKS—Observabilité : Jaeger](#)

## En savoir plus sur Jaeger

- [Documentation Jaeger](#)

## Datadog pour le suivi

Datadog peut être utilisé pour le suivi ainsi que pour les métriques. Pour plus d'informations et des instructions d'installation, consultez le guide spécifique à la langue de votre application dans la [documentation Datadog](#).

# Outillage App Mesh

App Mesh permet aux clients d'interagir indirectement avec ses API à l'aide d'outils tels que :

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- Contrôleur App Mesh pour Kubernetes
- Terraform

## App Mesh et AWS CloudFormation

AWS CloudFormation est un service qui vous permet de créer un modèle contenant toutes les ressources dont vous avez besoin pour votre application, puis AWS CloudFormation de configurer et de fournir les ressources pour vous. Il configurera également toutes les dépendances, afin que vous puissiez vous concentrer davantage sur votre application et moins sur la gestion des ressources.

Pour plus d'informations et des exemples d'utilisation AWS CloudFormation d'App Mesh, consultez la [AWS CloudFormation documentation](#).

## App Mesh et AWS CDK

AWS CDK est un framework de développement permettant d'utiliser du code pour définir votre infrastructure cloud et de l'utiliser AWS CloudFormation pour la provisionner. AWS CDK prend en charge plusieurs langages de programmation TypeScript JavaScript, notamment Python, Java et C#/Net.

Pour plus d'informations sur l'utilisation AWS CDK d'App Mesh, consultez la [AWS CDK documentation](#).

## Contrôleur App Mesh pour Kubernetes

Le contrôleur App Mesh pour Kubernetes vous aide à gérer les ressources de votre App Mesh pour un cluster Kubernetes et à injecter des sidecars dans des pods. Ce contrôleur est spécifiquement destiné à être utilisé avec Amazon EKS et vous permet de gérer vos ressources d'une manière native à Kubernetes.

Pour plus d'informations sur le contrôleur App Mesh, consultez la [documentation du contrôleur App Mesh](#).

## App Mesh et Terraform

[Terraform](#) est une infrastructure open source en tant qu'outil logiciel de code. Terraform peut gérer les services cloud à l'aide de sa CLI et interagir avec les API à l'aide de fichiers de configuration déclaratifs.

Pour en savoir plus sur l'utilisation d'App Mesh avec Terraform, consultez la documentation de [Terraform](#).

# Utilisation de maillages partagés

Vous pouvez partager vos maillages App Mesh entre différents AWS comptes à l'aide du AWS Resource Access Manager service. Un maillage partagé permet aux ressources créées par différents AWS comptes de communiquer entre elles dans le même maillage.

Un AWS compte peut être le propriétaire d'une ressource maillée, un consommateur de maillage, ou les deux. Les consommateurs peuvent créer des ressources dans un maillage partagé avec leur compte. Les propriétaires peuvent créer des ressources dans n'importe quel maillage que possède le compte. Le propriétaire d'un maillage peut partager un maillage avec les types de consommateurs de maillage suivants.

- AWS Comptes spécifiques à l'intérieur ou à l'extérieur de son organisation dans AWS Organizations
- Une unité organisationnelle au sein de son organisation dans AWS Organizations
- Toute son organisation en AWS Organizations

Pour une end-to-end présentation du partage d'un maillage, voir Présentation [du maillage entre comptes sur](#) GitHub.

## Octroi d'autorisations pour partager des maillages

Lorsque vous partagez des maillages entre comptes, des autorisations sont requises pour que le principal IAM partage le maillage et des autorisations au niveau des ressources pour le maillage lui-même.

### Octroi de l'autorisation de partager un maillage

Un ensemble minimal d'autorisations est requis pour qu'un directeur IAM puisse partager un maillage. Nous vous recommandons d'utiliser les politiques IAM `AWSResourceAccessManagerFullAccess` gérées `AWSAppMeshFullAccess` et les politiques IAM pour vous assurer que vos principaux IAM disposent des autorisations requises pour partager et utiliser des maillages partagés.

Si vous utilisez une politique IAM personnalisée, les `appmesh:DeleteMeshPolicy` actions `appmesh:PutMeshPolicy` et `appmesh:GetMeshPolicy`, et sont obligatoires. Il s'agit d'actions IAM avec autorisation uniquement. Si ces autorisations ne sont pas accordées à un principal IAM, une erreur se produira lors de la tentative de partage du maillage à l'aide du AWS RAM service.

Pour plus d'informations sur la manière dont le AWS Resource Access Manager service utilise IAM, consultez la section [Comment AWS RAM utilise IAM](#) dans le guide de l'AWS Resource Access Manager utilisateur.

## Octroi d'autorisations pour un maillage

Un maillage partagé possède les autorisations suivantes.

- Les consommateurs peuvent répertorier et décrire toutes les ressources d'un maillage partagé avec le compte.
- Les propriétaires peuvent répertorier et décrire toutes les ressources de tous les maillages que possède le compte.
- Les propriétaires et les consommateurs peuvent modifier les ressources d'un maillage créé par le compte, mais ils ne peuvent pas modifier les ressources créées par un autre compte.
- Les consommateurs peuvent supprimer n'importe quelle ressource d'un maillage créé par le compte.
- Les propriétaires peuvent supprimer n'importe quelle ressource d'un maillage créée par n'importe quel compte.
- Les ressources du propriétaire ne peuvent faire référence qu'à d'autres ressources du même compte. Par exemple, un nœud virtuel ne peut faire référence AWS Cloud Map qu'à un AWS Certificate Manager certificat qui se trouve dans le même compte que le propriétaire du nœud virtuel.
- Les propriétaires et les consommateurs peuvent connecter un proxy Envoy à App Mesh en tant que nœud virtuel détenu par le compte.
- Les propriétaires peuvent créer des passerelles virtuelles et des routes de passerelle virtuelles.
- Les propriétaires et les consommateurs peuvent répertorier les tags et baliser/détacher les ressources dans un maillage créé par le compte. Ils ne peuvent pas répertorier les tags et baliser/détacher les ressources dans un maillage qui n'est pas créé par le compte.

Les maillages partagés utilisent une autorisation basée sur des politiques. Un maillage est partagé avec un ensemble fixe d'autorisations. Ces autorisations sont sélectionnées pour être ajoutées à une politique de ressources, et une stratégie IAM facultative peut également être sélectionnée en fonction de l'utilisateur/du rôle IAM. L'intersection des autorisations autorisées dans ces politiques, moins les autorisations explicitement refusées, détermine l'accès du principal au maillage.

Lorsque vous partagez un maillage, le AWS Resource Access Manager service crée une politique gérée nommée `AWSRAMDefaultPermissionAppMesh` et l'associe à votre App Mesh qui fournit les autorisations suivantes.

- `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`

## Conditions préalables au partage de maillages

Pour partager un maillage, vous devez remplir les conditions préalables suivantes.



- Vous devez être propriétaire du maillage de votre AWS compte. Vous ne pouvez pas partager un maillage qui a été partagé avec vous.
- Pour partager un maillage avec votre organisation ou une unité organisationnelle AWS Organizations, vous devez activer le partage avec AWS Organizations. Pour plus d'informations, consultez [Activation du partage avec AWS Organizations](#) dans le Guide de l'utilisateur AWS RAM .
- Vos services doivent être déployés dans un Amazon VPC doté d'une connectivité partagée entre les comptes qui incluent les ressources maillées que vous souhaitez communiquer entre vous. L'un des moyens de partager la connectivité réseau consiste à déployer tous les services que vous souhaitez utiliser dans votre maillage sur un sous-réseau partagé. Pour plus d'informations et pour connaître les limites, consultez la section [Partage d'un sous-réseau](#).
- Les services doivent être détectables via DNS ou AWS Cloud Map. Pour plus d'informations sur la découverte de services, consultez la section [Nœuds virtuels](#).

## Services connexes

Le partage de maillage s'intègre à AWS Resource Access Manager (AWS RAM). AWS RAM est un service qui vous permet de partager vos AWS ressources avec n'importe quel AWS compte ou via AWS Organizations. Avec AWS RAM, vous partagez les ressources que vous possédez en créant un partage de ressources. Un partage de ressources spécifie les ressources à partager, ainsi que les consommateurs avec qui elles seront partagées. Les consommateurs peuvent être AWS des comptes individuels, des unités organisationnelles ou une organisation entière AWS Organizations.

Pour plus d'informations AWS RAM, consultez le [guide de AWS RAM l'utilisateur](#).

## Partage d'un maillage

Le partage d'un maillage permet aux ressources de maillage créées par différents comptes de communiquer entre elles dans le même maillage. Vous ne pouvez partager qu'un maillage dont vous êtes le propriétaire. Pour partager un maillage, vous devez l'ajouter à un partage de ressources. Un partage de ressources est une AWS RAM ressource qui vous permet de partager vos ressources entre différents AWS comptes. Un partage de ressources indique les ressources à partager et les consommateurs avec lesquels elles sont partagées. Lorsque vous partagez un maillage à l'aide de la console Amazon Linux, vous l'ajoutez à un partage de ressources existant. Pour ajouter le maillage à un nouveau partage de ressources, créez le partage de ressources à l'aide de la [AWS RAM console](#).

Si vous faites partie d'une organisation AWS Organizations et que le partage au sein de votre organisation est activé, les consommateurs de votre organisation peuvent accéder automatiquement

au maillage partagé. Dans le cas contraire, les consommateurs reçoivent une invitation à rejoindre le partage de ressources et ont accès au maillage partagé après avoir accepté l'invitation.

Vous pouvez partager un maillage dont vous êtes propriétaire à l'aide de la AWS RAM console ou du AWS CLI.

Pour partager un maillage dont vous êtes propriétaire à l'aide de la AWS RAM console

Pour obtenir des instructions, reportez-vous à [la section Création d'un partage de ressources](#) dans le guide de AWS RAM l'utilisateur. Lorsque vous sélectionnez un type de ressource, sélectionnez Maillages, puis sélectionnez le maillage que vous souhaitez partager. Si aucun maillage n'est répertorié, créez d'abord un maillage. Pour plus d'informations, consultez [Créer un mesh de services](#).

Pour partager un maillage dont vous êtes propriétaire à l'aide du AWS CLI

Utilisez la commande [create-resource-share](#). Pour l'`--resource-arn` option, spécifiez l'ARN du maillage que vous souhaitez partager.

## Annulation du partage d'un maillage partagé

Lorsque vous annulez le partage d'un maillage, App Mesh désactive l'accès ultérieur au maillage pour les anciens utilisateurs du maillage. Cependant, App Mesh ne supprime pas les ressources créées par les consommateurs. Une fois que le maillage n'est plus partagé, seul le propriétaire du maillage peut accéder aux ressources et les supprimer. App Mesh empêche le compte qui possédait les ressources du maillage de recevoir des informations de configuration une fois le maillage annulé. App Mesh empêche également tout autre compte dont les ressources se trouvent dans le maillage de recevoir des informations de configuration provenant d'un maillage non partagé. Seul le propriétaire du maillage peut annuler son partage.

Pour annuler le partage d'un maillage partagé dont vous êtes le propriétaire, vous devez le supprimer du partage de ressources. Vous pouvez le faire à l'aide de la AWS RAM console ou du AWS CLI.

Pour annuler le partage d'un maillage partagé dont vous êtes propriétaire à l'aide de la console AWS RAM

Pour obtenir des instructions, voir [Mettre à jour un partage de ressources](#) dans le guide de AWS RAM l'utilisateur.

Pour annuler le partage d'un maillage partagé dont vous êtes le propriétaire à l'aide du AWS CLI

Utilisez la commande [disassociate-resource-share](#).

## Identifier un maillage partagé

Les propriétaires et les consommateurs peuvent identifier les maillages partagés et les ressources de maillage à l'aide de la console Amazon Linux et AWS CLI

Pour identifier un maillage partagé à l'aide de la console Amazon Linux

1. Ouvrez la console App Mesh à l'[adresse https://console.aws.amazon.com/appmesh/](https://console.aws.amazon.com/appmesh/).
2. Dans le menu de navigation de gauche, sélectionnez Meshes. L'ID de compte du propriétaire du maillage pour chaque maillage est répertorié dans la colonne Propriétaire du maillage.
3. Dans le menu de navigation de gauche, sélectionnez Services virtuels, Routeurs virtuels ou Nœuds virtuels. Vous voyez l'ID de compte du propriétaire du maillage et du propriétaire de la ressource pour chacune des ressources.

Pour identifier un maillage partagé à l'aide du AWS CLI

Utilisez la `aws appmesh list resource` commande, telle que `aws appmesh list-meshes`. La commande renvoie les maillages que vous possédez et ceux qui sont partagés avec vous. La `meshOwner` propriété indique l'ID de AWS compte du propriétaire de la ressource `meshOwner` et la `resourceOwner` propriété indique l'ID de AWS compte du propriétaire de la ressource. Toute commande exécutée sur une ressource de maillage renvoie ces propriétés.

Les balises définies par l'utilisateur que vous attachez à un maillage partagé ne sont disponibles que pour votre Compte AWS. Ils ne sont pas disponibles pour les autres comptes avec lesquels le maillage est partagé. L'accès à la `aws appmesh list-tags-for-resource` commande pour un maillage dans un autre compte est refusé.

## Facturation et mesures

Le partage d'un maillage est gratuit.

## Quotas d'instances

Tous les quotas d'un maillage s'appliquent également aux maillages partagés, quelle que soit la personne qui a créé les ressources dans le maillage. Seul le propriétaire du maillage peut demander

des augmentations de quota. Pour plus d'informations, consultez [App Mesh](#). Le AWS Resource Access Manager service dispose également de quotas. Pour de plus amples informations, veuillez consulter [Quotas de service](#).

# AWSservices intégrés à App Mesh

App Mesh fonctionne avec d'autresAWS services pour fournir des solutions supplémentaires aux défis de votre entreprise. Cette rubrique identifie les services qui utilisent App Mesh pour ajouter des fonctionnalités, ou les services qu'App Mesh utilise pour effectuer des tâches.

## Table des matières

- [Création de ressources App Mesh avecAWS CloudFormation](#)
- [App Mesh surAWS Outposts](#)

## Création de ressources App Mesh avecAWS CloudFormation

App Mesh est intégré àAWS CloudFormation, un service qui vous aide à modéliser et à configurer vosAWS ressources afin que vous puissiez consacrer moins de temps à la création et à la gestion de vos ressources et de votre infrastructure. Vous créez un modèle qui décrit toutes lesAWS ressources que vous souhaitez, par exemple un maillage App Mesh, etAWS CloudFormation approvisionne et configure ces ressources automatiquement.

Lorsque vous utilisezAWS CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources App Mesh de manière cohérente et répétée. Il suffit de décrire vos ressources une fois, puis d'allouer les mêmes ressources à l'infini dans plusieurs comptes et régions AWS.

## App Mesh etAWS CloudFormation modèles

Pour provisionner et configurer des ressources pour App Mesh et les services associés, vous devez maîtriser les [AWS CloudFormationmodèles](#). Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez allouer dans vos piles AWS CloudFormation. Si JSON ou YAML ne vous est pas familier, vous pouvez utiliser AWS CloudFormation Designer pour vous aider à démarrer avec des modèles AWS CloudFormation. Pour plus d'informations, consultez [Qu'est-ce que AWS CloudFormation Designer ?](#) dans le AWS CloudFormationGuide de l'utilisateur.

App Mesh prend en charge la création de maillages, de routes, de nœuds virtuels, de routeurs virtuels et de services virtuels dansAWS CloudFormation. Pour plus d'informations, y compris des exemples de modèles JSON et YAML pour vos ressources App Mesh, consultez la [Référence du type de ressource de App Mesh application](#) dans le Guide deAWS CloudFormation l'utilisateur.

## En savoir plus sur AWS CloudFormation

Pour en savoir plus sur AWS CloudFormation, consultez les ressources suivantes :

- [AWS CloudFormation](#)
- [Guide de l'utilisateur AWS CloudFormation](#)
- [Guide de l'utilisateur de l'interface de ligne de commande AWS CloudFormation](#)

## App Mesh surAWS Outposts

AWSOutposts active lesAWS services, l'infrastructure et les modèles d'exploitation natifs dans les installations sur site. Dans les environnementsAWS Outposts, vous pouvez utiliser les mêmesAWS API, outils et infrastructure que dans leAWS cloud. App Mesh surAWS Outposts est idéal pour les charges de travail à faible latence qui ont besoin d'être exécutées à proximité des données et des applications sur site. Pour plus d'informations surAWS Outposts, consultez le [Guide de l'utilisateur d'AWSOutposts](#).

### Prérequis

Les conditions préalables à l'utilisation d'App Mesh surAWS Outposts sont les suivantes :

- Vous devez avoir installé et configuré un Outpost dans votre centre de données sur site.
- Vous devez avoir une connexion réseau fiable entre votre Outpost et sa région AWS.
- LaAWS région pour l'Outpost doit apporter son soutienAWS App Mesh. Pour obtenir la liste des régions prises en charge, consultez la section [AWS App MeshPoints de terminaison et quotas](#) dans le Références générales AWS.

### Limites

Les limites de l'utilisation d'App Mesh surAWS Outposts sont les suivantes :

- AWS Identity and Access Management, Application Load Balancer, le Network Load Balancer, le Classic Load Balancer et Amazon Route 53 s'exécutent dans laAWS région, et non sur les Outposts. Cela a pour effet d'accroître les latences entre ces services et les conteneurs.

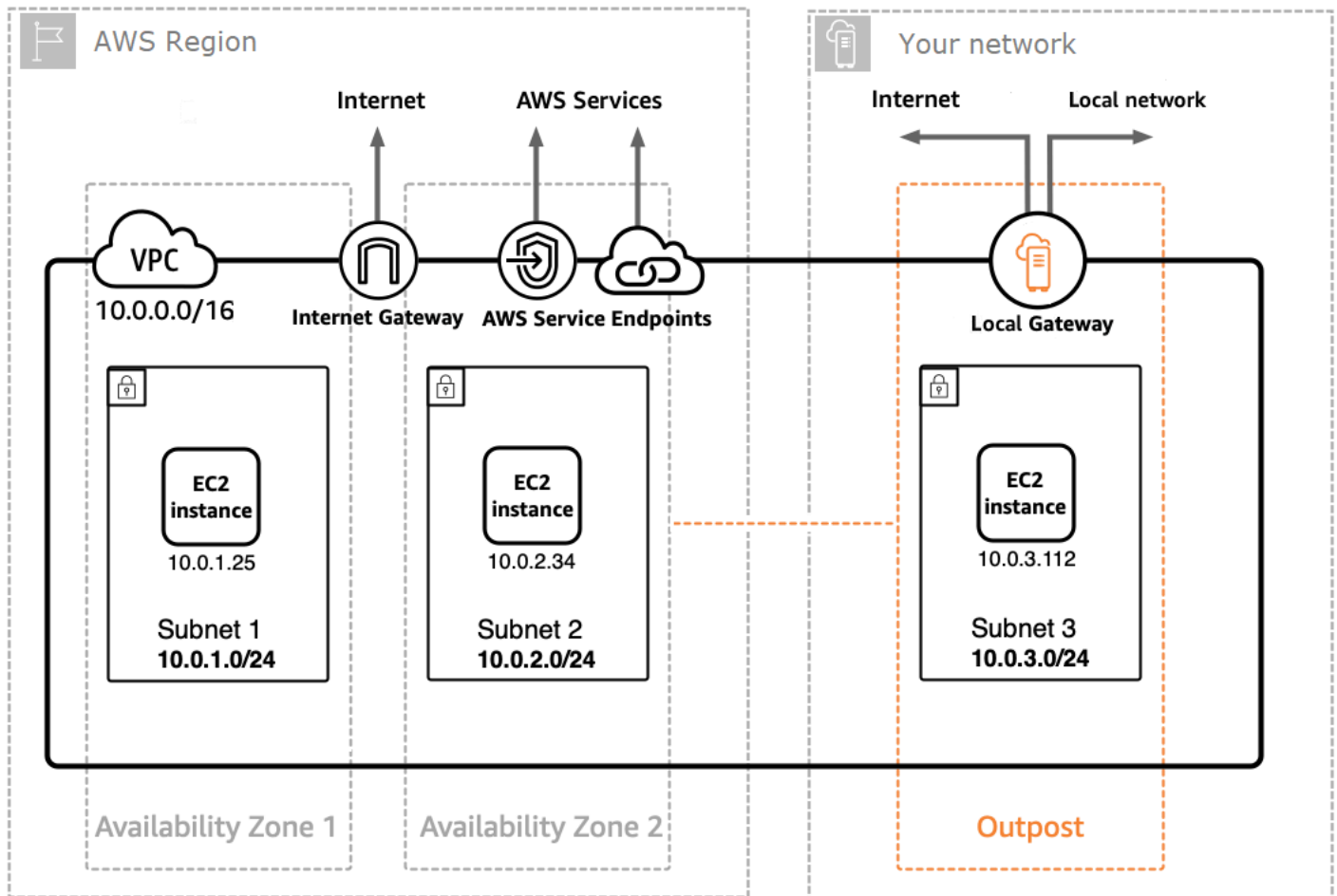
## Considérations relatives à la connectivité réseau

Voici les considérations relatives à la connectivité réseau pour Amazon EKS AWS Outposts :

- Si la connectivité réseau entre votre Outpost et sa AWS région est perdue, les proxys App Mesh Envoy continuent de s'exécuter. Vous ne serez toutefois pas en mesure de modifier votre maillage de service tant que la connectivité n'est pas rétablie.
- Nous vous recommandons de fournir une connectivité fiable, hautement disponible et à faible latence entre votre Outpost et sa AWS région.

## Création d'un proxy App Mesh Envoy sur un Outpost

Un Outpost est une extension d'une AWS région, et vous pouvez étendre un VPC Amazon dans un compte pour couvrir plusieurs zones de disponibilité et tous les associés. Lorsque vous configurez votre Outpost, vous lui associez un groupe de sous-réseaux pour étendre votre environnement VPC régional à votre installation sur site. Les instances d'un Outpost apparaissent dans le cadre de votre VPC régional, similaire à une zone de disponibilité avec des sous-réseaux associés.



Pour créer un proxy App Mesh Envoy sur un Outpost, ajoutez l'image du conteneur App Mesh Envoy à la tâche Amazon ECS ou au pod Amazon EKS exécuté sur un Outpost. Pour plus d'informations, consultez [Amazon Elastic Container Service on AWS Outposts](#) dans le Amazon Elastic Container Service Developer Guide et [Amazon Elastic Kubernetes Service on AWS Outposts](#) dans le Guide de l'utilisateur Amazon EKS.



# Meilleures pratiques en matière d'App Mesh

Pour atteindre l'objectif de zéro demande ayant échoué lors des déploiements planifiés et lors de la perte imprévue de certains hôtes, les meilleures pratiques décrites dans cette rubrique mettent en œuvre la stratégie suivante :

- Augmentez les chances de réussite d'une demande du point de vue de l'application en utilisant une stratégie de nouvelle tentative sûre par défaut. Pour plus d'informations, consultez [Instrumenter tous les itinéraires avec de nouvelles tentatives](#).
- Augmentez les chances de réussite d'une nouvelle tentative en maximisant la probabilité que la nouvelle demande soit envoyée à une destination réelle. Pour plus d'informations, consultez [Ajuster la vitesse de déploiement](#), [Élargir avant d'intégrer](#) et [Mettre en œuvre des contrôles de santé des conteneurs](#).

Pour réduire ou éliminer de manière significative les défaillances, nous vous recommandons de mettre en œuvre les recommandations dans toutes les pratiques suivantes.

## Instrumenter tous les itinéraires avec de nouvelles tentatives

Configurez tous les services virtuels pour utiliser un routeur virtuel et définissez une politique de nouvelle tentative par défaut pour tous les itinéraires. Cela limitera les demandes échouées en résélectionnant un hôte et en envoyant une nouvelle demande. Pour les politiques de nouvelle tentative, nous recommandons une valeur d'au moins deux pour `maxRetries` et de spécifier les options suivantes pour chaque type d'événement de nouvelle tentative dans chaque type de route qui prend en charge le type d'événement de nouvelle tentative :

- TCP — `connection-error`
- HTTP et HTTP/2 — `stream-error` et `gateway-error`
- gRPC — et `cancelled unavailable`

Les autres événements de nouvelle tentative doivent être pris en compte dans case-by-case la mesure où ils peuvent ne pas être sûrs, par exemple si la demande n'est pas idempotente. Vous devrez prendre en compte et tester les valeurs correspondant au compromis approprié entre la latence maximale d'une demande ( $\text{maxRetries} * \text{perRetryTimeout}$ ) `maxRetries` et `perRetryTimeout` le taux de réussite accru d'un plus grand nombre de tentatives. De plus,

lorsqu'Envoy tente de se connecter à un point de terminaison qui n'est plus présent, vous devez vous attendre à ce que cette demande soit entièrement consommée par `RetryTimeout`. Pour configurer une politique de nouvelle tentative, consultez [Création d'une route](#) puis sélectionnez le protocole que vous souhaitez router.

### Note

Si vous avez implémenté un itinéraire le 29 juillet 2020 ou après cette date et que vous n'avez pas spécifié de politique de nouvelles tentatives, App Mesh a peut-être automatiquement créé une politique de relance par défaut similaire à la politique précédente pour chaque itinéraire que vous avez créé le 29 juillet 2020 ou après cette date. Pour plus d'informations, consultez [Politique de nouvelle tentative d'itinéraire par défaut](#).

## Ajuster la vitesse de déploiement

Lorsque vous utilisez des déploiements progressifs, réduisez la vitesse globale de déploiement. Par défaut, Amazon ECS configure une stratégie de déploiement comprenant au moins 100 % de tâches saines et 200 % de tâches au total. Lors du déploiement, cela se traduit par deux points de dérive élevée :

- La taille totale du parc de nouvelles tâches peut être visible par les envoyés avant qu'ils ne soient prêts à traiter les demandes (voir [Mettre en œuvre des contrôles de santé des conteneurs](#) pour les mesures d'atténuation).
- La taille totale du parc des anciennes tâches peut être visible par les envoyés lorsque les tâches sont terminées.

Lorsqu'ils sont configurés avec ces contraintes de déploiement, les orchestrateurs de conteneurs peuvent entrer dans un état dans lequel ils masquent simultanément toutes les anciennes destinations et rendent toutes les nouvelles destinations visibles. Comme votre plan de données Envoy est finalement cohérent, cela peut entraîner des périodes pendant lesquelles l'ensemble des destinations visibles dans votre plan de données diverge du point de vue de l'orchestrateur. Pour pallier ce problème, nous recommandons de maintenir un minimum de 100 % de tâches saines, mais de réduire le nombre total de tâches à 125 %. Cela réduira les divergences et améliorera la fiabilité des nouvelles tentatives. Nous recommandons les paramètres suivants pour les différents temps d'exécution des conteneurs :

## Amazon ECS

Si le nombre souhaité de deux ou trois pour votre service est `maximumPercent` de 150 %. Sinon, `maximumPercent` fixez-le à 125 %.

## Kubernetes

Configurez ceux de votre déploiement `update strategy`, en `maxUnavailable` les réglant sur 0 % et `maxSurge` sur 25 %. [Pour plus d'informations sur les déploiements, consultez la documentation relative aux déploiements Kubernetes.](#)

## Élargir avant d'intégrer

La mise à l'échelle externe et la mise à l'échelle interne peuvent toutes deux entraîner une certaine probabilité d'échec des demandes lors de nouvelles tentatives. Bien que certaines recommandations relatives aux tâches limitent la mise à l'échelle, la seule recommandation en matière d'extension est de minimiser le pourcentage de tâches redimensionnées à un moment donné. Nous vous recommandons d'utiliser une stratégie de déploiement qui intègre les nouvelles tâches Amazon ECS ou les déploiements Kubernetes avant d'intégrer les anciennes tâches ou déploiements. Cette stratégie de mise à l'échelle permet de réduire le pourcentage de tâches ou de déploiements échelonnés, tout en maintenant la même rapidité. Cette pratique s'applique à la fois aux tâches Amazon ECS et aux déploiements Kubernetes.

## Mettre en œuvre des contrôles de santé des conteneurs

Dans le scénario de mise à l'échelle, les conteneurs d'une tâche Amazon ECS peuvent être hors service et ne pas répondre initialement. Nous recommandons les suggestions suivantes pour les différents temps d'exécution des conteneurs :

### Amazon ECS

Pour atténuer ce problème, nous recommandons d'effectuer des vérifications de l'état des conteneurs et de classer les dépendances des conteneurs afin de garantir qu'Envoy fonctionne et fonctionne correctement avant le démarrage de tout conteneur nécessitant une connectivité réseau sortante. Pour configurer correctement un conteneur d'applications et un conteneur Envoy dans une définition de tâche, voir [Dépendance du conteneur](#).

### Kubernetes

[Aucune, car les sondes de réactivité et de disponibilité de Kubernetes ne sont pas prises en compte lors de l'enregistrement et de la désinscription des instances dans AWS Cloud Map le contrôleur App Mesh pour Kubernetes.](#) Pour plus d'informations, consultez le GitHub numéro [#132](#).

## Optimisation de la résolution DNS

Si vous utilisez le DNS pour la découverte de services, il est essentiel de sélectionner le protocole IP approprié pour optimiser la résolution DNS lors de la configuration de vos maillages. App Mesh prend en charge les deux IPv6, IPv4 et votre choix peut avoir un impact sur les performances et la compatibilité de votre service. Si votre infrastructure n'est pas compatible IPv6, nous vous recommandons de spécifier un paramètre IP adapté à votre infrastructure plutôt que de vous fier au IPv6\_PREFERRED comportement par défaut. Le IPv6\_PREFERRED comportement par défaut peut dégrader les performances du service.

- `IPv6_Preferred` — Il s'agit du paramètre par défaut. Envoy effectue d'abord une recherche DNS pour les adresses IPv6 et revient à zéro IPv4 si aucune IPv6 adresse n'est trouvée. Cela est avantageux si votre infrastructure prend principalement en charge IPv6 mais doit être IPv4 compatible.
- `IPv4_Preferred` — Envoy recherche d'abord IPv4 des adresses et y revient IPv6 si aucune IPv4 adresse n'est disponible. Utilisez ce paramètre si votre infrastructure prend principalement en charge IPv4 mais présente une certaine IPv6 compatibilité.
- `IPv6_Only` — Choisissez cette option si vos services prennent exclusivement en charge le trafic. IPv6 Envoy effectue uniquement des recherches DNS pour les IPv6 adresses, s'assurant ainsi que tout le trafic est acheminé. IPv6
- `IPv4_Only` — Choisissez ce paramètre si vos services prennent exclusivement en charge le trafic. IPv4 Envoy effectue uniquement des recherches DNS pour les IPv4 adresses, s'assurant ainsi que tout le trafic est acheminé. IPv4

Vous pouvez définir les préférences de version IP à la fois au niveau du maillage et au niveau du nœud virtuel, les paramètres du nœud virtuel remplaçant ceux du niveau du maillage.

Pour plus d'informations, consultez [Service Meshes](#) et [nœuds virtuels](#).

# Sécurité dans AWS App Mesh

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le [modèle de responsabilité partagée](#) décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des [programmes de conformité AWS](#). Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS App Mesh, veuillez consulter [AWS Services in Scope by Compliance Program](#) (français non garanti). App Mesh est chargé de fournir en toute sécurité la configuration aux proxys locaux, y compris les secrets tels que les clés privées des certificats TLS.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, notamment :
  - La sensibilité de vos données, les exigences de votre entreprise et les lois et réglementations applicables.
  - La configuration de sécurité du plan de données App Mesh, y compris la configuration des groupes de sécurité qui permettent au trafic de passer entre les services au sein de votre VPC.
  - Configuration de vos ressources de calcul associées à App Mesh.
  - Les politiques IAM associées à vos ressources de calcul et la configuration qu'elles sont autorisées à récupérer depuis le plan de contrôle App Mesh.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'App Mesh. Les rubriques suivantes expliquent comment configurer App Mesh pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources App Mesh.

## Principe de sécurité App Mesh

Les clients doivent être en mesure de régler la sécurité selon leurs besoins. La plate-forme ne doit pas les empêcher d'être plus sécurisés. Les fonctionnalités de la plateforme sont sécurisées par défaut.

## Rubriques

- [protocole TLS \(Transport Layer Security\)](#)
- [Authentification TLS mutuelle](#)
- [Comment AWS App Mesh fonctionne avec IAM](#)
- [Journalisation des appels AWS App Mesh d'API à l'aide AWS CloudTrail](#)
- [Protection des données dans AWS App Mesh](#)
- [Validation de conformité pour AWS App Mesh](#)
- [Sécurité de l'infrastructure dans AWS App Mesh](#)
- [Résilience dans AWS App Mesh](#)
- [Analyse de configuration et de vulnérabilité dans AWS App Mesh](#)

## protocole TLS (Transport Layer Security)

Dans App Mesh, Transport Layer Security (TLS) chiffre les communications entre les proxys Envoy déployés sur les ressources informatiques représentées dans App Mesh par des points de terminaison du maillage, tels que et. [Nœuds virtuels](#) [Passerelle](#) Le proxy négocie et met fin au protocole TLS. Lorsque le proxy est déployé avec une application, le code de votre application n'est pas responsable de la négociation d'une session TLS. Le proxy négocie le protocole TLS au nom de votre application.

App Mesh vous permet de fournir le certificat TLS au proxy de la manière suivante :

- Un certificat privé de AWS Certificate Manager (ACM) émis par un AWS Private Certificate Authority (AWS Private CA).
- Certificat stocké sur le système de fichiers local d'un nœud virtuel émis par votre propre autorité de certification (CA)
- Certificat fourni par un point de terminaison SDS (Secrets Discovery Service) via un socket de domaine Unix local.

[Automatisation du proxy Envoy](#) doit être activé pour le proxy Envoy déployé représenté par un point de terminaison maillé. Lorsque vous activez l'autorisation du proxy, nous vous recommandons de restreindre l'accès uniquement au point de terminaison maillé pour lequel vous activez le chiffrement.

## Exigences du certificat

L'un des noms alternatifs de sujet (SAN) figurant sur le certificat doit répondre à des critères spécifiques, en fonction de la manière dont le service réel représenté par un point de terminaison de maillage est découvert.

- DNS — L'un des réseaux SAN de certificats doit correspondre à la valeur spécifiée dans les paramètres de découverte du service DNS. Pour une application portant le nom Service Discovery `mesh-endpoint.apps.local`, vous pouvez créer un certificat correspondant à ce nom ou un certificat avec le caractère générique `*.apps.local`.
- AWS Cloud Map— L'un des SAN de certificats doit correspondre à la valeur fournie dans les paramètres de découverte des AWS Cloud Map services à l'aide du format `service-name.namespace-name`. Pour une application dont les paramètres de découverte de AWS Cloud Map service sont ServiceName `mesh-endpoint` et NamespaceName `apps.local`, vous pouvez créer un certificat correspondant au nom `mesh-endpoint.apps.local` ou un certificat avec le caractère générique `*.apps.local`.

Pour les deux mécanismes de découverte, si aucun des SAN de certificats ne correspond aux paramètres de découverte du service DNS, la connexion entre Envoy échoue avec le message d'erreur suivant, tel qu'il est indiqué par le client Envoy.

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

## Certificats d'authentification TLS

App Mesh prend en charge plusieurs sources de certificats lors de l'utilisation de l'authentification TLS.


### AWS Private CA

Le certificat doit être stocké dans ACM dans la même région et dans le même AWS compte que le point de terminaison du maillage qui utilisera le certificat. Le certificat de l'autorité de certification ne doit pas nécessairement se trouver dans le même AWS compte, mais il doit tout de même se trouver dans la même région que le point de terminaison du maillage. Si vous n'en avez pas Autorité de certification privée AWS, vous devez en [créer un](#) avant de pouvoir lui demander un certificat. Pour plus d'informations sur la demande d'un certificat auprès d'un utilisateur AWS Private CA ACM existant, consultez la section [Demander un certificat privé](#). Le certificat ne peut pas être un certificat public.

Les autorités de certification privées que vous utilisez pour les politiques du client TLS doivent être des autorités de certification utilisateur root.


Pour configurer un nœud virtuel avec des certificats et des autorités de certification AWS Private CA, le principal (tel qu'un utilisateur ou un rôle) que vous utilisez pour appeler App Mesh doit disposer des autorisations IAM suivantes :

- Pour tous les certificats que vous ajoutez à la configuration TLS d'un écouteur, le principal doit disposer de `acm:DescribeCertificate` autorisation.
- Pour toutes les autorités de certification configurées selon une politique client TLS, le principal doit avoir `acm-pca:DescribeCertificateAuthority` autorisation.

 Important

Le partage d'une autorité de certification avec d'autres comptes peut conférer à ces comptes des privilèges involontaires à l'autorité de certification. Nous recommandons d'utiliser des politiques basées sur les ressources afin de restreindre l'accès aux seuls `acm-pca:DescribeCertificateAuthority` comptes qui n'ont pas besoin d'émettre de certificats auprès de l'autorité de certification. `acm-pca:GetCertificateAuthorityCertificate`

Vous pouvez ajouter ces autorisations à une stratégie IAM existante attachée à un principal ou créer un nouveau principal et une nouvelle stratégie et associer la stratégie au principal. Pour plus d'informations, consultez les sections [Modification des politiques IAM](#), [Création de politiques IAM](#) et [Ajout d'autorisations d'identité IAM](#).

 Note

Vous payez des frais mensuels pour le fonctionnement de chacune d'elles AWS Private CA jusqu'à ce que vous les supprimiez. Vous payez également pour les certificats privés que vous émettez chaque mois et pour les certificats privés que vous exportez. Pour plus d'informations, consultez [Tarification d'AWS Certificate Manager](#).

Lorsque vous activez l'[autorisation de proxy](#) pour le proxy Envoy représenté par un point de terminaison maillé, les autorisations IAM suivantes doivent être attribuées au rôle IAM que vous utilisez :



- Pour tous les certificats configurés sur l'écouteur d'un nœud virtuel, le rôle doit disposer de `acm:ExportCertificateautorisation`.
- Pour toutes les autorités de certification configurées selon une politique client TLS, le rôle doit disposer de `acm-pca:GetCertificateAuthorityCertificateautorisation`.

## Système de fichiers

Vous pouvez distribuer des certificats à Envoy à l'aide du système de fichiers. Vous pouvez le faire en rendant la chaîne de certificats et la clé privée correspondante disponibles sur le chemin du fichier. De cette façon, ces ressources sont accessibles depuis le proxy du sidecar Envoy.

## Service de découverte secrète (SDS) d'Envoy

Envoy récupère des secrets tels que des certificats TLS à partir d'un point de terminaison spécifique via le protocole Secrets Discovery. Pour plus d'informations sur ce protocole, consultez la [documentation SDS](#) d'Envoy.

App Mesh configure le proxy Envoy pour qu'il utilise un socket de domaine Unix local au proxy pour servir de point de terminaison du Secret Discovery Service (SDS) lorsque le SDS sert de source pour vos certificats et chaînes de certificats. Vous pouvez configurer le chemin d'accès à ce point de terminaison à l'aide de la variable d'`APPMESH_SDS_SOCKET_PATH` environnement.

### Important

Le service Local Secrets Discovery utilisant un socket de domaine Unix est pris en charge sur les versions 1.15.1.0 et ultérieures du proxy App Mesh Envoy. App Mesh prend en charge le protocole V2 SDS à l'aide de gRPC.

## Intégration à l'environnement d'exécution SPIFFE (SPIRE)

Vous pouvez utiliser n'importe quelle implémentation parallèle de l'API SDS, y compris les chaînes d'outils existantes telles que [SPIFFE Runtime Environment \(SPIRE\)](#). SPIRE est conçu pour permettre le déploiement de l'authentification TLS mutuelle entre plusieurs charges de travail dans des systèmes distribués. Il atteste l'identité des charges de travail au moment de l'exécution. SPIRE fournit également des clés et des certificats spécifiques aux charges de travail, de courte durée et en rotation automatique directement vers les charges de travail.

Vous devez configurer l'agent SPIRE en tant que fournisseur SDS pour Envoy. Permettez-lui de fournir directement à Envoy le matériel clé dont il a besoin pour fournir une authentification TLS mutuelle. Exécutez les agents SPIRE dans des sidecars situés à côté des proxys Envoy. L'agent

se charge de régénérer les clés et les certificats éphémères selon les besoins. L'agent atteste Envoy et détermine les identités de service et les certificats CA qu'il doit mettre à la disposition d'Envoy lorsqu'Envoy se connecte au serveur SDS exposé par l'agent SPIRE.

Au cours de ce processus, les identités de service et les certificats CA font l'objet d'une rotation, et les mises à jour sont renvoyées à Envoy. Envoy les applique immédiatement aux nouvelles connexions, sans interruption ni interruption et sans que les clés privées ne touchent le système de fichiers.

## Comment App Mesh configure Envoys pour négocier le TLS

App Mesh utilise la configuration des points de terminaison du maillage du client et du serveur pour déterminer comment configurer la communication entre les envoyés dans un maillage.

### Avec les politiques du client

Lorsqu'une politique client impose l'utilisation du protocole TLS et que l'un des ports de la politique client correspond au port de la stratégie du serveur, la stratégie client est utilisée pour configurer le contexte de validation TLS du client. Par exemple, si la politique client d'une passerelle virtuelle correspond à la politique de serveur d'un nœud virtuel, une négociation TLS sera tentée entre les proxys en utilisant les paramètres définis dans la politique client de la passerelle virtuelle. Si la politique du client ne correspond pas au port de la politique du serveur, le protocole TLS entre les proxys peut être négocié ou non, en fonction des paramètres TLS de la politique du serveur.

### Sans politiques relatives aux clients

Si le client n'a pas configuré de politique client ou si la politique client ne correspond pas au port du serveur, App Mesh utilisera le serveur pour déterminer s'il convient ou non de négocier le protocole TLS avec le client, et comment. Par exemple, si une passerelle virtuelle n'a pas spécifié de politique client et qu'un nœud virtuel n'a pas configuré la terminaison TLS, le protocole TLS ne sera pas négocié entre les proxys. Si aucun client n'a spécifié de politique client correspondante et qu'un serveur a été configuré avec les modes STRICT TLSPERMISSIVE, les proxys seront configurés pour négocier le protocole TLS. En fonction de la manière dont les certificats ont été fournis pour la terminaison du protocole TLS, le comportement supplémentaire suivant s'applique.

- Certificats TLS gérés par ACM — Lorsqu'un serveur a configuré la terminaison TLS à l'aide d'un certificat géré par ACM, App Mesh configure automatiquement les clients pour négocier le protocole TLS et valider le certificat auprès de l'autorité de certification de l'utilisateur racine à laquelle le certificat est lié.

- Certificats TLS basés sur des fichiers : lorsqu'un serveur a configuré la terminaison TLS à l'aide d'un certificat issu du système de fichiers local du proxy, App Mesh configure automatiquement un client pour négocier le protocole TLS, mais le certificat du serveur n'est pas validé.

## Noms alternatifs du sujet

Vous pouvez éventuellement spécifier une liste de noms alternatifs de sujets (SAN) auxquels vous pouvez faire confiance. Les SAN doivent être au format FQDN ou URI. Si des SAN sont fournis, Envoy vérifie que le nom alternatif du sujet du certificat présenté correspond à l'un des noms de cette liste.

Si vous ne spécifiez pas de SAN sur le point de terminaison du maillage terminal, le proxy Envoy pour ce nœud ne vérifie pas le SAN sur un certificat client homologué. Si vous ne spécifiez pas de SAN sur le point de terminaison du maillage d'origine, le SAN du certificat fourni par le point de terminaison terminal doit correspondre à la configuration de découverte de service du point de terminaison du maillage.

Pour plus d'informations, consultez App Mesh [TLS : Exigences relatives aux certificats](#).

### Important

Vous ne pouvez utiliser des SAN à caractères génériques que si la politique client pour TLS est définie sur `not enforced`. Si la politique client du nœud virtuel ou de la passerelle virtuelle du client est configurée pour appliquer le protocole TLS, il ne peut pas accepter un SAN générique.

## Vérifier le chiffrement

Une fois que vous avez activé le protocole TLS, vous pouvez interroger le proxy Envoy pour confirmer que les communications sont cryptées. Le proxy Envoy émet des statistiques sur les ressources qui peuvent vous aider à déterminer si votre communication TLS fonctionne correctement. Par exemple, le proxy Envoy enregistre des statistiques sur le nombre de handshakes TLS réussis qu'il a négociés pour un point de terminaison de maillage spécifié. Déterminez le nombre de handshakes TLS réussis pour un point de terminaison de maillage nommé à `'my-mesh-endpoint'` aide de la commande suivante.

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep ssl.handshake
```

Dans l'exemple de sortie renvoyé suivant, il y a eu trois poignées de main pour le point de terminaison du maillage. La communication est donc cryptée.

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

Le proxy Envoy émet également des statistiques lorsque la négociation TLS échoue. Déterminez s'il y a eu des erreurs TLS pour le point de terminaison du maillage.

```
curl -s 'http://my-mesh-endpoint.apps.local:9901/stats' | grep -e "ssl.*\((fail|error\n)"
```

Dans l'exemple renvoyé, aucune erreur n'a été détectée pour plusieurs statistiques. La négociation TLS a donc réussi.

```
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.fail_verify_san: 0
```

Pour plus d'informations sur les statistiques Envoy TLS, consultez [Envoy Listener Statistics](#).

## Renouvellement des certificats

### AWS Private CA

Lorsque vous renouvelez un certificat auprès d'ACM, le certificat renouvelé est automatiquement distribué à vos proxys connectés dans les 35 minutes suivant la fin du renouvellement. Nous vous recommandons d'utiliser le renouvellement géré pour renouveler automatiquement les certificats à l'approche de la fin de leur période de validité. Pour plus d'informations, consultez la section [Gestion du renouvellement des certificats émis par Amazon par ACM](#) dans le guide de l' AWS Certificate Manager utilisateur.

### Votre propre certificat

Lorsque vous utilisez un certificat issu du système de fichiers local, Envoy ne le recharge pas automatiquement lorsqu'il est modifié. Vous pouvez redémarrer ou redéployer le processus Envoy pour charger un nouveau certificat. Vous pouvez également placer un nouveau certificat sur un chemin de fichier différent et mettre à jour la configuration du nœud virtuel ou de la passerelle avec ce chemin de fichier.

## Configurer les charges de travail Amazon ECS pour utiliser l'authentification TLS avec AWS App Mesh

Vous pouvez configurer votre maillage pour utiliser l'authentification TLS. Assurez-vous que les certificats sont disponibles pour les sidecars proxy Envoy que vous ajoutez à vos charges de travail. Vous pouvez associer un volume EBS ou EFS à votre sidecar Envoy, ou vous pouvez stocker et récupérer des certificats depuis Secrets Manager AWS .

- Si vous utilisez la distribution de certificats basée sur des fichiers, attachez un volume EBS ou EFS à votre sidecar Envoy. Assurez-vous que le chemin d'accès au certificat et à la clé privée correspond à celui configuré dans AWS App Mesh.
- Si vous utilisez une distribution basée sur le SDS, ajoutez un sidecar qui implémente l'API SDS d'Envoy avec accès au certificat.

### Note

SPIRE n'est pas pris en charge sur Amazon ECS.

## Configurer les charges de travail Kubernetes pour utiliser l'authentification TLS avec AWS App Mesh

Vous pouvez configurer le AWS App Mesh Controller for Kubernetes pour activer l'authentification TLS pour les backends et les écouteurs des services de nœuds virtuels et de passerelle virtuelle. Assurez-vous que les certificats sont disponibles pour les sidecars proxy Envoy que vous ajoutez à vos charges de travail. Vous trouverez un exemple pour chaque type de distribution dans la section de [présentation de](#) l'authentification TLS mutuelle.

- Si vous utilisez la distribution de certificats basée sur des fichiers, attachez un volume EBS ou EFS à votre sidecar Envoy. Assurez-vous que le chemin d'accès au certificat et à la clé privée correspond à celui configuré dans le contrôleur. Vous pouvez également utiliser un secret Kubernetes monté sur le système de fichiers.
- Si vous utilisez une distribution basée sur SDS, vous devez configurer un fournisseur SDS local de nœuds qui implémente l'API SDS d'Envoy. Envoy l'atteindra via USD. Pour activer la prise en charge des MTLs basés sur le SDS dans le AppMesh contrôleur EKS, définissez l'`enable-sdsindicator` sur `true` et fournissez le chemin UDS du fournisseur SDS local au contrôleur via

l'indicateur. `sds-uds-path` Si vous utilisez helm, vous devez les configurer dans le cadre de l'installation de votre contrôleur :

```
--set sds.enabled=true
```

#### Note

Vous ne pourrez pas utiliser SPIRE pour distribuer vos certificats si vous utilisez Amazon Elastic Kubernetes Service (Amazon EKS) en mode Fargate.

## Authentification TLS mutuelle

L'authentification mutuelle TLS (Transport Layer Security) est un composant facultatif du protocole TLS qui offre une authentification bidirectionnelle entre pairs. L'authentification TLS mutuelle ajoute une couche de sécurité par rapport au protocole TLS et permet à vos services de vérifier le client qui établit la connexion.

Dans la relation client-serveur, le client fournit également un certificat X.509 pendant le processus de négociation de session. Le serveur utilise ce certificat pour identifier et authentifier le client. Ce processus permet de vérifier si le certificat est émis par une autorité de certification (CA) fiable et s'il s'agit d'un certificat valide. Il utilise également le nom alternatif du sujet (SAN) sur le certificat pour identifier le client.

Vous pouvez activer l'authentification TLS mutuelle pour tous les protocoles pris en charge par AWS App Mesh. Il s'agit de TCP, HTTP/1.1, HTTP/2, gRPC.

#### Note

À l'aide d'App Mesh, vous pouvez configurer l'authentification TLS mutuelle pour les communications entre les proxys Envoy depuis vos services. Cependant, les communications entre vos applications et les proxys Envoy ne sont pas cryptées.

## Certificats d'authentification TLS mutuels

AWS App Mesh prend en charge deux sources de certificats possibles pour l'authentification TLS mutuelle. Les certificats client dans une politique client TLS et la validation du serveur dans une configuration TLS d'écouteur peuvent provenir de :

- **Système de fichiers** : certificats provenant du système de fichiers local du proxy Envoy en cours d'exécution. Pour distribuer des certificats à Envoy, vous devez fournir des chemins de fichiers pour la chaîne de certificats et une clé privée vers l'API App Mesh.
- **Service de découverte secrète (SDS) d'Envoy** : des ring-your-own sidecars B qui implémentent le SDS et autorisent l'envoi de certificats à Envoy. Ils incluent l'environnement d'exécution SPIFFE (SPIRE).

### Important

App Mesh ne stocke pas les certificats ou les clés privées utilisés pour l'authentification TLS mutuelle. Au lieu de cela, Envoy les stocke en mémoire.

## Configurer les points de terminaison du maillage

Configurez l'authentification TLS mutuelle pour vos points de terminaison maillés, tels que les nœuds virtuels ou les passerelles. Ces points de terminaison fournissent des certificats et spécifient des autorités fiables.

Pour ce faire, vous devez fournir des certificats X.509 à la fois pour le client et pour le serveur, et définir explicitement les certificats d'autorité de confiance dans le contexte de la validation de la terminaison et de l'origine du protocole TLS.

### La confiance à l'intérieur d'un maillage

Les certificats côté serveur sont configurés dans les écouteurs Virtual Node (terminaison TLS), et les certificats côté client sont configurés dans les backends du service Virtual Nodes (origine TLS). Comme alternative à cette configuration, vous pouvez définir une politique client par défaut pour tous les backends de services d'un nœud virtuel, puis, si nécessaire, vous pouvez remplacer cette politique pour des backends spécifiques selon les besoins. Les passerelles virtuelles ne peuvent être configurées qu'avec une politique client par défaut qui s'applique à tous ses backends.

Vous pouvez configurer la confiance entre différents maillages en activant l'authentification TLS mutuelle pour le trafic entrant sur les passerelles virtuelles pour les deux maillages.

### Confiance en dehors d'un maillage

Spécifiez les certificats côté serveur dans l'écouteur Virtual Gateway pour la terminaison TLS. Configurez le service externe qui communique avec votre passerelle virtuelle pour présenter les certificats côté client. Les certificats doivent être dérivés de l'une des mêmes autorités de certification (CA) que les certificats côté serveur utilisent sur l'écouteur Virtual Gateway pour la création du protocole TLS.

## Migrer les services vers l'authentification TLS mutuelle

Suivez ces directives pour maintenir la connectivité lors de la migration de vos services existants dans App Mesh vers l'authentification TLS mutuelle.

### Migration des services communiquant en texte brut

1. Activer le PERMISSIVE mode pour la configuration TLS sur le point de terminaison du serveur. Ce mode permet au trafic en texte brut de se connecter au point de terminaison.
2. Configurez l'authentification TLS mutuelle sur votre serveur, en spécifiant le certificat du serveur, la chaîne de confiance et éventuellement les SAN fiables.
3. Vérifiez que la communication s'effectue via une connexion TLS.
4. Configurez l'authentification TLS mutuelle sur vos clients, en spécifiant le certificat client, la chaîne de confiance et éventuellement les SAN fiables.
5. STRICTMode d'activation pour la configuration TLS sur le serveur.

### Migration des services communiquant via TLS

1. Configurez les paramètres TLS mutuels sur vos clients, en spécifiant le certificat client et éventuellement les SAN fiables. Le certificat client n'est envoyé à son serveur principal qu'une fois que le serveur principal l'a demandé.
2. Configurez les paramètres TLS mutuels sur votre serveur, en spécifiant la chaîne de confiance et éventuellement les SAN fiables. Pour cela, votre serveur demande un certificat client.



## Vérification de l'authentification TLS mutuelle

Vous pouvez vous référer à la documentation [Transport Layer Security : Verify encryption](#) pour savoir exactement comment Envoy émet les statistiques relatives au TLS. Pour l'authentification TLS mutuelle, vous devez vérifier les statistiques suivantes :

- `ssl.handshake`
- `ssl.no_certificate`
- `ssl.fail_verify_no_cert`
- `ssl.fail_verify_san`

Les deux exemples de statistiques suivants montrent ensemble que les connexions TLS réussies aboutissant au nœud virtuel proviennent toutes d'un client qui a fourni un certificat.

```
listener.0.0.0.0_15000.ssl.handshake: 3
```

```
listener.0.0.0.0_15000.ssl.no_certificate: 0
```

L'exemple de statistique suivant montre que les connexions entre un nœud client virtuel (ou passerelle) et un nœud virtuel principal ont échoué. Le nom alternatif du sujet (SAN) présenté dans le certificat du serveur ne correspond à aucun des SAN approuvés par le client.

```
cluster.cds_egress_my-mesh_my-backend-node_http_9080.ssl.fail_verify_san: 5
```

## Procédures pas à pas de l'authentification TLS mutuelle App Mesh

- [Procédure pas à pas de l'authentification TLS mutuelle](#) : cette procédure explique comment utiliser l'App Mesh CLI pour créer une application couleur avec authentification TLS mutuelle.
- [Procédure pas à pas basée sur Amazon EKS Mutual TLS SDS](#) : cette présentation explique comment utiliser l'authentification mutuelle basée sur TLS SDS avec Amazon EKS et SPIFFE Runtime Environment (SPIRE).
- [Procédure pas à pas basée sur des fichiers TLS mutuels Amazon EKS](#) : cette présentation explique comment utiliser l'authentification mutuelle basée sur des fichiers TLS avec Amazon EKS et SPIFFE Runtime Environment (SPIRE).

# Comment AWS App Mesh fonctionne avec IAM

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources App Mesh. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

## Rubriques

- [Public ciblé](#)
- [Authentification par des identités](#)
- [Gestion des accès à l'aide de politiques](#)
- [Comment AWS App Mesh fonctionne avec IAM](#)
- [AWS App Mesh exemples de politiques basées sur l'identité](#)
- [AWS politiques gérées pour App Mesh](#)
- [Utilisation de rôles liés à un service pour App Mesh](#)
- [Automatisation du proxy Envoy](#)
- [Résolution des problèmes AWS App Mesh d'identité et d'accès](#)

## Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans App Mesh.

Utilisateur du service : si vous utilisez le service App Mesh pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de plus en plus de fonctionnalités App Mesh pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans App Mesh, consultez [Résolution des problèmes AWS App Mesh d'identité et d'accès](#).

Administrateur du service — Si vous êtes responsable des ressources App Mesh dans votre entreprise, vous avez probablement un accès complet à App Mesh. C'est à vous de déterminer les fonctionnalités et les ressources d'App Mesh auxquelles les utilisateurs de votre service doivent

accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser l'IAM avec App Mesh, consultez [Comment AWS App Mesh fonctionne avec IAM](#).

Administrateur IAM : si vous êtes administrateur IAM, vous souhaitez peut-être en savoir plus sur la manière dont vous pouvez rédiger des politiques pour gérer l'accès à App Mesh. Pour consulter des exemples de politiques basées sur l'identité App Mesh que vous pouvez utiliser dans IAM, consultez [AWS App Mesh exemples de politiques basées sur l'identité](#)

## Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section [Comment vous connecter à votre compte Compte AWS dans](#) le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section [Signature des demandes AWS d'API](#) dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir

plus, consultez [Authentification multifactorielle](#) dans le Guide de l'utilisateur AWS IAM Identity Center et [Utilisation de l'authentification multifactorielle \(MFA\) dans l'interface AWS](#) dans le Guide de l'utilisateur IAM.

## Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez [Tâches nécessitant les informations d'identification de l'utilisateur racine](#) dans le Guide de l'utilisateur IAM.

## Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité au sein de votre Compte AWS qui possède des autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez [Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification](#) dans le Guide de l'utilisateur IAM.

Un [groupe IAM](#) est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez [Quand créer un utilisateur IAM \(au lieu d'un rôle\)](#) dans le Guide de l'utilisateur IAM.

## Rôles IAM

Un [rôle IAM](#) est une identité au sein de votre Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console [changeant de rôle](#). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez [Utilisation de rôles IAM](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez la rubrique [Jeux d'autorisations](#) dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour connaître la différence entre les rôles et les politiques basées sur les ressources pour l'accès entre comptes, consultez la section Accès aux [ressources entre comptes dans IAM dans le guide de l'utilisateur IAM](#).
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
- Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service.

FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez [Sessions de transmission d'accès](#).

- Rôle de service : il s'agit d'un [rôle IAM](#) attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez [Création d'un rôle pour la délégation d'autorisations à un Service AWS](#) dans le Guide de l'utilisateur IAM.
- Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez [Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2](#) dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez [Quand créer un rôle IAM \(au lieu d'un utilisateur\)](#) dans le Guide de l'utilisateur IAM.

## Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des

documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

## Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Création de politiques IAM](#) dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

## politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour

contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

## Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez [Vue d'ensemble des listes de contrôle d'accès \(ACL\)](#) dans le Guide du développeur Amazon Simple Storage Service.

## Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- **Limite d'autorisations** : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- **Politiques de contrôle des services (SCP)** — Les SCP sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans. AWS



Organizations AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les organisations et les SCP, consultez [Fonctionnement des SCP](#) dans le Guide de l'utilisateur AWS Organizations .

- **Politiques de séance** : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez [politiques de séance](#) dans le Guide de l'utilisateur IAM.

## Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section [Logique d'évaluation des politiques](#) dans le guide de l'utilisateur IAM.

## Comment AWS App Mesh fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à App Mesh, vous devez connaître les fonctionnalités IAM disponibles avec App Mesh. Pour obtenir une vue d'ensemble de la façon dont App Mesh et les autres AWS services fonctionnent avec IAM, consultez la section [AWS Services That Work with IAM](#) dans le guide de l'utilisateur IAM.

### Rubriques

- [Politiques basées sur l'identité App Mesh](#)
- [Politiques basées sur les ressources App Mesh](#)
- [Autorisation basée sur les tags App Mesh](#)
- [Rôles IAM d'App Mesh](#)

## Politiques basées sur l'identité App Mesh

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. App Mesh prend en charge des actions, des ressources et des clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, consultez [Références des éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

### Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions politiques dans App Mesh utilisent le préfixe suivant avant l'action : `appmesh:`. Par exemple, pour autoriser une personne à répertorier les maillages d'un compte avec l'opération `appmesh:ListMeshes` API, vous devez inclure `appmesh:ListMeshes` dans sa politique. Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`.

Pour spécifier plusieurs actions dans une seule instruction, séparez-les par des virgules, comme suit :

```
"Action": [
 "appmesh:ListMeshes",
 "appmesh:ListVirtualNodes"
]
```

Vous pouvez aussi préciser plusieurs actions à l'aide de caractères génériques (\*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante.

```
"Action": "appmesh:Describe*"
```

Pour consulter la liste des actions App Mesh, consultez la section [Actions définies par AWS App Mesh](#) dans le guide de l'utilisateur IAM.

## Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son [Amazon Resource Name \(ARN\)](#). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (\*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

La mesh ressource App Mesh possède l'ARN suivant.

```
arn:${Partition}:appmesh:${Region}:${Account}:mesh/${MeshName}
```

Pour plus d'informations sur le format des ARN, consultez [Amazon Resource Names \(ARN\) et AWS Service Namespaces](#).

Par exemple, pour spécifier le maillage nommé *apps* dans la région *Region-code* dans votre instruction, utilisez l'ARN suivant.

```
arn:aws:appmesh:Region-code:111122223333:mesh/apps
```

Pour spécifier toutes les instances qui appartiennent à un compte spécifique, utilisez le caractère générique (\*).

```
"Resource": "arn:aws:appmesh:Region-code:111122223333:mesh/*"
```

Certaines actions App Mesh, telles que celles relatives à la création de ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (\*).

```
"Resource": "*"
```

De nombreuses actions de l'API App Mesh impliquent plusieurs ressources. Par exemple, `CreateRoute` crée une route avec une cible de nœud virtuel, de sorte qu'un utilisateur IAM doit être autorisé à utiliser la route et le nœud virtuel. Pour spécifier plusieurs ressources dans une seule instruction, séparez leurs ARN par des virgules.

```
"Resource": [
 "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualRouter/serviceB/route/*",
 "arn:aws:appmesh:Region-code:111122223333:mesh/apps/virtualNode/serviceB"
]
```

Pour consulter la liste des types de ressources App Mesh et de leurs ARN, consultez la section [Ressources définies par AWS App Mesh](#) dans le guide de l'utilisateur IAM. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez [Actions définies par AWS App Mesh](#).

## Clés de condition

App Mesh prend en charge l'utilisation de certaines clés de condition globales. Pour afficher toutes les clés de condition globales AWS, consultez la rubrique [Clés de contexte de condition globale AWS](#) dans le Guide de l'utilisateur IAM. Pour consulter la liste des clés de condition globales prises en charge par App Mesh, consultez la section [Clés de condition correspondantes AWS App Mesh](#) dans le guide de l'utilisateur IAM. Pour savoir quelles actions et ressources vous pouvez utiliser avec une clé de condition, consultez la section [Actions définies par AWS App Mesh](#).

## Exemples

Pour consulter des exemples de politiques basées sur l'identité App Mesh, consultez [AWS App Mesh exemples de politiques basées sur l'identité](#)

## Politiques basées sur les ressources App Mesh

App Mesh ne prend pas en charge les politiques basées sur les ressources. Toutefois, si vous utilisez le service AWS Resource Access Manager (AWS RAM) pour partager un maillage entre les AWS services, une politique basée sur les ressources est appliquée à votre maillage par le AWS RAM service. Pour plus d'informations, consultez [Octroi d'autorisations pour un maillage](#).

## Autorisation basée sur les tags App Mesh

Vous pouvez associer des tags aux ressources App Mesh ou transmettre des tags dans une demande à App Mesh. Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'[élément de condition](#) d'une politique utilisant les clés de condition `appmesh:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Pour plus d'informations sur le balisage des ressources App Mesh, consultez la section Ressources de [balisage AWS](#).

Pour visualiser un exemple de politique basée sur l'identité permettant de limiter l'accès à une ressource en fonction des balises de cette ressource, consultez [Création de maillages App Mesh avec des balises restreintes](#).

## Rôles IAM d'App Mesh

Un [rôle IAM](#) est une entité de votre AWS compte qui possède des autorisations spécifiques.

### Utilisation d'informations d'identification temporaires avec App Mesh

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à l'aide de la fédération, endosser un rôle IAM ou encore pour endosser un rôle intercompte. Vous obtenez des informations d'identification de sécurité temporaires en appelant des opérations d' AWS STS API telles que [AssumeRole](#) ou [GetFederationToken](#).

App Mesh prend en charge l'utilisation d'informations d'identification temporaires.

### Rôles liés à un service

Les [rôles liés aux](#) AWS services permettent aux services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre compte IAM et sont la propriété du service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

App Mesh prend en charge les rôles liés aux services. Pour plus d'informations sur la création ou la gestion des rôles liés au service App Mesh, consultez [Utilisation de rôles liés à un service pour App Mesh](#)

## Rôles de service

Cette fonction permet à un service d'endosser une [fonction du service](#) en votre nom. Ce rôle autorise le service à accéder à des ressources d'autres services pour effectuer une action en votre nom. Les rôles de service s'affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu'un administrateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

App Mesh ne prend pas en charge les rôles de service.

## AWS App Mesh exemples de politiques basées sur l'identité

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou à modifier des ressources App Mesh. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API AWS Management Console AWS CLI, ou. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour savoir comment créer une stratégie IAM basée sur l'identité à l'aide de ces exemples de documents de stratégie JSON, veuillez consulter [Création de stratégies dans l'onglet JSON](#) dans le Guide de l'utilisateur IAM.

### Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Utilisation de la console App Mesh](#)
- [Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations](#)
- [Création d'un maillage](#)
- [Répertoire et décrire tous les maillages](#)
- [Création de maillages App Mesh avec des balises restreintes](#)

## Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer des ressources App Mesh dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [politiques gérées par AWS](#) ou [politiques gérées par AWS pour les activités professionnelles](#) dans le Guide de l'utilisateur IAM.
- Accorder les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation de IAM pour appliquer des autorisations, consultez [politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.
- Utiliser des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : IAM Access Analyzer valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez [Validation de politique IAM Access Analyzer](#) dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue.

Compte AWS Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Configuration de l'accès aux API protégé par MFA](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez [Bonnes pratiques de sécurité dans IAM](#) dans le Guide de l'utilisateur IAM.

## Utilisation de la console App Mesh

Pour accéder à la AWS App Mesh console, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les informations relatives aux ressources App Mesh de votre AWS compte. Si vous créez une politique basée sur l'identité qui est plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs et rôles IAM) tributaires de cette politique. Vous pouvez associer la politique [AWSAppMeshReadOnly](#) gérée aux utilisateurs. Pour plus d'informations, consultez [Ajout d'autorisations à un utilisateur](#) dans le Guide de l'utilisateur IAM.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API que vous tentez d'effectuer.

## Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ViewOwnUserInfo",
 "Effect": "Allow",
 "Action": [
 "iam:GetUserPolicy",
 "iam:ListGroupsWithUser",
 "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": "*"
 }
]
}

```

## Création d'un maillage

Cet exemple montre comment créer une politique qui permet à un utilisateur de créer un maillage pour un compte, dans n'importe quelle région.

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "appmesh:CreateMesh",
 "Resource": "arn:aws:appmesh::*:123456789012:CreateMesh"
 }
]
}

```

## Répertorier et décrire tous les maillages

Cet exemple montre comment créer une politique permettant à un utilisateur d'accéder en lecture seule à la liste et à la description de tous les maillages.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "appmesh:DescribeMesh",
 "appmesh:ListMeshes"
],
 "Resource": "*"
 }
]
}
```

## Création de maillages App Mesh avec des balises restreintes

Vous pouvez utiliser des balises dans vos politiques IAM pour contrôler les balises qui peuvent être transmises dans la demande IAM. Vous pouvez spécifier les paires clé-valeur de balise qui peuvent être ajoutées, modifiées ou supprimées d'un utilisateur ou d'un rôle IAM. Cet exemple montre comment créer une politique permettant de créer un maillage, mais uniquement si le maillage est créé avec une balise nommée *TeamName* et une valeur *BookSteam*.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "appmesh:CreateMesh",
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "aws:RequestTag/teamName": "booksTeam"
 }
 }
 }
]
}
```

Vous pouvez rattacher cette politique aux utilisateurs IAM de votre compte. Si un utilisateur tente de créer un maillage, celui-ci doit inclure une balise nommée *teamName* et une valeur *debooksTeam*. Si le maillage n'inclut pas cette balise et cette valeur, la tentative de création du maillage échoue.

Pour plus d'informations, consultez [Éléments de politique JSON IAM : Condition](#) dans le Guide de l'utilisateur IAM.

## AWS politiques gérées pour App Mesh

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des [politiques gérées par le client](#) qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez la section [Politiques gérées par AWS](#) dans le Guide de l'utilisateur IAM.

### AWS politique gérée : AWSAppMeshServiceRolePolicy

Vous pouvez attacher AWSAppMeshServiceRolePolicy à vos entités IAM. Permet d'accéder aux AWS services et aux ressources utilisés ou gérés par AWS App Mesh.

Pour consulter les autorisations associées à cette politique, reportez-vous [AWSAppMeshServiceRolePolicy](#) à la référence des politiques AWS gérées.

Pour plus d'informations sur les détails des autorisations pour le AWSAppMeshServiceRolePolicy, voir [Autorisations de rôle liées à un service pour App Mesh](#).

### AWS politique gérée : AWSAppMeshEnvoyAccess

Vous pouvez attacher AWSAppMeshEnvoyAccess à vos entités IAM. Politique d'App Mesh Envoy pour accéder à la configuration des nœuds virtuels.

Pour consulter les autorisations associées à cette politique, reportez-vous [AWSAppMeshEnvoyAccess](#) à la référence des politiques AWS gérées.

### AWS politique gérée : AWSAppMeshFullAccess

Vous pouvez attacher `AWSAppMeshFullAccess` à vos entités IAM. Fournit un accès complet aux AWS App Mesh API et AWS Management Console.

Pour consulter les autorisations associées à cette politique, reportez-vous [AWSAppMeshFullAccess](#) à la référence des politiques AWS gérées.

### AWS politique gérée : AWSAppMeshPreviewEnvoyAccess

Vous pouvez attacher `AWSAppMeshPreviewEnvoyAccess` à vos entités IAM. Politique d'App Mesh Preview Envoy pour accéder à la configuration des nœuds virtuels.

Pour consulter les autorisations associées à cette politique, reportez-vous [AWSAppMeshPreviewEnvoyAccess](#) à la référence des politiques AWS gérées.

### AWS politique gérée : AWSAppMeshPreviewServiceRolePolicy

Vous pouvez attacher `AWSAppMeshPreviewServiceRolePolicy` à vos entités IAM. Permet d'accéder aux AWS services et aux ressources utilisés ou gérés par AWS App Mesh.

Pour consulter les autorisations associées à cette politique, reportez-vous [AWSAppMeshPreviewServiceRolePolicy](#) à la référence des politiques AWS gérées.

### AWS politique gérée : AWSAppMeshReadOnly

Vous pouvez attacher `AWSAppMeshReadOnly` à vos entités IAM. Fournit un accès en lecture seule aux AWS App Mesh API et AWS Management Console.

Pour consulter les autorisations associées à cette politique, reportez-vous [AWSAppMeshReadOnly](#) à la référence des politiques AWS gérées.

### AWS App Mesh mises à jour des politiques AWS gérées

Consultez les détails des mises à jour des politiques AWS gérées AWS App Mesh depuis que ce service a commencé à suivre ces modifications. Pour obtenir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page d'historique du document AWS App Mesh .

| Modification                                                                                                    | Description                                                                                                                                 | Date            |
|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <a href="#">AWSAppMeshFullAccess</a> — Politique mise à jour.                                                   | Mis AWSAppMeshFullAccess à jour pour permettre l'accès aux UntagResource API TagResource et.                                                | 24 avril 2024   |
| <a href="#">AWSAppMeshServiceRolePolicy</a> , <a href="#">AWSServiceRoleForAppMesh</a> — Politique mise à jour. | Mis à jour AWSServiceRoleForAppMesh et AWSAppMeshServiceRolePolicy pour permettre l'accès à l' AWS Cloud Map DiscoverInstancesRevision API. | 12 octobre 2023 |

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

Créez un jeu d'autorisations. Suivez les instructions de la rubrique [Création d'un jeu d'autorisations](#) du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

Créez un rôle pour la fédération d'identité. Pour plus d'informations, voir la rubrique [Création d'un rôle pour un fournisseur d'identité tiers \(fédération\)](#) du Guide de l'utilisateur IAM.

- Utilisateurs IAM :

- Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique [Création d'un rôle pour un utilisateur IAM](#) du Guide de l'utilisateur IAM.
- (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique [Ajout d'autorisations à un utilisateur \(console\)](#) du Guide de l'utilisateur IAM.

## Utilisation de rôles liés à un service pour App Mesh

AWS App Mesh utilise des [rôles AWS Identity and Access Management \(IAM\) liés aux services](#). Un rôle lié à un service est un type unique de rôle IAM directement lié à App Mesh. Les rôles liés aux

services sont prédéfinis par App Mesh et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'App Mesh, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. App Mesh définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul App Mesh peut assumer ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources App Mesh, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés aux services, consultez [Services AWS fonctionnant avec IAM](#) et recherchez les services où Oui figure dans la colonne Rôle lié à un service. Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

## Autorisations de rôle liées à un service pour App Mesh

App Mesh utilise le rôle lié au service nommé `AWSServiceRoleForAppMesh`— Le rôle permet à App Mesh d'appeler AWS des services en votre nom.

Le rôle `AWSServiceRoleForAppMesh` lié au service fait confiance au `appmesh.amazonaws.com` service pour assumer le rôle.

### Détails de l'autorisation

- `servicediscovery:DiscoverInstances`- Permet à App Mesh d'effectuer des actions sur toutes les AWS ressources.
- `servicediscovery:DiscoverInstancesRevision`- Permet à App Mesh d'effectuer des actions sur toutes les AWS ressources.

### `AWSServiceRoleForAppMesh`

Cette politique inclut les autorisations suivantes :

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

```
{
 "Sid": "CloudMapServiceDiscovery",
 "Effect": "Allow",
 "Action": [
 "servicediscovery:DiscoverInstances",
 "servicediscovery:DiscoverInstancesRevision"
],
 "Resource": "*"
},
{
 "Sid": "ACMCertificateVerification",
 "Effect": "Allow",
 "Action": [
 "acm:DescribeCertificate"
],
 "Resource": "*"
}
]
```

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez [Service-Linked Role Permissions \(autorisations du rôle lié à un service\)](#) dans le IAM User Guide (guide de l'utilisateur IAM).

## Création d'un rôle lié à un service pour App Mesh

Si vous avez créé un maillage après le 5 juin 2019 dans l'AWS Management Console, l'AWS CLI ou l'AWS CLI, App Mesh a créé le rôle lié au service pour vous. Pour que le rôle lié à un service ait été créé pour vous, le compte IAM que vous avez utilisé pour créer le maillage doit être associé à la politique [AWSAppMeshFullAccess](#) IAM ou à une politique contenant l'autorisation.

`iam:CreateServiceLinkedRole` Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un maillage, App Mesh crée à nouveau le rôle lié au service pour vous. Si votre compte contient uniquement des maillages créés avant le 5 juin 2019 et que vous souhaitez utiliser le rôle lié à un service avec ces maillages, vous pouvez créer le rôle à l'aide de la console IAM.

Vous pouvez utiliser la console IAM pour créer un rôle lié à un service avec le cas d'utilisation d'App Mesh. Dans l'interface AWS CLI ou l'API AWS, créez un rôle lié à un service avec le nom de service `appmesh.amazonaws.com`. Pour plus d'informations, consultez [Création d'un rôle lié à un service](#)

dans le Guide de l'utilisateur IAM. Si vous supprimez ce rôle lié à un service, vous pouvez utiliser ce même processus pour créer le rôle à nouveau.

## Modification d'un rôle lié à un service pour App Mesh

App Mesh ne vous permet pas de modifier le rôle `AWSServiceRoleForAppMesh` lié au service. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez [Editing a Service-Linked Role](#) (Modification d'un rôle lié à un service) dans le Guide de l'utilisateur IAM.

## Supprimer un rôle lié à un service pour App Mesh

Si vous n'avez plus besoin d'utiliser une fonction ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer les ressources de votre rôle lié à un service avant de pouvoir les supprimer manuellement.

### Note

Si le service App Mesh utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour supprimer les ressources App Mesh utilisées par le `AWSServiceRoleForAppMesh`

1. Supprimez tous les [itinéraires](#) définis pour tous les routeurs du maillage.
2. Supprimez tous les [routeurs virtuels](#) du maillage.
3. Supprimez tous les [services virtuels](#) du maillage.
4. Supprimez tous les [nœuds virtuels](#) du maillage.
5. Supprimez le [maillage](#).

Effectuez les étapes précédentes pour tous les maillages de votre compte.

Pour supprimer manuellement le rôle lié à un service à l'aide d'IAM

Utilisez la console IAM, l'AWS CLI, le ou l'AWSAPI pour supprimer le rôle lié au `AWSServiceRoleForAppMesh` service. Pour plus d'informations, veuillez consulter [Deleting a Service-Linked Role](#) (Suppression d'un rôle lié à un service) dans le Guide de l'utilisateur IAM.



## Régions prises en charge pour les rôles liés au service App Mesh

App Mesh prend en charge l'utilisation de rôles liés au service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez la section [App Mesh Endpoints and Quotas](#).

## Automatisation du proxy Envoy

L'autorisation de proxy autorise le proxy [Envoy](#) exécuté dans le cadre d'une tâche Amazon ECS, dans un pod Kubernetes exécuté sur Amazon EKS ou sur une instance Amazon EC2 à lire la configuration d'un ou de plusieurs points de terminaison de maillage à partir du service de gestion App Mesh Envoy. Pour les comptes clients dont les Envoys étaient déjà connectés à leur point de terminaison App Mesh avant le 26 avril 2021, une autorisation de proxy est requise pour les nœuds virtuels utilisant le [protocole TLS \(Transport Layer Security\)](#) et pour les passerelles virtuelles (avec ou sans TLS). Pour les comptes clients qui souhaitent connecter Envoys à leur point de terminaison App Mesh après le 26 avril 2021, une autorisation de proxy est requise pour toutes les fonctionnalités d'App Mesh. Il est recommandé à tous les comptes clients d'activer l'autorisation par proxy pour tous les nœuds virtuels, même s'ils n'utilisent pas le protocole TLS, afin de bénéficier d'une expérience sécurisée et cohérente en utilisant IAM pour l'autorisation de ressources spécifiques. L'autorisation du proxy nécessite que `appmesh:StreamAggregatedResourcesautorisation` soit spécifiée dans une politique IAM. La politique doit être associée à un rôle IAM, et ce rôle IAM doit être associé à la ressource de calcul sur laquelle vous hébergez le proxy.

## Créer une politique IAM

Si vous souhaitez que tous les points d'extrémité du maillage d'un maillage de service puissent lire la configuration de tous les points d'extrémité du maillage, passez directement à [Créez un rôle IAM](#). Si vous souhaitez limiter les points de terminaison du maillage à partir desquels la configuration peut être lue par des points de terminaison de maillage individuels, vous devez créer une ou plusieurs politiques IAM. Il est recommandé de limiter les points de terminaison du maillage à partir desquels la configuration peut être lue uniquement au proxy Envoy s'exécutant sur des ressources de calcul spécifiques. Créez une stratégie IAM et ajoutez `appmesh:StreamAggregatedResourcesautorisation` à la stratégie. L'exemple de politique suivant permet de configurer les nœuds virtuels nommés `serviceBv1` et `serviceBv2` les lire dans un maillage de service. La configuration ne peut être lue pour aucun autre nœud virtuel défini dans le maillage de service. Pour plus d'informations sur la création ou la modification d'une stratégie IAM, consultez [Création de stratégies IAM](#), consultez [Création de stratégies IAM](#) et [Création de stratégies 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"
]
 }
]
}
```

Vous pouvez créer plusieurs politiques, chaque politique restreignant l'accès aux différents points de terminaison du maillage.

## Créez un rôle IAM

Si vous souhaitez que tous les points de terminaison d'un maillage de service puissent lire la configuration de tous les points de terminaison du maillage, il vous suffit de créer un seul rôle IAM. Si vous souhaitez limiter les points d'extrémité du maillage à partir desquels la configuration peut être lue par des points de terminaison de maillage individuels, vous devez créer un rôle pour chaque politique que vous avez créée à l'étape précédente. Suivez les instructions relatives à la ressource de calcul sur laquelle le proxy s'exécute.

- Amazon EKS — Si vous souhaitez utiliser un seul rôle, vous pouvez utiliser le rôle existant qui a été créé et attribué aux nœuds de travail lorsque vous avez créé votre cluster. Pour utiliser plusieurs rôles, votre cluster doit répondre aux exigences définies dans [Activation des rôles IAM pour les rôles IAM pour les comptes de service sur votre cluster](#). Créez les rôles IAM et associez-les à des comptes de service Kubernetes. Pour plus d'informations, consultez [Création d'un rôle et d'une politique IAM pour votre compte de service](#) et [Spécification d'un rôle IAM pour votre compte de service](#).
- Amazon ECS : sélectionnez un AWSservice, sélectionnez Elastic Container Service, puis sélectionnez le cas d'utilisation d'Elastic Container Service Task lors de la création de votre rôle IAM.

- Amazon EC2 : sélectionnez le AWSservice, sélectionnez EC2, puis sélectionnez le cas d'utilisation d'EC2 lors de la création de votre rôle IAM. Cela s'applique, que vous hébergiez le proxy directement sur une instance Amazon EC2 ou sur Kubernetes, exécuté sur une instance.

Pour plus d'informations sur la façon de créer un rôle IAM, consultez [Création d'un rôle pour unAWS service](#).

## Attachement d'une stratégie IAM

Si vous souhaitez que tous les points de terminaison d'un maillage de service puissent lire la configuration de tous les points de terminaison du maillage, associez la politique IAM [AWSAppMeshEnvoyAccess](#) gérée au rôle IAM que vous avez créé à l'étape précédente. Si vous souhaitez limiter les points de terminaison du maillage à partir desquels la configuration peut être lue par des points de terminaison de maillage individuels, associez chaque politique que vous avez créée à chaque rôle que vous avez créé. Pour plus d'informations sur l'association d'une politique IAM personnalisée ou gérée à un rôle IAM, consultez la section [Ajout d'autorisations d'identité IAM](#).

## Attachement d'un rôle IAM

Associez chaque rôle IAM à la ressource de calcul appropriée :

- Amazon EKS : si vous avez associé la politique au rôle associé à vos nœuds de travail, vous pouvez ignorer cette étape. Si vous avez créé des rôles distincts, attribuez chaque rôle à un compte de service Kubernetes distinct et attribuez chaque compte de service à une spécification de déploiement de pod Kubernetes individuelle qui inclut le proxy Envoy. Pour plus d'informations, consultez la section [Définition d'un rôle IAM pour votre compte de service](#) dans le guide de l'utilisateur Amazon EKS et la section [Configuration de comptes de service pour les pods](#) dans la documentation Kubernetes.
- Amazon ECS : associez un rôle de tâche Amazon ECS à la définition de tâche qui inclut le proxy Envoy. La tâche peut être déployée avec le type de lancement EC2 ou Fargate. Pour plus d'informations sur la façon de créer un rôle de tâche Amazon ECS et de l'associer à une tâche, consultez [Spécifier un rôle IAM pour vos tâches](#).
- Amazon EC2 — Le rôle IAM doit être associé à l'instance Amazon EC2 qui héberge le proxy Envoy. Pour plus d'informations sur la manière d'associer un rôle à une instance Amazon EC2, consultez [J'ai créé un rôle IAM et je souhaite maintenant l'attribuer à une instance EC2](#).

## Confirmation de l'autorisation

Vérifiez que `appmesh:StreamAggregatedResources` autorisation est attribuée à la ressource de calcul sur laquelle vous hébergez le proxy en sélectionnant l'un des noms de service de calcul.

### Amazon EKS

Une politique personnalisée peut être attribuée au rôle attribué aux nœuds de travail, à des pods individuels, ou aux deux. Il est toutefois recommandé d'attribuer la politique uniquement à des espaces individuels, afin de pouvoir restreindre l'accès de chaque espace à des points de terminaison du maillage individuels. Si la politique est associée au rôle attribué aux nœuds de travail, sélectionnez l'onglet Amazon EC2 et suivez les étapes qui s'y trouvent pour vos instances de nœuds de travail. Pour déterminer quel rôle IAM est attribué à un pod Kubernetes, procédez comme suit.

1. Consultez les détails d'un déploiement Kubernetes qui inclut le pod auquel vous souhaitez confirmer qu'un compte de service Kubernetes est attribué. La commande suivante permet d'afficher les détails d'un déploiement nommé *my-deployment*.

```
kubectl describe deployment my-deployment
```

Dans la sortie renvoyée, notez la valeur située à droite de `Service Account` :. Si aucune ligne commençant par `Service Account` : n'existe, aucun compte de service Kubernetes personnalisé n'est actuellement attribué au déploiement. Vous aurez besoin d'en attribuer un. Pour plus d'informations, consultez [Configurer des comptes de service pour les pods](#) dans la documentation Kubernetes.

2. Consultez les détails du service renvoyé à l'étape précédente. La commande suivante permet d'afficher les détails d'un compte de service nommé *my-service-account*.

```
kubectl describe serviceaccount my-service-account
```

À condition que le compte de service Kubernetes soit associé à un AWS Identity and Access Management rôle, l'une des lignes renvoyées ressemblera à l'exemple suivant.

```
Annotations: eks.amazonaws.com/role-arn=arn:aws:iam::123456789012:role/
my-deployment
```

Dans l'exemplemy -deployment précédent, le nom du rôle IAM auquel le compte de service est associé. Si la sortie du compte de service ne contient pas de ligne similaire à l'exemple ci-dessus, le compte de service Kubernetes n'est pas associé à unAWS Identity and Access Management compte et vous devez l'associer à un compte. Pour de plus amples informations, consultez [Spécification d'un rôle IAM pour votre compte de service](#).

3. Connectez-vous à l'outil AWS Management Console, puis ouvrez la console IAM à l'adresse <https://console.aws.amazon.com/iam/>.
4. Sélectionnez Rôles dans le volet de navigation de gauche. Sélectionnez le nom du rôle IAM que vous avez noté lors d'une étape précédente.
5. Vérifiez que la politique personnalisée que vous avez créée précédemment ou que la politique [AWSAppMeshEnvoyAccess](#) gérée est répertoriée. Si aucune stratégie n'est associée, [associez une stratégie IAM](#) au rôle IAM. Si vous souhaitez associer une politique IAM personnalisée mais que vous n'en avez pas, vous devez [créer une stratégie IAM personnalisée](#) avec les autorisations requises. Si une stratégie IAM personnalisée est jointe, sélectionnez-la et confirmez qu'elle contient "Action" : "appmesh:StreamAggregatedResources". Si ce n'est pas le cas, vous devez ajouter cette autorisation à votre politique IAM personnalisée. Vous pouvez également vérifier que l'Amazon Resource (ARN) approprié pour un point de maillage (ARN) approprié pour un point de maillage spécifique. Si aucun ARN n'est répertorié, vous pouvez modifier la politique pour ajouter, supprimer ou modifier les ARN répertoriés. Pour plus d'informations, consultez [Modifier les stratégies IAM](#) et [Créer une politique IAM](#).
6. Répétez les étapes précédentes pour chaque pod Kubernetes contenant le proxy Envoy.

## Amazon ECS

1. Dans la console Amazon ECS, choisissez Task Definitions.
2. Sélectionnez votre tâche Amazon ECS.
3. Sur la page Nom de la définition de la tâche, sélectionnez votre définition de tâche.
4. Sur la page Définition de la tâche, sélectionnez le lien du nom du rôle IAM qui se trouve à droite du rôle de tâche. Si aucun rôle IAM n'est répertorié, vous devez [créer un rôle IAM](#) et l'associer à votre tâche en [mettant à jour votre définition de tâche](#).
5. Sur la page Résumé, dans l'onglet Autorisations, confirmez que la politique personnalisée que vous avez créée précédemment ou que la politique [AWSAppMeshEnvoyAccess](#) gérée est répertoriée. Si aucune stratégie n'est associée, [associez une stratégie IAM](#)

au rôle IAM. Si vous souhaitez associer une politique IAM personnalisée mais que vous n'en avez pas, vous devez [créer la politique IAM personnalisée](#). Si une stratégie IAM personnalisée est jointe, sélectionnez-la et confirmez qu'elle contient "Action": "appmesh:StreamAggregatedResources". Si ce n'est pas le cas, vous devez ajouter cette autorisation à votre politique IAM personnalisée. Vous pouvez également vérifier que l'Amazon Resource (ARN) approprié pour un maillage spécifique est répertorié. Si aucun ARN n'est répertorié, vous pouvez modifier la politique pour ajouter, supprimer ou modifier les ARN répertoriés. Pour plus d'informations, consultez [Modifier les stratégies IAM](#) et [Créer une politique IAM](#).

6. Répétez les étapes précédentes pour chaque définition de tâche contenant le proxy Envoy.

## Amazon EC2

1. Sélectionnez Instances dans le panneau de navigation de gauche.
2. Sélectionnez l'une de vos instances qui héberge le proxy Envoy.
3. Dans l'onglet Description, sélectionnez le lien du nom du rôle IAM situé à droite du rôle IAM. Si aucun rôle IAM n'est répertorié, vous devez en [créer un](#).
4. Sur la page Résumé, dans l'onglet Autorisations, confirmez que la politique personnalisée que vous avez créée précédemment ou que la politique [AWSAppMeshEnvoyAccess](#) gérée est répertoriée. Si aucune stratégie n'est associée, [associez la stratégie IAM](#) au rôle IAM. Si vous souhaitez associer une politique IAM personnalisée mais que vous n'en avez pas, vous devez [créer la politique IAM personnalisée](#). Si une stratégie IAM personnalisée est jointe, sélectionnez-la et confirmez qu'elle contient "Action": "appmesh:StreamAggregatedResources". Si ce n'est pas le cas, vous devez ajouter cette autorisation à votre politique IAM personnalisée. Vous pouvez également vérifier que l'Amazon Resource (ARN) approprié pour un maillage spécifique est répertorié. Si aucun ARN n'est répertorié, vous pouvez modifier la politique pour ajouter, supprimer ou modifier les ARN répertoriés. Pour plus d'informations, consultez [Modifier les stratégies IAM](#) et [Créer une politique IAM](#).
5. Répétez les étapes précédentes pour chaque instance sur laquelle vous hébergez le proxy Envoy.

## Résolution des problèmes AWS App Mesh d'identité et d'accès

Utilisez les informations suivantes pour diagnostiquer et résoudre les problèmes courants que vous pouvez rencontrer lors de l'utilisation d'App Mesh et d'IAM.

### Rubriques

- [Je ne suis pas autorisé à effectuer une action dans App Mesh](#)
- [Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources App Mesh](#)

### Je ne suis pas autorisé à effectuer une action dans App Mesh

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'erreur suivante se produit lorsque l'utilisateur mateojackson IAM essaie d'utiliser la console pour créer un nœud virtuel nommé *my-virtual-node* dans le maillage nommé *my-mesh* mais n'en a pas l'autorisation. `appmesh:CreateVirtualNode`

```
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
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre de créer un nœud virtuel à l'aide de l'`appmesh:CreateVirtualNode` action.

#### Note

Comme un nœud virtuel est créé dans un maillage, le compte de Mateo nécessite également les `appmesh:ListMeshes` actions `appmesh:DescribeMesh` et pour créer le nœud virtuel dans la console.

## Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources App Mesh

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si App Mesh prend en charge ces fonctionnalités, consultez [Comment AWS App Mesh fonctionne avec IAM](#).
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section [Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section [Fournir un accès à des ressources Comptes AWS détenues par des tiers](#) dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez [Fournir un accès à des utilisateurs authentifiés en externe \(fédération d'identité\)](#) dans le Guide de l'utilisateur IAM.
- Pour connaître la différence entre l'utilisation de rôles et de politiques basées sur les ressources pour l'accès entre comptes, consultez la section Accès aux [ressources entre comptes dans IAM dans le guide de l'utilisateur d'IAM](#).

## Journalisation des appels AWS App Mesh d'API à l'aide AWS CloudTrail

AWS App Mesh est intégré à [AWS CloudTrail](#) un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS. CloudTrail capture tous les appels d'API pour App Mesh sous forme d'événements. Les appels capturés incluent des appels provenant de la console App Mesh et des appels de code vers les opérations de l'API App Mesh. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à App Mesh, l'adresse IP à partir de laquelle la demande a été faite, la date à laquelle elle a été faite et des informations supplémentaires.



Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou d'utilisateur root.
- Si la demande a été faite au nom d'un utilisateur de l'IAM Identity Center.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

CloudTrail est actif dans votre compte Compte AWS lorsque vous créez le compte et vous avez automatiquement accès à l'historique des CloudTrail événements. L'historique des CloudTrail événements fournit un enregistrement consultable, consultable, téléchargeable et immuable des 90 derniers jours des événements de gestion enregistrés dans un. Région AWS Pour plus d'informations, consultez la section [Utilisation de l'historique des CloudTrail événements](#) dans le guide de AWS CloudTrail l'utilisateur. La consultation de CloudTrail l'historique des événements est gratuite.

Pour un enregistrement continu des événements de vos 90 Compte AWS derniers jours, créez un magasin de données sur les événements de Trail ou [CloudTrailLake](#).

## CloudTrail sentiers

Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Tous les sentiers créés à l'aide du AWS Management Console sont multirégionaux. Vous pouvez créer un parcours à région unique ou multirégionale à l'aide du. AWS CLI Il est recommandé de créer un parcours multirégional, car vous capturez l'activité dans l'ensemble Régions AWS de votre compte. Si vous créez un parcours à région unique, vous ne pouvez voir que les événements enregistrés dans le parcours. Région AWS Pour plus d'informations sur les sentiers, consultez les [sections Création d'un sentier pour votre organisation](#) Compte AWS et [Création d'un sentier pour une organisation](#) dans le guide de AWS CloudTrail l'utilisateur.

Vous pouvez envoyer une copie de vos événements de gestion en cours à votre compartiment Amazon S3 gratuitement CloudTrail en créant un journal. Toutefois, des frais de stockage Amazon S3 sont facturés. Pour plus d'informations sur la CloudTrail tarification, consultez la section [AWS CloudTrail Tarification](#). Pour obtenir des informations sur la tarification Amazon S3, consultez [Tarification Amazon S3](#).

## CloudTrail Stockages de données sur les événements du lac

CloudTrail Lake vous permet d'exécuter des requêtes SQL sur vos événements. CloudTrail Lake convertit les événements existants au format JSON basé sur les lignes au format [Apache ORC](#). ORC est un format de stockage en colonnes qui est optimisé pour une récupération rapide des données. Les événements sont agrégés dans des magasins de données d'événement. Ceux-ci constituent des collections immuables d'événements basées sur des critères que vous sélectionnez en appliquant des [sélecteurs d'événements avancés](#). Les sélecteurs que vous appliquez à un magasin de données d'événement contrôlent les événements qui persistent et que vous pouvez interroger. Pour plus d'informations sur CloudTrail Lake, consultez la section [Travailler avec AWS CloudTrail Lake](#) dans le guide de AWS CloudTrail l'utilisateur.

CloudTrail Les stockages et requêtes de données sur les événements de Lake entraînent des coûts. Lorsque vous créez un magasin de données d'événement, vous choisissez l'[option de tarification](#) que vous voulez utiliser pour le magasin de données d'événement. L'option de tarification détermine le coût d'ingestion et de stockage des événements, ainsi que les périodes de conservation par défaut et maximale pour le magasin de données d'événement. Pour plus d'informations sur la CloudTrail tarification, consultez la section [AWS CloudTrail Tarification](#).

## Événements de gestion d'App Mesh dans CloudTrail

[Les événements de gestion](#) fournissent des informations sur les opérations de gestion effectuées sur les ressources de votre Compte AWS. Ils sont également connus sous le nom opérations de plan de contrôle. Par défaut, CloudTrail enregistre les événements de gestion.

AWS App Mesh enregistre toutes les opérations du plan de contrôle App Mesh en tant qu'événements de gestion. Pour obtenir la liste des opérations du plan de AWS App Mesh contrôle auxquelles App Mesh se connecte CloudTrail, consultez la [référence des AWS App Mesh API](#).

## Exemples d'événements App Mesh

Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'opération d'API demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics. Les événements n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`StreamAggregatedResources` action.

```
{
 "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"
}
```

Pour plus d'informations sur le contenu des CloudTrail enregistrements, voir [le contenu des CloudTrail enregistrements](#) dans le Guide de AWS CloudTrail l'utilisateur.

## Protection des données dans AWS App Mesh

Le [modèle de responsabilité AWS partagée](#) de s'applique à la protection des données dans AWS App Mesh. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration

et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez [Questions fréquentes \(FAQ\) sur la confidentialité des données](#). Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog [Modèle de responsabilité partagée AWS et RGPD \(Règlement général sur la protection des données\)](#) sur le Blog de sécurité AWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez [Federal Information Processing Standard \(FIPS\) 140-2](#) (Normes de traitement de l'information fédérale).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec App Mesh ou autre à Services AWS l'aide de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

## Chiffrement des données

Vos données sont cryptées lorsque vous utilisez App Mesh.

### Chiffrement au repos

Par défaut, les configurations App Mesh que vous créez sont chiffrées au repos.

### Chiffrement en transit

Les points de terminaison du service App Mesh utilisent le protocole HTTPS. Toutes les communications entre le proxy Envoy et le service de gestion App Mesh Envoy sont cryptées. Si vous avez besoin d'un chiffrement conforme à la norme FIPS pour la communication entre le proxy Envoy et le service de gestion App Mesh Envoy, vous pouvez utiliser une variante FIPS de l'image de conteneur du proxy Envoy. Pour plus d'informations, consultez [Image de l'envoyé](#).

La communication entre les conteneurs au sein des nœuds virtuels n'est pas cryptée, mais ce trafic ne quitte pas l'espace de noms du réseau.

## Validation de conformité pour AWS App Mesh

Pour savoir si un [programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité](#) spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de [AWS conformité Programmes AWS](#) de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir [Téléchargement de rapports dans AWS Artifact](#) .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- [Guides de démarrage rapide sur la sécurité et la conformité](#) : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur AWS la sécurité et la conformité.
- [Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services](#) : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.

**Note**

Tous ne Services AWS sont pas éligibles à la loi HIPAA. Pour plus d'informations, consultez le [HIPAA Eligible Services Reference](#).

- AWS Ressources de <https://aws.amazon.com/compliance/resources/> de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- [AWS Guides de conformité destinés aux clients](#) — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résumant les meilleures pratiques en matière de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).
- [Évaluation des ressources à l'aide des règles](#) du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- [AWS Security Hub](#)— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez [Référence des contrôles Security Hub](#).
- [Amazon GuardDuty](#) — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- [AWS Audit Manager](#)— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

## Sécurité de l'infrastructure dans AWS App Mesh

En tant que service géré, AWS App Mesh il est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement

en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à App Mesh via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser [AWS Security Token Service](#) (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Vous pouvez améliorer le niveau de sécurité de votre VPC en configurant App Mesh pour qu'il utilise un point de terminaison VPC d'interface. Pour plus d'informations, voir [Points de terminaison VPC de l'interface App Mesh \(\)AWS PrivateLink](#).

## Points de terminaison VPC de l'interface App Mesh ()AWS PrivateLink

Vous pouvez améliorer le niveau de sécurité de votre Amazon VPC en configurant App Mesh pour qu'il utilise un point de terminaison VPC d'interface. Les points de terminaison de l'interface sont alimentés par AWS PrivateLink une technologie qui vous permet d'accéder en privé aux API App Mesh en utilisant des adresses IP privées. PrivateLink restreint tout le trafic réseau entre votre Amazon VPC et App Mesh vers le réseau Amazon.

Vous n'êtes pas obligé de le configurer PrivateLink, mais nous vous recommandons de le faire. Pour plus d'informations sur les points de terminaison VPC PrivateLink et leur interface, consultez la section [Accès aux services via. AWS PrivateLink](#)

## Considérations relatives aux points de terminaison VPC de l'interface App Mesh

Avant de configurer les points de terminaison VPC de l'interface pour App Mesh, tenez compte des points suivants :

- Si votre Amazon VPC ne possède pas de passerelle Internet et que vos tâches utilisent le pilote de journal pour envoyer des informations de `awslogs` journal à CloudWatch Logs, vous devez créer

un point de terminaison VPC d'interface pour les journaux. CloudWatch Pour plus d'informations, consultez la section [Utilisation CloudWatch des journaux avec les points de terminaison VPC d'interface dans le guide](#) de l'utilisateur Amazon CloudWatch Logs.

- Les points de terminaison VPC ne prennent pas en charge AWS les demandes interrégionales. Assurez-vous de créer votre point de terminaison dans la même région que celle où vous prévoyez d'envoyer vos appels d'API à App Mesh.
- Les points de terminaison d'un VPC prennent uniquement en charge le DNS fourni par Amazon via Amazon Route 53. Si vous souhaitez utiliser votre propre DNS, vous pouvez utiliser le transfert DNS conditionnel. Pour en savoir plus, consultez [Jeux d'options DHCP](#) dans le Guide de l'utilisateur Amazon VPC.
- Le groupe de sécurité attaché au point de terminaison du VPC doit autoriser les connexions entrantes sur le port 443 depuis le sous-réseau privé de l'Amazon VPC.

#### Note

Le contrôle de l'accès à App Mesh en associant une politique de point de terminaison au point de terminaison VPC (par exemple, en utilisant le nom du service `com.amazonaws.Region.appmesh-envoy-management`) n'est pas pris en charge pour la connexion Envoy.

Pour des considérations et limitations supplémentaires, voir [Considérations relatives à la zone de disponibilité des points de terminaison d'interface et Propriétés et limites des points de terminaison d'interface](#).

## Créez le point de terminaison VPC de l'interface pour App Mesh

Pour créer le point de terminaison VPC d'interface pour le service App Mesh, utilisez la procédure de [création d'un point de terminaison d'interface](#) dans le guide de l'utilisateur Amazon VPC. Spécifiez `com.amazonaws.Region.appmesh-envoy-management` le nom du service pour que votre proxy Envoy se connecte au service de gestion Envoy public d'App Mesh et `com.amazonaws.Region.appmesh` pour les opérations de maillage.

#### Note

La *région* représente l'identifiant de région d'une AWS région prise en charge par App Mesh, par exemple `us-east-2` pour la région USA Est (Ohio).



Bien que vous puissiez définir un point de terminaison VPC d'interface pour App Mesh dans toutes les régions où App Mesh est pris en charge, il se peut que vous ne puissiez pas définir un point de terminaison pour toutes les zones de disponibilité de chaque région. Pour savoir quelles zones de disponibilité sont prises en charge avec les points de terminaison VPC d'interface dans une région, utilisez la [describe-vpc-endpoint-services](#) commande ou utilisez le. AWS Management Console Par exemple, les commandes suivantes renvoient les zones de disponibilité dans lesquelles vous pouvez déployer des points de terminaison VPC d'interface App Mesh dans la région USA Est (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[]'
```

## Résilience dans AWS App Mesh

L'infrastructure mondiale d'AWS repose sur les Régions AWS et les zones de disponibilité AWS. Les Régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

App Mesh exécute ses instances du plan de disponibilité sur plusieurs zones de disponibilité, ce qui permet d'assurer une haute disponibilité. App Mesh détecte et remplace automatiquement les instances du plan de contrôle qui ne sont pas en bonne santé, et fournit des mises à jour et des correctifs automatiques pour ces instances.

## Reprise après sinistre dans AWS App Mesh

Le service App Mesh gère les sauvegardes des données des clients. Vous n'avez rien à faire pour gérer les sauvegardes. Les données sauvegardées sont cryptées.

## Analyse de configuration et de vulnérabilité dans AWS App Mesh

App Mesh vend une [image de conteneur Docker proxy Envoy](#) gérée que vous déployez avec vos microservices. App Mesh garantit que l'image du conteneur est corrigée avec les derniers correctifs

de vulnérabilité et de performance. App Mesh teste les nouvelles versions du proxy Envoy par rapport à l'ensemble de fonctionnalités App Mesh avant de mettre les images à votre disposition.

Vous devez mettre à jour vos microservices pour utiliser la version actualisée de l'image du conteneur. Voici la dernière version de l'image.

```
840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-envoy:v1.29.5.0-prod
```

# Résolution des problèmes liés à App Mesh

Ce chapitre décrit les meilleures pratiques de résolution des problèmes et les problèmes courants que vous pouvez rencontrer lors de l'utilisation d'App Mesh. Sélectionnez l'un des domaines suivants pour passer en revue les meilleures pratiques et les problèmes courants dans ce domaine.

## Rubriques

- [Meilleures pratiques de résolution des problèmes liés à App Mesh](#)
- [Résolution des problèmes liés à la configuration d'App Mesh](#)
- [Résolution des problèmes de connectivité App Mesh](#)
- [Résolution des problèmes liés au dimensionnement de l'App Mesh](#)
- [Résolution des problèmes liés à l'observabilité de l'App Mesh](#)
- [Résolution des problèmes de sécurité liés à App Mesh](#)
- [Résolution des problèmes liés à App Mesh Kubernetes](#)

## Meilleures pratiques de résolution des problèmes liés à App Mesh

Nous vous recommandons de suivre les meilleures pratiques décrites dans cette rubrique pour résoudre les problèmes liés à l'utilisation d'App Mesh.

### Activer l'interface d'administration du proxy Envoy

Le proxy Envoy est fourni avec une interface d'administration que vous pouvez utiliser pour découvrir la configuration et les statistiques et pour exécuter d'autres fonctions administratives telles que le drainage des connexions. Pour plus d'informations, consultez la section [Interface d'administration](#) dans la documentation d'Envoy.

Si vous utilisez le point de terminaison géré [Image de l'envoyé](#), le point de terminaison d'administration est activé par défaut sur le port 9901. Les exemples fournis dans [Résolution des problèmes liés à la configuration d'App Mesh](#) affichent l'exemple d'URL du point de terminaison d'administration sous la forme `http://my-app.default.svc.cluster.local:9901/`.

#### Note

Le terminal d'administration ne doit jamais être exposé à l'Internet public. En outre, nous vous recommandons de surveiller les journaux des points de terminaison d'administration, qui sont

définis par la variable d'ENVVOY\_ADMIN\_ACCESS\_LOG\_FILE environnement sur « /tmp/ envoy\_admin\_access.log par défaut ».

## Activer l'intégration d'Envoy DogStats D pour le téléchargement métrique

Le proxy Envoy peut être configuré pour télécharger les statistiques relatives au trafic OSI des couches 4 et 7 et à l'état des processus internes. Bien que cette rubrique explique comment utiliser ces statistiques sans les transférer vers des indicateurs tels que CloudWatch Metrics et Prometheus, le fait de disposer de ces statistiques dans un emplacement centralisé pour toutes vos applications peut vous aider à diagnostiquer les problèmes et à confirmer le comportement plus rapidement. Pour plus d'informations, consultez [Using Amazon CloudWatch Metrics](#) et la documentation [Prometheus](#).

Vous pouvez configurer les métriques DogStats D en définissant les paramètres définis dans [DogStatsVariables D](#). Pour plus d'informations sur DogStats D, consultez la documentation [DogStatsD](#). Vous trouverez une démonstration du transfert des métriques vers les AWS CloudWatch métriques dans l'[App Mesh, ainsi que la présentation des principes de base d'Amazon ECS](#). GitHub

## Activer les journaux d'accès

Nous vous recommandons d'activer les journaux d'accès sur votre [Passerelle](#) ordinateur [Nœuds virtuels](#) et de découvrir les détails du trafic transitant entre vos applications. Pour plus d'informations, consultez la section [Journalisation des accès](#) dans la documentation d'Envoy. Les journaux fournissent des informations détaillées sur le comportement du trafic des couches 4 et 7 de l'OSI. Lorsque vous utilisez le format par défaut d'Envoy, vous pouvez analyser les journaux d'accès avec CloudWatch Logs [Insights](#) à l'aide de l'instruction d'analyse suivante.

```
parse @message "[*] \"* * *\" * * * * * * * * * * *\" as StartTime, Method, Path, Protocol, ResponseCode, ResponseFlags, BytesReceived, BytesSent, DurationMillis, UpstreamServiceTimeMillis, ForwardedFor, UserAgent, RequestId, Authority, UpstreamHost
```

## Activer la journalisation du débogage d'Envoy dans les environnements de pré-production

Nous recommandons de définir le niveau de journalisation du proxy Envoy sur debug un environnement de pré-production. Les journaux de débogage peuvent vous aider à identifier les problèmes avant d'intégrer la configuration App Mesh associée à votre environnement de production.

Si vous utilisez l'[image Envoy](#), vous pouvez définir le niveau de journalisation debug via la variable d'ENVVOY\_LOG\_LEVEL environnement.

#### Note

Nous déconseillons d'utiliser ce debug niveau dans les environnements de production. La définition du niveau de manière à debug augmenter la journalisation peut affecter les performances et le coût global des journaux déchargés vers des solutions telles que [CloudWatch Logs](#).

Lorsque vous utilisez le format par défaut d'Envoy, vous pouvez analyser les journaux de processus avec CloudWatch Logs [Insights](#) à l'aide de l'instruction d'analyse suivante :

```
parse @message "[*][*][*][*] [*] *" as Time, Thread, Level, Name, Source, Message
```

## Surveillez la connectivité du proxy Envoy avec le plan de contrôle App Mesh

Nous vous recommandons de surveiller les métriques Envoy `control_plane.connected_state` pour vous assurer que le proxy Envoy communique avec le plan de contrôle App Mesh pour récupérer les ressources de configuration dynamique. Pour plus d'informations, consultez [la section Serveur de gestion](#).

## Résolution des problèmes liés à la configuration d'App Mesh

Cette rubrique décrit les problèmes courants que vous pouvez rencontrer lors de la configuration d'App Mesh.

### Impossible d'extraire l'image du conteneur Envoy

#### Symptômes

Vous recevez le message d'erreur suivant dans une tâche Amazon ECS. L'*ID du compte* Amazon ECR et *la région* indiqués dans le message suivant peuvent être différents, selon le référentiel Amazon ECR d'où vous avez extrait l'image du conteneur.

```
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'
```

## Résolution

Cette erreur indique que le rôle d'exécution de tâche utilisé n'est pas autorisé à communiquer avec Amazon ECR et ne peut pas extraire l'image du conteneur Envoy du référentiel. Le rôle d'exécution de tâche attribué à votre tâche Amazon ECS nécessite une politique IAM comportant les déclarations suivantes :

```
{
 "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"
}
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Impossible de se connecter au service de gestion App Mesh Envoy

### Symptômes

Votre proxy Envoy ne parvient pas à se connecter au service de gestion App Mesh Envoy. Vous êtes en train de voir :

- Erreurs de connexion refusées
- Délais de connexion
- Erreurs lors de la résolution du point de terminaison du service de gestion App Mesh Envoy
- Erreurs gRPC

### Résolution

Assurez-vous que votre proxy Envoy a accès à Internet ou à un point de [terminaison VPC](#) privé et que vos [groupes de sécurité](#) autorisent le trafic sortant sur le port 443. Les points de terminaison du service de gestion Envoy public d'App Mesh suivent le format du nom de domaine complet (FQDN).

```
App Mesh Production Endpoint
appmesh-envoy-management.Region-code.amazonaws.com

App Mesh Preview Endpoint
appmesh-preview-envoy-management.Region-code.amazonaws.com
```

Vous pouvez déboguer votre connexion à EMS à l'aide de la commande ci-dessous. Cela envoie une demande gRPC valide mais vide au service de gestion Envoy.

```
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
```

Si vous recevez ces messages en retour, votre connexion à Envoy Management Service est fonctionnelle. Pour le débogage des erreurs liées au gRPC, consultez les erreurs [dans Envoy déconnecté du service de gestion App Mesh Envoy avec un texte](#) d'erreur.

```
grpc-status: 16
grpc-message: Missing Authentication Token
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Envoy s'est déconnecté du service de gestion App Mesh Envoy avec un texte d'erreur

### Symptômes

Votre proxy Envoy ne parvient pas à se connecter au service de gestion App Mesh Envoy et à recevoir sa configuration. Les journaux de votre proxy Envoy contiennent une entrée de journal comme celle-ci.

```
gRPC config stream closed: gRPC status code, message
```

## Résolution

Dans la plupart des cas, le message du journal doit indiquer le problème. Le tableau suivant répertorie les codes d'état gRPC les plus courants que vous pouvez voir, leurs causes et leurs résolutions.

| Code d'état gRPC | Cause                                                                                                                                               | Résolution                                                                                                                                                                                                                                                                                                          |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                | Déconnectez-vous gracieusement du service de gestion Envoy.                                                                                         | Il n'y a aucun problème. App Mesh déconnecte parfois les proxys Envoy avec ce code d'état. Envoy se reconnectera et continuera à recevoir des mises à jour.                                                                                                                                                         |
| 3                | Le point de terminaison du maillage (nœud virtuel ou passerelle virtuelle), ou l'une de ses ressources associées, est introuvable.                  | Vérifiez votre configuration Envoy pour vous assurer qu'elle porte le nom approprié de la ressource App Mesh qu'elle représente. Si votre ressource App Mesh est intégrée à d'autres AWS ressources, telles que des AWS Cloud Map espaces de noms ou des certificats ACM, assurez-vous que ces ressources existent. |
| 7                | Le proxy Envoy n'est pas autorisé à effectuer une action, telle que se connecter au service de gestion Envoy ou récupérer les ressources associées. | Assurez-vous de <a href="#">créer une politique IAM contenant les déclarations de politique</a> appropriées pour App Mesh et les autres services et d'associer cette politique à l'utilisateur ou au rôle IAM que votre proxy Envoy utilise pour                                                                    |



| Code d'état gRPC | Cause                                                                                                         | Résolution                                                                                                                                                            |
|------------------|---------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  |                                                                                                               | se connecter au service de gestion Envoy.                                                                                                                             |
| 8                | Le nombre de proxys Envoy pour une ressource App Mesh donnée dépasse le quota de service au niveau du compte. | Consultez <a href="#">App Mesh</a> pour plus d'informations sur les quotas de compte par défaut et sur la procédure à suivre pour demander une augmentation de quota. |

| Code d'état gRPC | Cause                                                                             | Résolution                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16               | Le proxy Envoy ne dispose pas d'informations d'authentification valides pour AWS. | Assurez-vous que l'Envoy dispose des informations d'identification appropriées pour se connecter aux AWS services via un utilisateur ou un rôle IAM. Un problème connu, <a href="#">#24136</a> , dans Envoy pour les versions v1.24 précédentes ne parvient pas à récupérer les informations d'identification si le processus Envoy utilise des descripteurs de 1024 fichiers supplémentaires. Cela se produit lorsque Envoy dessert un volume de trafic élevé. Vous pouvez confirmer ce problème en vérifiant la présence du texte « A libcurl function was given a bad argument » dans les journaux Envoy au niveau du débogage. Pour atténuer ce problème, passez à la version Envoy v1.25.1.0-prod ou à une version ultérieure. |

Vous pouvez consulter les codes d'état et les messages de votre proxy Envoy avec [Amazon CloudWatch Insights](#) à l'aide de la requête suivante :

```
filter @message like /gRPC config stream closed/
| parse @message "gRPC config stream closed: *, *" as StatusCode, Message
```

Si le message d'erreur fourni ne vous a pas aidé ou si votre problème n'est toujours pas résolu, envisagez d'ouvrir un [GitHub problème](#).

## Échec de la vérification de l'état du conteneur Envoy, de la sonde de disponibilité ou de la sonde de vivacité

### Symptômes

Votre proxy Envoy échoue aux contrôles de santé d'une tâche Amazon ECS, d'une instance Amazon EC2 ou d'un pod Kubernetes. Par exemple, vous interrogez l'interface d'administration d'Envoy à l'aide de la commande suivante et vous recevez un statut autre que LIVE.

```
curl -s http://my-app.default.svc.cluster.local:9901/server_info | jq '.state'
```

### Résolution

Voici une liste des étapes de correction en fonction de l'état renvoyé par le proxy Envoy.

- **PRE\_INITIALIZING** ou **INITIALIZING** — Le proxy Envoy n'a pas encore reçu de configuration ou ne peut pas se connecter et récupérer la configuration depuis le service de gestion App Mesh Envoy. L'Envoy reçoit peut-être une erreur du service de gestion Envoy lorsqu'il tente de se connecter. Pour plus d'informations, consultez les erreurs dans [Envoy s'est déconnecté du service de gestion App Mesh Envoy avec un texte d'erreur](#).
- **DRAINING** — Le proxy Envoy a commencé à épuiser les connexions en réponse à une `/drain_listeners` demande `/healthcheck/fail` ou sur l'interface d'administration d'Envoy. Nous vous déconseillons d'invoquer ces chemins sur l'interface d'administration, sauf si vous êtes sur le point de mettre fin à votre tâche Amazon ECS, à votre instance Amazon EC2 ou à votre pod Kubernetes.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Health : le contrôle de santé entre l'équilibreur de charge et le point de terminaison du maillage échoue

### Symptômes

Votre point de terminaison de maillage est considéré comme sain par le contrôle de l'état du conteneur ou la sonde de disponibilité, mais le contrôle de santé entre l'équilibreur de charge et le point de terminaison du maillage échoue.

## Résolution

Pour résoudre le problème, effectuez les tâches suivantes.

- Assurez-vous que le [groupe de sécurité](#) associé à votre point de terminaison maillé accepte le trafic entrant sur le port que vous avez configuré pour votre bilan de santé.
- Assurez-vous que le contrôle de santé aboutit systématiquement lorsqu'il est demandé manuellement, par exemple, depuis un [hôte bastion au sein de votre VPC](#).
- Si vous configurez un contrôle de santé pour un nœud virtuel, nous vous recommandons d'implémenter un point de terminaison de contrôle de santé dans votre application, par exemple, / ping pour HTTP. Cela garantit que le proxy Envoy et votre application sont routables depuis l'équilibreur de charge.
- Vous pouvez utiliser n'importe quel type d'équilibreur de charge élastique pour le nœud virtuel, en fonction des fonctionnalités dont vous avez besoin. Pour plus d'informations, consultez la section [Fonctionnalités d'Elastic Load Balancing](#).
- Si vous configurez un contrôle de santé pour une [passerelle virtuelle](#), nous vous recommandons d'utiliser un [équilibreur de charge réseau](#) avec un contrôle de santé TCP ou TLS sur le port d'écoute de la passerelle virtuelle. Cela garantit que l'écouteur de passerelle virtuelle est amorcé et prêt à accepter les connexions.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## La passerelle virtuelle n'accepte pas le trafic sur les ports 1024 ou moins

### Symptômes

Votre passerelle virtuelle n'accepte pas le trafic sur le port 1024 ou moins, mais accepte le trafic sur un numéro de port supérieur à 1024. Par exemple, vous interrogez les statistiques d'Envoy à l'aide de la commande suivante et vous recevez une valeur autre que zéro.

```
curl -s http://my-app.default.svc.cluster.local:9901/stats | grep "update_rejected"
```

Vous pouvez voir un texte similaire au texte suivant dans vos journaux décrivant un échec de liaison à un port privilégié :

```
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
```

## Résolution

Pour résoudre le problème, l'utilisateur spécifié pour la passerelle doit disposer de la fonctionnalité `LinuxCAP_NET_BIND_SERVICE`. Pour plus d'informations, consultez les [sections Fonctionnalités](#) dans le manuel du programmeur [Linux, Paramètres Linux dans les paramètres](#) de définition des tâches ECS et [Définir les capacités d'un conteneur](#) dans la documentation de Kubernetes.

### Important

Fargate doit utiliser une valeur de port supérieure à 1024.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Résolution des problèmes de connectivité App Mesh

Cette rubrique décrit les problèmes courants que vous pouvez rencontrer avec la connectivité App Mesh.

### Impossible de résoudre le nom DNS d'un service virtuel

#### Symptômes

Votre application ne parvient pas à résoudre le nom DNS d'un service virtuel auquel elle tente de se connecter.

#### Résolution

Il s'agit d'un problème connu. Pour plus d'informations, consultez le problème [Name VirtualServices by any hostname or FQDN](#) GitHub . Les services virtuels dans App Mesh peuvent porter n'importe quel nom. Tant qu'il existe un A enregistrement DNS pour le nom du service virtuel et que l'application peut résoudre le nom du service virtuel, la demande sera transmise par proxy par Envoy

et acheminée vers la destination appropriée. Pour résoudre le problème, ajoutez un A enregistrement DNS à toute adresse IP sans boucle, par exemple pour le 10.10.10.10 nom du service virtuel. L'Aenregistrement DNS peut être ajouté dans les conditions suivantes :

- Dans Amazon Route 53, si le nom est suffixé par le nom de votre zone hébergée privée
- Dans le `/etc/hosts` fichier du conteneur de l'application
- Dans un serveur DNS tiers que vous gérez

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Impossible de se connecter à un backend de service virtuel

### Symptômes

Votre application n'est pas en mesure d'établir une connexion à un service virtuel défini comme un backend sur votre nœud virtuel. Lorsque vous essayez d'établir une connexion, celle-ci peut échouer complètement ou la demande, du point de vue de l'application, peut échouer avec un code de HTTP 503 réponse.

### Résolution

Si l'application ne parvient pas du tout à se connecter (aucun code de HTTP 503 réponse n'est renvoyé), procédez comme suit :

- Assurez-vous que votre environnement informatique a été configuré pour fonctionner avec App Mesh.
  - Pour Amazon ECS, assurez-vous que la [configuration de proxy](#) appropriée est activée. Pour une end-to-end présentation détaillée, consultez [Getting Started with App Mesh et Amazon ECS](#).
  - Pour Kubernetes, y compris Amazon EKS, assurez-vous que le dernier contrôleur App Mesh est installé via Helm. Pour plus d'informations, consultez [App Mesh Controller](#) sur Helm Hub ou [Tutoriel : Configurer l'intégration d'App Mesh avec Kubernetes](#).
  - Pour Amazon EC2, assurez-vous d'avoir configuré votre instance Amazon EC2 pour transmettre le trafic App Mesh par proxy. Pour plus d'informations, consultez la section [Services de mise à jour](#).
- Assurez-vous que le conteneur Envoy qui s'exécute sur votre service informatique s'est correctement connecté au service de gestion App Mesh Envoy. Vous pouvez le confirmer en

vérifiant les statistiques d'Envoy pour le champ `control_plane.connected_state`. Pour plus d'informations `control_plane.connected_state`, consultez la section [Surveiller la connectivité du proxy Envoy](#) dans nos meilleures pratiques de dépannage.

Si l'Envoy a réussi à établir la connexion initialement, mais qu'il a ensuite été déconnecté et n'a jamais été reconnecté, voir [Envoy s'est déconnecté du service de gestion App Mesh Envoy avec un texte d'erreur](#) pour savoir pourquoi il a été déconnecté.

Si l'application se connecte mais que la demande échoue avec un code de HTTP 503 réponse, essayez ce qui suit :

- Assurez-vous que le service virtuel auquel vous vous connectez existe dans le maillage.
- Assurez-vous que le service virtuel dispose d'un fournisseur (routeur virtuel ou nœud virtuel).
- Lorsque vous utilisez Envoy comme proxy HTTP, si vous voyez du trafic sortant arriver au `cluster.cds_egress*_mesh-allow-all` lieu de la bonne destination via les statistiques d'Envoy, il est fort probable qu'Envoy n'achemine pas correctement les demandes. `filter_chains` Cela peut être dû à l'utilisation d'un nom de service virtuel non qualifié. Nous vous recommandons d'utiliser le nom de découverte du service réel comme nom du service virtuel, car le proxy Envoy communique avec les autres services virtuels par le biais de leurs noms.

Pour plus d'informations, consultez la section [Services virtuels](#).

- Consultez les journaux du proxy Envoy pour détecter l'un des messages d'erreur suivants :
  - `No healthy upstream`— Le nœud virtuel vers lequel le proxy Envoy tente de se diriger ne possède aucun point de terminaison résolu ou aucun point de terminaison sain. Assurez-vous que le nœud virtuel cible possède les paramètres de découverte des services et de vérification de l'état corrects.

Si les demandes adressées au service échouent lors du déploiement ou du dimensionnement du service virtuel principal, suivez les instructions figurant dans [Certaines demandes échouent avec le code d'état HTTP 503 lorsqu'un service virtuel dispose d'un fournisseur de nœuds virtuels](#).

- `No cluster match for URL`— Cela se produit très probablement lorsqu'une demande est envoyée à un service virtuel qui ne correspond aux critères définis par aucune des routes définies par un fournisseur de routeur virtuel. Assurez-vous que les demandes de l'application sont envoyées vers une route prise en charge en vérifiant que le chemin et les en-têtes de requête HTTP sont corrects.

- `No matching filter chain found`— Cela est probablement dû au fait qu'une demande est envoyée à un service virtuel sur un port non valide. Assurez-vous que les demandes provenant de l'application utilisent le même port que celui spécifié sur le routeur virtuel.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Impossible de se connecter à un service externe

### Symptômes

Votre application ne parvient pas à se connecter à un service en dehors du maillage, tel que `amazon.com`.

### Résolution

Par défaut, App Mesh n'autorise pas le trafic sortant des applications situées dans le maillage vers une destination en dehors du maillage. Pour activer la communication avec un service externe, deux options s'offrent à vous :

- Définissez le [filtre sortant](#) de la ressource de maillage sur `ALLOW_ALL`. Ce paramètre permet à n'importe quelle application au sein du maillage de communiquer avec n'importe quelle adresse IP de destination à l'intérieur ou à l'extérieur du maillage.
- Modélisez le service externe dans le maillage à l'aide d'un service virtuel, d'un routeur virtuel, d'une route et d'un nœud virtuel. Par exemple, pour modéliser le service externe `example.com`, vous pouvez créer un service virtuel nommé `example.com` avec un routeur et une route virtuels qui envoient tout le trafic vers un nœud virtuel dont le nom d'hôte de découverte du `example.com` service DNS est.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Impossible de se connecter à un serveur MySQL ou SMTP

### Symptômes

Lorsque vous autorisez le trafic sortant vers toutes les destinations (`Mesh EgressFilter type =ALLOW_ALL`), telles qu'un serveur SMTP ou une base de données MySQL à l'aide d'une définition



de nœud virtuel, la connexion depuis votre application échoue. À titre d'exemple, le message d'erreur suivant provient d'une tentative de connexion à un serveur MySQL.

```
ERROR 2013 (HY000): Lost connection to MySQL server at 'reading initial communication packet', system error: 0
```

## Résolution

Il s'agit d'un problème connu qui est résolu à l'aide de la version 1.15.0 ou ultérieure de l'image App Mesh. Pour plus d'informations, consultez le GitHub problème [Impossible de se connecter à MySQL avec App Mesh](#). Cette erreur se produit car l'écouteur sortant dans Envoy configuré par App Mesh ajoute le filtre d'écouteur Envoy TLS Inspector. Pour plus d'informations, consultez [TLS Inspector](#) dans la documentation d'Envoy. Ce filtre évalue si une connexion utilise le protocole TLS en inspectant le premier paquet envoyé par le client. Cependant, avec MySQL et SMTP, le serveur envoie le premier paquet après la connexion. Pour plus d'informations sur MySQL, consultez [Initial Handshake](#) dans la documentation MySQL. Comme le serveur envoie le premier paquet, l'inspection du filtre échoue.

Pour contourner ce problème en fonction de votre version d'Envoy :

- Si la version d'Envoy de votre image App Mesh est 1.15.0 ou ultérieure, ne modélisez pas de services externes tels que MySQL, SMTP, MSSQL, etc. comme backend pour le nœud virtuel de votre application.
- Si la version d'Envoy de votre image App Mesh est antérieure à la version 1.15.0, ajoutez le port **3306** à la liste des valeurs de vos services pour MySQL et **APPMESH\_EGRESS\_IGNORED\_PORTS** en tant que port que vous utilisez pour le protocole SMTP.

### Important

Bien que les ports SMTP standard soient 25587, et465, vous ne devez ajouter que le port que vous utilisez APPMESH\_EGRESS\_IGNORED\_PORTS et non les trois.

Pour plus d'informations, consultez [Services de mise à jour](#) pour Kubernetes, Services de [mise à jour pour](#) Amazon ECS ou [Services de mise à jour pour Amazon EC2](#).

Si votre problème n'est toujours pas résolu, vous pouvez nous fournir des informations sur ce que vous rencontrez en lien avec le [GitHub problème](#) existant ou contacter le [AWSSupport](#).

# Impossible de se connecter à un service modélisé comme un nœud virtuel TCP ou un routeur virtuel dans App Mesh

## Symptômes

Votre application ne parvient pas à se connecter à un backend qui utilise le paramètre du protocole TCP dans la définition de l'App Mesh [PortMapping](#).

## Résolution

Il s'agit d'un problème connu. Pour plus d'informations, consultez la section [Routage vers plusieurs destinations TCP sur le même port activé](#). GitHub App Mesh n'autorise actuellement pas plusieurs destinations de backend modélisées en TCP à partager le même port en raison des restrictions relatives aux informations fournies au proxy Envoy au niveau de la couche 4 de l'OSI. Pour vous assurer que le trafic TCP peut être acheminé de manière appropriée pour toutes les destinations principales, procédez comme suit :

- Assurez-vous que toutes les destinations utilisent un port unique. Si vous utilisez un fournisseur de routeur virtuel pour le service virtuel principal, vous pouvez modifier le port du routeur virtuel sans modifier le port des nœuds virtuels vers lesquels il est acheminé. Cela permet aux applications d'ouvrir des connexions sur le port du routeur virtuel tandis que le proxy Envoy continue d'utiliser le port défini dans le nœud virtuel.
- Si la destination modélisée en TCP est un serveur MySQL ou tout autre protocole basé sur le protocole TCP dans lequel le serveur envoie les premiers paquets après la connexion, consultez [Impossible de se connecter à un serveur MySQL ou SMTP](#)

Si votre problème n'est toujours pas résolu, vous pouvez nous fournir des informations sur ce que vous rencontrez en lien avec le [GitHub problème](#) existant ou contacter le [AWSSupport](#).

# La connectivité réussit à ce que le service ne soit pas répertorié en tant que backend de service virtuel pour un nœud virtuel

## Symptômes

Votre application est capable de se connecter et d'envoyer du trafic vers une destination qui n'est pas spécifiée en tant que backend de service virtuel sur votre nœud virtuel.

## Résolution

Si les demandes aboutissent à une destination qui n'a pas été modélisée dans les API App Mesh, la cause la plus probable est que le type de [filtre sortant](#) du maillage a été défini sur `ALLOW_ALL`. Lorsque le filtre sortant est défini sur `ALLOW_ALL`, une demande sortante provenant de votre application qui ne correspond pas à une destination modélisée (backend) sera envoyée à l'adresse IP de destination définie par l'application.

Si vous souhaitez interdire le trafic vers des destinations non modélisées dans le maillage, pensez à définir la valeur du filtre sortant sur `DROP_ALL`.

#### Note

La définition de la valeur du filtre sortant du maillage affecte tous les nœuds virtuels du maillage.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Certaines demandes échouent avec le code d'état HTTP **503** lorsqu'un service virtuel dispose d'un fournisseur de nœuds virtuels

### Symptômes

Une partie des demandes de votre application ne parviennent pas à un backend de service virtuel qui utilise un fournisseur de nœuds virtuels au lieu d'un fournisseur de routeur virtuel. Lorsque vous utilisez un fournisseur de routeur virtuel pour le service virtuel, les demandes n'échouent pas.

### Résolution

Il s'agit d'un problème connu. Pour plus d'informations, voir [Politique de nouvelle tentative sur le fournisseur de nœuds virtuels pour un service virtuel activé](#) GitHub. Lorsque vous utilisez un nœud virtuel en tant que fournisseur d'un service virtuel, vous ne pouvez pas spécifier la politique de nouvelle tentative par défaut que vous souhaitez que les clients de votre service virtuel utilisent. En comparaison, les fournisseurs de routeurs virtuels autorisent la spécification de politiques de nouvelle tentative, car elles sont une propriété des ressources de la route enfant.

Pour réduire le nombre d'échecs de demandes adressées aux fournisseurs de nœuds virtuels, utilisez plutôt un fournisseur de routeur virtuel et spécifiez une politique de nouvelle tentative sur ses itinéraires. Pour d'autres moyens de réduire le nombre d'échecs de requêtes adressées à vos applications, consultez [Meilleures pratiques en matière d'App Mesh](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Impossible de se connecter à un système de fichiers Amazon EFS

### Symptômes

Lors de la configuration d'une tâche Amazon ECS avec un système de fichiers Amazon EFS en tant que volume, la tâche ne démarre pas avec l'erreur suivante.

```
ResourceInitializationError: failed to invoke EFS utils commands to set up EFS volumes:
stderr: mount.nfs4: Connection refused : unsuccessful EFS utils command execution;
code: 32
```

### Résolution

Il s'agit d'un problème connu. Cette erreur se produit car la connexion NFS à Amazon EFS est établie avant le démarrage des conteneurs de votre tâche. Ce trafic est acheminé par la configuration du proxy vers Envoy, qui ne sera pas exécuté à ce stade. En raison de l'ordre de démarrage, le client NFS ne parvient pas à se connecter au système de fichiers Amazon EFS et la tâche ne démarre pas. Pour résoudre le problème, ajoutez le port 2049 à la liste de valeurs pour le `EgressIgnoredPorts` paramètre dans la configuration proxy de votre définition de tâche Amazon ECS. Pour plus d'informations, consultez la section [Configuration du proxy](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## La connectivité réussit à assurer le service, mais la demande entrante n'apparaît pas dans les journaux d'accès, les traces ou les métriques d'Envoy

### Symptômes

Même si votre application peut se connecter et envoyer des demandes à une autre application, vous ne pouvez pas voir les demandes entrantes dans les journaux d'accès ou dans les informations de suivi du proxy Envoy.

### Résolution

Il s'agit d'un problème connu. Pour plus d'informations, consultez le problème de [configuration des règles iptables](#) sur Github. Le proxy Envoy intercepte uniquement le trafic entrant vers le port écouté par le nœud virtuel correspondant. Les demandes adressées à tout autre port contourneront le proxy Envoy et atteindront directement le service qui le sous-tend. Pour permettre au proxy Envoy d'intercepter le trafic entrant pour votre service, vous devez configurer votre nœud virtuel et votre service pour qu'ils écoutent sur le même port.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## La définition des variables d'`HTTPS_PROXY` environnement `HTTP_PROXY`/au niveau du conteneur ne fonctionne pas comme prévu.

### Symptômes

Lorsque `HTTP_PROXY/HTTPS_PROXY` est défini comme variable d'environnement dans :

- Conteneur d'applications dans la définition des tâches avec App Mesh activé, les demandes envoyées à l'espace de noms des services App Mesh recevront des réponses HTTP 500 d'erreur de la part du sidecar Envoy.
- Conteneur Envoy dans la définition des tâches avec App Mesh activé, les demandes provenant du sidecar Envoy ne passeront pas par le serveur HTTPS proxy HTTP/et la variable d'environnement ne fonctionnera pas.

### Résolution

Pour le conteneur d'applications :

App Mesh fonctionne en faisant passer le trafic au sein de votre tâche par le proxy Envoy. `HTTP_PROXY/HTTPS_PROXY` configuration remplace ce comportement en configurant le trafic de conteneurs pour qu'il passe par un autre proxy externe. Le trafic sera toujours intercepté par Envoy, mais il ne prend pas en charge le transfert par proxy du trafic maillé à l'aide d'un proxy externe.

Si vous souhaitez utiliser un proxy pour tout le trafic non maillé, configurez `NO_PROXY` pour inclure le CIDR/namespaces de votre maillage, localhost et les points de terminaison des informations d'identification, comme dans l'exemple suivant.

```
NO_PROXY=localhost,127.0.0.1,169.254.169.254,169.254.170.2,10.0.0.0/16
```

Pour le conteneur Envoy :

Envoy ne prend pas en charge les proxys génériques. Il est déconseillé de définir ces variables.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Expiration des demandes en amont, même après avoir défini le délai d'expiration pour les itinéraires.

### Symptômes

Vous avez défini le délai d'expiration pour :

- Les itinéraires, mais vous recevez toujours une erreur de délai d'attente de la demande en amont.
- L'écouteur du nœud virtuel, le délai d'expiration et le délai de nouvelle tentative pour les itinéraires, mais vous obtenez toujours une erreur de délai d'expiration de la demande en amont.

### Résolution

Pour que les demandes à latence élevée supérieure à 15 secondes soient traitées avec succès, vous devez spécifier un délai d'attente au niveau de la route et au niveau de l'écouteur du nœud virtuel.

Si vous spécifiez un délai d'attente d'itinéraire supérieur à la valeur par défaut de 15 secondes, assurez-vous que le délai d'expiration est également spécifié pour l'écouteur pour tous les nœuds virtuels participants. Toutefois, si vous réduisez le délai d'expiration à une valeur inférieure à la valeur par défaut, il est facultatif de mettre à jour le délai d'expiration au niveau des nœuds virtuels. Pour plus d'informations sur les options de configuration de nœuds et de routes virtuels, consultez la section [Nœuds et itinéraires virtuels](#).

Si vous avez défini une politique de nouvelles tentatives, la durée que vous spécifiez pour le délai d'expiration des demandes doit toujours être supérieure ou égale au délai de nouvelles tentatives multiplié par le nombre maximum de tentatives que vous avez défini dans la politique de nouvelles tentatives. Cela permet à votre demande, avec toutes les nouvelles tentatives, de se terminer avec succès. Pour plus d'informations, consultez la section [itinéraires](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Envoy répond par une requête HTTP Bad.

### Symptômes

Envoy répond par une requête HTTP 400 Bad pour toutes les demandes envoyées via le Network Load Balancer (NLB). Lorsque nous consultons les journaux d'Envoy, nous constatons que :

- Journaux de débogage :

```
dispatch error: http/1.1 protocol error: HPE_INVALID_METHOD
```

- Journaux d'accès :

```
"- - HTTP/1.1" 400 DPE 0 11 0 - "-" "-" "-" "-" "-"
```

### Résolution

La solution consiste à désactiver le protocole proxy version 2 (PPv2) sur les attributs du [groupe cible](#) de votre NLB.

À ce jour, le PPv2 n'est pas pris en charge par la passerelle virtuelle et le nœud virtuel Envoy qui sont exécutés à l'aide du plan de contrôle App Mesh. Si vous déployez NLB à l'aide d'un contrôleur d'équilibrage de AWS charge sur Kubernetes, désactivez PPv2 en définissant l'attribut suivant sur : `false`

```
service.beta.kubernetes.io/aws-load-balancer-target-group-attributes:
proxy_protocol_v2.enabled
```

Consultez les [annotations du AWS Load Balancer Controller](#) pour plus de détails sur les attributs des ressources NLB.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Impossible de configurer correctement le délai d'expiration.

### Symptômes

Votre demande expire dans les 15 secondes, même après avoir configuré le délai d'attente sur l'écouteur du nœud virtuel et le délai d'attente sur la route vers le backend du nœud virtuel.

## Résolution

Assurez-vous que le service virtuel approprié est inclus dans la liste des backends.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Résolution des problèmes liés au dimensionnement de l'App Mesh

Cette rubrique décrit les problèmes courants que vous pouvez rencontrer lors de la mise à l'échelle d'App Mesh.

### La connectivité échoue et les vérifications de l'état du conteneur échouent lorsque le dimensionnement dépasse les 50 répliques pour un nœud virtuel/une passerelle virtuelle

#### Symptômes

Lorsque vous augmentez le nombre de répliques, telles que les tâches Amazon ECS, les pods Kubernetes ou les instances Amazon EC2, pour un nœud virtuel/une passerelle virtuelle au-delà de 50, les contrôles de santé des conteneurs Envoy pour les nouveaux Envoy ou ceux en cours d'exécution commencent à échouer. Les applications en aval qui envoient du trafic vers le nœud virtuel/la passerelle virtuelle commencent à rencontrer des échecs de requête avec le code d'état HTTP. 503

#### Résolution

Le quota par défaut d'App Mesh pour le nombre d'envoyés par nœud virtuel/passerelle virtuelle est de 50. Lorsque le nombre d'envoyés en cours d'exécution dépasse ce quota, les nouveaux envoyés et ceux en cours d'exécution ne parviennent pas à se connecter au service de gestion Envoy d'App Mesh avec le code d'état gRPC (`RESOURCE_EXHAUSTED`). Ce quota peut être augmenté. Pour plus d'informations, consultez [App Mesh](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

### Les demandes échouent **503** lorsqu'un backend de service virtuel évolue horizontalement vers l'extérieur ou vers l'intérieur

#### Symptômes



Lorsqu'un service virtuel principal est étendu ou intégré horizontalement, les demandes des applications en aval échouent avec un code d'HTTP 503 état.

## Résolution

App Mesh recommande plusieurs approches pour atténuer les cas de défaillance tout en dimensionnant les applications horizontalement. Pour obtenir des informations détaillées sur la manière de prévenir ces défaillances, consultez [Meilleures pratiques en matière d'App Mesh](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Le conteneur Envoy se bloque avec segfault en cas de charge accrue

### Symptômes

En cas de charge de trafic élevée, le proxy Envoy se bloque en raison d'une erreur de segmentation (code de sortie Linux 139). Les journaux de processus d'Envoy contiennent une déclaration comme celle-ci.

```
Caught Segmentation fault, suspect faulting address 0x0"
```

### Résolution

Le proxy Envoy a probablement dépassé la limite `nofile ulimit` par défaut du système d'exploitation, la limite du nombre de fichiers qu'un processus peut ouvrir à la fois. Cette violation est due au fait que le trafic entraîne l'augmentation du nombre de connexions, qui consomment des sockets supplémentaires du système d'exploitation. Pour résoudre ce problème, augmentez la valeur `ulimit nofile` sur le système d'exploitation hôte. Si vous utilisez Amazon ECS, cette limite peut être modifiée via les [paramètres Ulimit](#) des [limites de ressources](#) de la définition de tâche.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## L'augmentation des ressources par défaut n'est pas reflétée dans les limites de service

### Symptômes

Après avoir augmenté la limite par défaut des ressources App Mesh, la nouvelle valeur n'est pas prise en compte lorsque vous examinez vos limites de service.

## Résolution

Bien que les nouvelles limites ne soient pas affichées actuellement, les clients peuvent toujours les appliquer.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## L'application se bloque en raison d'un grand nombre d'appels de bilan de santé.

### Symptômes

Après avoir activé les contrôles de santé actifs pour un nœud virtuel, le nombre d'appels de contrôle de santé augmente. L'application se bloque en raison de l'augmentation considérable du nombre d'appels de vérification de santé effectués vers l'application.

### Résolution

Lorsque le contrôle de santé actif est activé, chaque point de terminaison Envoy du terminal en aval (client) envoie des demandes d'état à chaque point de terminaison du cluster en amont (serveur) afin de prendre des décisions de routage. Par conséquent, le nombre total de demandes de bilan de santé serait de `number of client Envoys * number of server Envoys * active health check frequency`.

Pour résoudre ce problème, modifiez la fréquence de la sonde de contrôle de santé, ce qui réduirait le volume total des sondes de contrôle de santé. Outre les bilans de santé actifs, App Mesh permet de configurer la [détection des valeurs aberrantes](#) comme moyen de vérification passive de l'état de santé. Utilisez la détection des valeurs aberrantes pour configurer le moment où il convient de supprimer un hôte spécifique en fonction 5xx des réponses consécutives.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Résolution des problèmes liés à l'observabilité de l'App Mesh

Cette rubrique décrit les problèmes courants que vous pouvez rencontrer avec l'observabilité d'App Mesh.

# Impossible de voir les AWS X-Ray traces de mes applications

## Symptômes

Votre application dans App Mesh n'affiche pas les informations de suivi X-Ray dans la console X-Ray ni dans les API.

## Résolution

Pour utiliser X-Ray dans App Mesh, vous devez configurer correctement les composants afin de permettre la communication entre votre application, les conteneurs annexes et le service X-Ray. Suivez les étapes suivantes pour vérifier que X-Ray a été correctement configuré :

- Assurez-vous que le protocole d'écoute App Mesh Virtual Node n'est pas défini comme TCP.
- Assurez-vous que le conteneur X-Ray déployé avec votre application expose le port UDP 2000 et s'exécute en tant qu'utilisateur. 1337 Pour plus d'informations, consultez l'[exemple d'Amazon ECS X-Ray](#) sur GitHub.
- Assurez-vous que le suivi est activé sur le conteneur Envoy. Si vous utilisez l'[image App Mesh Envoy](#), vous pouvez activer X-Ray en définissant la variable d'ENVIRONNEMENT `ENABLE_ENVOY_XRAY_TRACING` sur une valeur de 1 et la variable d'ENVIRONNEMENT `XRAY_DAEMON_PORT` sur 2000.
- Si vous avez instrumenté X-Ray dans le code de votre application avec l'un des [SDK spécifiques au langage](#), assurez-vous qu'il est correctement configuré en suivant les guides correspondant à votre langue.
- Si tous les éléments précédents sont correctement configurés, consultez les journaux des conteneurs X-Ray pour détecter les erreurs et suivez les instructions de la section [Dépannage AWS X-Ray](#). Vous trouverez une explication plus détaillée de l'intégration de X-Ray dans App Mesh dans [Integrating X-Ray with App Mesh](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

# Impossible de voir les métriques Envoy pour mes applications dans CloudWatch les métriques Amazon

## Symptômes

Votre application dans App Mesh n'émet pas de métriques générées par le proxy Envoy vers CloudWatch des métriques.

## Résolution

Lorsque vous utilisez CloudWatch des métriques dans App Mesh, vous devez configurer correctement plusieurs composants pour permettre la communication entre votre proxy Envoy, le sidecar de l' CloudWatch agent et le service de CloudWatch métriques. Suivez les étapes suivantes pour vérifier que CloudWatch les métriques du proxy Envoy ont été correctement configurées :

- Assurez-vous que vous utilisez l'image de l' CloudWatch agent pour App Mesh. Pour plus d'informations, consultez la section [CloudWatchAgent App Mesh](#) activé GitHub.
- Assurez-vous d'avoir correctement configuré l' CloudWatch agent pour App Mesh en suivant les instructions d'utilisation spécifiques à la plate-forme. Pour plus d'informations, consultez la section [CloudWatchAgent App Mesh](#) activé GitHub.
- Si tous les éléments précédents sont correctement configurés, consultez les journaux du conteneur de l' CloudWatch agent pour détecter les erreurs et suivez les instructions fournies dans la section [Dépannage de l' CloudWatch agent](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Impossible de configurer des règles d'échantillonnage personnalisées pour les AWS X-Ray traces

### Symptômes

Votre application utilise le traçage X-Ray, mais vous ne parvenez pas à configurer les règles d'échantillonnage pour vos traces.

### Résolution

Étant donné qu'App Mesh Envoy ne prend actuellement pas en charge la configuration d'échantillonnage Dynamic X-Ray, les solutions de contournement suivantes sont disponibles.

Si votre version d'Envoy est 1.19.1 ou ultérieure, les options suivantes s'offrent à vous.

- Pour définir uniquement le taux d'échantillonnage, utilisez la variable d'XRAY\_SAMPLING\_RATE environnement sur le conteneur Envoy. La valeur doit être spécifiée

sous forme décimale entre 0 et 1.00 (100 %). Pour plus d'informations, consultez [AWS X-Ray variables](#).

- Pour configurer les règles d'échantillonnage personnalisées localisées pour le traceur X-Ray, utilisez la variable d'XRAY\_SAMPLING\_RULE\_MANIFESTenvironnement pour spécifier un chemin de fichier dans le système de fichiers du conteneur Envoy. Pour plus d'informations, consultez la section [Règles d'échantillonnage](#) dans le Guide du AWS X-Ray développeur.

Si votre version d'Envoy est antérieure à 1.19.1, procédez comme suit.

- Utilisez la variable d'ENVOY\_TRACING\_CFG\_FILEenvironnement pour modifier votre taux d'échantillonnage. Pour plus d'informations, consultez [Variables de configuration Envoy](#). Spécifiez une configuration de suivi personnalisée et définissez des règles d'échantillonnage locales. Pour plus d'informations, consultez la section [Configuration d'Envoy X-Ray](#).
- Exemple de configuration de suivi personnalisée pour la variable d'ENVOY\_TRACING\_CFG\_FILEenvironnement :

```
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
```

- Pour plus de détails sur la configuration du manifeste des règles d'échantillonnage dans la `samplingRuleManifest` propriété, consultez [Configuring the X-Ray SDK for Go](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Résolution des problèmes de sécurité liés à App Mesh

Cette rubrique décrit les problèmes courants que vous pouvez rencontrer avec la sécurité d'App Mesh.

### Impossible de se connecter à un service virtuel principal avec une politique client TLS

#### Symptômes

Lorsque vous ajoutez une politique client TLS à un backend de service virtuel dans un nœud virtuel, la connectivité à ce backend échoue. Lorsque vous tentez d'envoyer du trafic vers le service principal, les demandes échouent avec un code de HTTP 503 réponse et le message d'erreur :`upstream connect error or disconnect/reset before headers. reset reason: connection failure`.

#### Résolution

Afin de déterminer la cause première du problème, nous vous recommandons d'utiliser les journaux de processus du proxy Envoy pour vous aider à diagnostiquer le problème. Pour plus d'informations, consultez [Activer la journalisation du débogage d'Envoy dans les environnements de pré-production](#). Utilisez la liste suivante pour déterminer la cause de l'échec de connexion :

- Assurez-vous que la connectivité au backend fonctionne correctement en excluant les erreurs mentionnées dans. [Impossible de se connecter à un backend de service virtuel](#)
- Dans les journaux de processus d'Envoy, recherchez les erreurs suivantes (enregistrées au niveau du débogage).

```
TLS error: 268435581:SSL routines:OPENSSL_internal:CERTIFICATE_VERIFY_FAILED
```

Cette erreur est due à une ou plusieurs des raisons suivantes :

- Le certificat n'a pas été signé par l'une des autorités de certification définies dans le bundle de confiance relatif à la politique client TLS.
- Le certificat n'est plus valide (expiré).
- Le nom alternatif du sujet (SAN) ne correspond pas au nom d'hôte DNS demandé.

- Assurez-vous que le certificat proposé par le service principal est valide, qu'il est signé par l'une des autorités de certification de votre bundle de confiance pour les politiques client TLS et qu'il répond aux critères définis dans. [protocole TLS \(Transport Layer Security\)](#)
- Si l'erreur que vous recevez est similaire à celle ci-dessous, cela signifie que la demande contourne le proxy Envoy et atteint directement l'application. Lors de l'envoi de trafic, les statistiques sur Envoy ne changent pas, ce qui indique qu'Envoy n'est pas sur le point de déchiffrer le trafic. Dans la configuration du proxy du nœud virtuel, assurez-vous qu'il `AppPorts` contient la bonne valeur que l'application écoute.

```
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
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#) Si vous pensez avoir découvert une faille de sécurité ou si vous avez des questions concernant la sécurité d'App Mesh, consultez les [directives relatives au signalement des AWS vulnérabilités](#).

## Impossible de se connecter à un service virtuel principal lorsque l'application est à l'origine du protocole TLS

### Symptômes

Lorsque vous lancez une session TLS depuis une application, plutôt que depuis le proxy Envoy, la connectivité à un service virtuel principal échoue.

### Résolution

Il s'agit d'un problème connu. Pour plus d'informations, consultez le GitHub problème de [demande de fonctionnalité : négociation TLS entre l'application en aval et le proxy en amont](#). Dans App Mesh, l'origine TLS est actuellement prise en charge par le proxy Envoy, mais pas par l'application. Pour utiliser la prise en charge de l'origine TLS au niveau de l'Envoy, désactivez la prise en charge de l'origine TLS dans l'application. Cela permet à l'Envoy de lire les en-têtes des demandes sortantes et de transmettre la demande à la destination appropriée via une session TLS. Pour plus d'informations, consultez [protocole TLS \(Transport Layer Security\)](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#) Si vous pensez avoir découvert une faille de sécurité ou si vous avez des questions

concernant la sécurité d'App Mesh, consultez les [directives relatives au signalement des AWS vulnérabilités](#).

## Impossible d'affirmer que la connectivité entre les proxys Envoy utilise le protocole TLS

### Symptômes

Votre application a activé la terminaison TLS sur le nœud virtuel ou l'écouteur de passerelle virtuelle, ou l'origine du TLS sur la politique du client TLS principal, mais vous ne pouvez pas affirmer que la connectivité entre les proxys Envoy se produit au cours d'une session négociée par TLS.

### Résolution

Les étapes définies dans cette résolution utilisent l'interface d'administration d'Envoy et les statistiques d'Envoy. Pour obtenir de l'aide pour les configurer, reportez-vous [Activer l'interface d'administration du proxy Envoy](#) aux sections et [Activer l'intégration d'Envoy DogStats D pour le déchargement métrique](#). Les exemples de statistiques suivants utilisent l'interface d'administration pour des raisons de simplicité.

- Pour le proxy Envoy effectuant la terminaison du protocole TLS :
  - Assurez-vous que le certificat TLS a été amorcé dans la configuration d'Envoy à l'aide de la commande suivante.

```
curl http://my-app.default.svc.cluster.local:9901/certs
```

Dans le résultat renvoyé, vous devriez voir au moins une entrée sous `certificates[].cert_chain` pour le certificat utilisé pour la terminaison du protocole TLS.

- Assurez-vous que le nombre de connexions entrantes réussies avec l'écouteur du proxy est exactement le même que le nombre de poignées de main SSL plus le nombre de sessions SSL réutilisées, comme le montrent les exemples de commandes et de sorties suivants.

```
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)

```

- Pour le proxy Envoy effectuant la création TLS :
  - Assurez-vous que le magasin de confiance TLS a été amorcé dans la configuration Envoy à l'aide de la commande suivante.

```
curl http://my-app.default.svc.cluster.local:9901/certs
```

Vous devriez voir au moins une entrée ci-dessous pour les certificats utilisés `certificates[].ca_certs` pour valider le certificat du backend lors de la création du protocole TLS.

- Assurez-vous que le nombre de connexions sortantes réussies vers le cluster principal est exactement le même que le nombre de connexions SSL plus le nombre de sessions SSL réutilisées, comme le montrent les exemples de commandes et de résultats suivants.

```

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)

```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#) Si vous pensez avoir découvert une faille de sécurité ou si vous avez des questions concernant la sécurité d'App Mesh, consultez les [directives relatives au signalement des AWS vulnérabilités](#).

## Résolution des problèmes liés au protocole TLS avec Elastic Load Balancing

### Symptômes

Lorsque vous essayez de configurer un Application Load Balancer ou un Network Load Balancer pour chiffrer le trafic vers un nœud virtuel, les vérifications de connectivité et de santé de l'équilibreur de charge peuvent échouer.

### Résolution

Afin de déterminer la cause première du problème, vous devez vérifier les points suivants :

- Pour que le proxy Envoy effectue la terminaison du protocole TLS, vous devez exclure toute erreur de configuration. Suivez les étapes indiquées ci-dessus dans le [Impossible de se connecter à un service virtuel principal avec une politique client TLS](#).
- Pour l'équilibreur de charge, vous devez examiner la configuration du TargetGroup :
  - Assurez-vous que le TargetGroup port correspond au port d'écoute défini par le nœud virtuel.
  - Pour les équilibreurs de charge d'application qui créent des connexions TLS via HTTP vers votre service, assurez-vous que le TargetGroup protocole est défini sur HTTPS. Si des bilans de santé sont utilisés, assurez-vous qu'ils HealthCheckProtocol sont définis sur HTTPS.
  - Pour les équilibreurs de charge réseau qui créent des connexions TLS via TCP vers votre service, assurez-vous que le TargetGroup protocole est défini sur TLS. Si des bilans de santé sont utilisés, assurez-vous qu'ils HealthCheckProtocol sont définis sur TCP.

#### Note

Toute mise à jour TargetGroup nécessitant un changement de TargetGroup nom.

Une fois cette configuration correctement configurée, votre équilibreur de charge devrait fournir une connexion sécurisée à votre service à l'aide du certificat fourni au proxy Envoy.

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#) Si vous pensez avoir découvert une faille de sécurité ou si vous avez des questions concernant la sécurité d'App Mesh, consultez les [directives relatives au signalement des AWS vulnérabilités](#).

## Résolution des problèmes liés à App Mesh Kubernetes

Cette rubrique décrit les problèmes courants que vous pouvez rencontrer lorsque vous utilisez App Mesh avec Kubernetes.

### Les ressources App Mesh créées dans Kubernetes sont introuvables dans App Mesh

#### Symptômes

Vous avez créé les ressources App Mesh à l'aide de la définition de ressource personnalisée (CRD) de Kubernetes, mais les ressources que vous avez créées ne sont pas visibles dans App Mesh lorsque vous utilisez les API or. AWS Management Console

#### Résolution

La cause probable est une erreur dans le contrôleur Kubernetes pour App Mesh. Pour plus d'informations, consultez la section [Résolution des problèmes](#) sur GitHub. Vérifiez les journaux du contrôleur pour détecter toute erreur ou tout avertissement indiquant que le contrôleur n'a pas pu créer de ressources.

```
kubectl logs -n appmesh-system -f \
$(kubectl get pods -n appmesh-system -o name | grep controller)
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

### Les dosettes échouent aux contrôles de préparation et de vivacité après l'injection du sidecar Envoy

#### Symptômes

Les modules de votre application fonctionnaient correctement auparavant, mais une fois que le sidecar Envoy a été injecté dans un module, les contrôles de disponibilité et de vivacité commencent à échouer.

## Résolution

Assurez-vous que le conteneur Envoy injecté dans le pod a démarré avec le service de gestion Envoy d'App Mesh. Vous pouvez vérifier les erreurs éventuelles en référençant les codes d'erreur dans [Envoy s'est déconnecté du service de gestion App Mesh Envoy avec un texte d'erreur](#). Vous pouvez utiliser la commande suivante pour inspecter les journaux Envoy pour le pod concerné.

```
kubectl logs -n appmesh-system -f \
$(kubectl get pods -n appmesh-system -o name | grep controller) \
| grep "gRPC config stream closed"
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Les pods ne s'enregistrent pas ou ne se désenregistrent pas en tant qu'instances AWS Cloud Map

### Symptômes

Vos pods Kubernetes ne sont pas enregistrés ou désenregistrés dans le AWS Cloud Map cadre de leur cycle de vie. Un module peut démarrer correctement et être prêt à recevoir du trafic, mais ne pas en recevoir. Lorsqu'un pod est fermé, les clients peuvent toujours conserver son adresse IP et tenter d'y envoyer du trafic, sans succès.

### Résolution

Il s'agit d'un problème connu. Pour plus d'informations, consultez la section Les [pods ne sont pas enregistrés/déenregistrés automatiquement](#) dans Kubernetes en cas de problème. AWS Cloud Map GitHub En raison de la relation entre les pods, les nœuds virtuels App Mesh et les AWS Cloud Map ressources, le [contrôleur App Mesh pour Kubernetes peut se](#) désynchroniser et perdre des ressources. Par exemple, cela peut se produire si une ressource de nœud virtuel est supprimée de Kubernetes avant de mettre fin à ses pods associés.

Pour pallier ce problème :

- Assurez-vous que vous utilisez la dernière version du contrôleur App Mesh pour Kubernetes.

- Assurez-vous que les AWS Cloud Map `namespaceName` et `serviceName` sont corrects dans la définition de votre nœud virtuel.
- Assurez-vous de supprimer tous les pods associés avant de supprimer la définition de votre nœud virtuel. Si vous avez besoin d'aide pour identifier les pods associés à un nœud virtuel, consultez [Impossible de déterminer où s'exécute un pod pour une ressource App Mesh](#).
- Si le problème persiste, exécutez la commande suivante pour inspecter les journaux de votre manette afin de détecter les erreurs susceptibles de révéler le problème sous-jacent.

```
kubectl logs -n appmesh-system \
$(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

- Envisagez d'utiliser la commande suivante pour redémarrer vos modules de manette. Cela peut résoudre les problèmes de synchronisation.

```
kubectl delete -n appmesh-system \
$(kubectl get pods -n appmesh-system -o name | grep appmesh-controller)
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## Impossible de déterminer où s'exécute un pod pour une ressource App Mesh

### Symptômes

Lorsque vous exécutez App Mesh sur un cluster Kubernetes, un opérateur ne peut pas déterminer où s'exécute une charge de travail, ou un pod, pour une ressource App Mesh donnée.

### Résolution

Les ressources du pod Kubernetes sont annotées avec le maillage et le nœud virtuel auxquels elles sont associées. Vous pouvez demander quels pods sont en cours d'exécution pour un nom de nœud virtuel donné à l'aide de la commande suivante.

```
kubectl get pods --all-namespaces -o json | \
jq '.items[] | { metadata } | select(.metadata.annotations."appmesh.k8s.aws/
virtualNode" == "virtual-node-name")'
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Impossible de déterminer la ressource App Mesh sous laquelle un pod est exécuté

### Symptômes

Lors de l'exécution d'App Mesh sur un cluster Kubernetes, un opérateur ne peut pas déterminer la ressource App Mesh utilisée par un pod donné.

### Résolution

Les ressources du pod Kubernetes sont annotées avec le maillage et le nœud virtuel auxquels elles sont associées. Vous pouvez générer les noms du maillage et des nœuds virtuels en interrogeant directement le pod à l'aide de la commande suivante.

```
kubectl get pod pod-name -n namespace -o json | \
jq '{ "mesh": .metadata.annotations."appmesh.k8s.aws/mesh",
"virtualNode": .metadata.annotations."appmesh.k8s.aws/virtualNode" }'
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWS Support.](#)

## Les envoyés clients ne sont pas en mesure de communiquer avec le service de gestion App Mesh Envoy lorsque IMDSv1 est désactivé

### Symptômes

Lorsqu'elle IMDSv1 est désactivée, le client Envoy ne peut pas communiquer avec le plan de contrôle App Mesh (Envoy Management Service). IMDSv2 le support n'est pas disponible sur la version App Mesh Envoy auparavant `v1.24.0.0-prod`.

### Résolution

Pour résoudre ce problème, vous pouvez effectuer l'une des trois opérations suivantes.

- Passez à la version App Mesh Envoy `v1.24.0.0-prod` ou à une version ultérieure, IMDSv2 compatible.

- Réactivez-le IMDSv1 sur l'instance sur laquelle Envoy est exécuté. Pour obtenir des instructions sur la restaurationIMDSv1, consultez [Configurer les options de métadonnées de l'instance](#).
- Si vos services s'exécutent sur Amazon EKS, il est recommandé d'utiliser les rôles IAM pour les comptes de service (IRSA) pour récupérer les informations d'identification. Pour obtenir des instructions sur l'activation de l'IRSA, consultez la section [Rôles IAM pour les comptes de service](#).

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

## L'IRSA ne fonctionne pas sur le conteneur d'applications lorsque App Mesh est activé et qu'Envoy est injecté

### Symptômes

Lorsque App Mesh est activé sur un cluster Amazon EKS à l'aide du contrôleur App Mesh pour Amazon EKS, Envoy et les `proxyinit` conteneurs sont injectés dans le module d'application. L'application n'est pas en mesure IRSA de supposer que `node role`. Lorsque nous décrivons les détails du pod, nous constatons alors que la variable d'AWS\_ROLE\_ARN environnement `AWS_WEB_IDENTITY_TOKEN_FILE` ou n'est pas incluse dans le conteneur de l'application.

### Résolution

Si l'une `AWS_WEB_IDENTITY_TOKEN_FILE` ou `AWS_ROLE_ARN` l'autre des variables d'environnement sont définies, le webhook ignorera le pod. Ne fournissez aucune de ces variables et le webhook se chargera de vous les injecter.

```
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
 }
}
```

[Si votre problème n'est toujours pas résolu, pensez à en ouvrir un GitHub ou à contacter le AWSSupport.](#)

# App Mesh MeshVersion préliminaire

Le canal App Mesh Preview est une variante distincte du service App Mesh fourni dans la us-west-2 région. La chaîne de prévisualisation présente les fonctionnalités à venir que vous pourrez essayer au fur et à mesure de leur développement. Lorsque vous utilisez des fonctionnalités dans le canal de prévisualisation, vous pouvez fournir des commentaires GitHub pour influencer le comportement des fonctionnalités. Une fois qu'une fonctionnalité possède toutes les fonctionnalités du canal de prévisualisation et que toutes les intégrations et vérifications nécessaires ont été effectuées, elle passe au service App Mesh de production.

La chaîne deAWS App Mesh prévisualisation est un service bêta et toutes les fonctionnalités sont des aperçus, tels que ces termes sont définis dans les [conditions d'AWSUtilisation](#). Votre participation à la Preview Channel est régie par votre accordAWS et les conditions deAWS service, en particulier les conditions de participation au service universel et bêta, et est confidentielle.

Les questions suivantes sont fréquemment posées à propos de la chaîne de prévisualisation.

## Qu'est-ce que la chaîne Preview ?

Le canal de prévisualisation est un point de terminaison du service public qui vous permet d'essayer et de fournir des commentaires sur les nouvelles fonctionnalités du service avant qu'elles ne soient généralement disponibles. Le point de terminaison du service pour le canal de prévisualisation est distinct du point de terminaison de production standard. Vous interagissez avec le point de terminaison en utilisant leAWS CLI, un fichier de modèle de service pour le canal de prévisualisation et des fichiers d'entrée de commandes pour leAWS CLI. Le canal de prévisualisation vous permet d'essayer de nouvelles fonctionnalités sans affecter votre infrastructure de production actuelle. Nous vous encourageons à faire part de [vos commentaires](#) à l'équipe App Mesh afin de garantir qu'App Mesh répond aux exigences les plus importantes des clients. Vos commentaires sur les fonctionnalités pendant qu'elles sont disponibles sur la chaîne de prévisualisation peuvent nous aider à façonner les fonctionnalités d'App Mesh afin que nous puissions fournir le meilleur service possible.

## En quoi le Preview Channel est-il différent de l'App Mesh de production ?

Le tableau suivant répertorie les aspects du service App Mesh qui sont différents de ceux du Preview Channel.



| Aspect                            | App MeshVersion Mesh                             | App Mesh MeshVersion préliminaire                                                                |
|-----------------------------------|--------------------------------------------------|--------------------------------------------------------------------------------------------------|
| 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

Bien que l'exemple figurant dans le tableau pour le service de production App Mesh répertorie laus-west-2 région, le service de production est disponible dans la plupart des régions. Pour obtenir la liste de toutes les régions dans lesquelles le service de production App Mesh est disponible, consultez la section [AWS App MeshEndpoints et quotas](#). Toutefois, le service App Mesh Preview Channel n'est disponible que dans laus-west-2 région.

## Comment puis-je utiliser les fonctionnalités de la chaîne de prévisualisation ?

1. Ajoutez le modèle de service Preview Channel qui inclut la fonctionnalité Preview Channel à l'AWS CLI à l'aide de la commande suivante.

```
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. Créez un fichier JSON qui inclut la fonctionnalité, en vous basant sur l'exemple JSON et les instructions fournies dans le [guide de l'utilisateur AWS App Mesh](#) de la fonctionnalité.
3. Implémentez la fonctionnalité à l'aide de la commande AWS CLI et du fichier d'entrée de commande appropriés. Par exemple, la commande suivante crée un itinéraire avec les fonctionnalités de Preview Channel à l'aide du fichier *route.json*.

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

4. Ajoutez-le `APPMESH_PREVIEW = 1` en tant que variable de configuration pour le conteneur Envoy lorsque vous l'ajoutez à vos définitions de tâches Amazon ECS, aux spécifications du Kubernetes Pod ou à vos instances Amazon EC2. Cette variable permet au conteneur Envoy de communiquer avec les points de terminaison du canal de prévisualisation. Pour plus d'informations sur l'ajout de variables de configuration, consultez [Mise à jour des services dans Amazon ECS](#), [Mise à jour des services dans Kubernetes](#) et [Mise à jour des services sur Amazon EC2](#).

## Comment fournir des commentaires ?

Vous pouvez nous faire part de vos commentaires directement sur le problème de [GitHub dépôt de la feuille de route App Mesh](#), dont le lien figure dans la documentation relative à la fonctionnalité.

## De combien de temps ai-je pour envoyer des commentaires sur une fonctionnalité de la chaîne de prévisualisation ?

Le temps de réponse varie en fonction de la taille et de la complexité de la fonctionnalité introduite. Nous avons l'intention d'accorder une période de commentaires de 14 jours entre la sortie d'une fonctionnalité jusqu'à la version préliminaire et la sortie de la fonctionnalité en production. L'équipe App Mesh peut prolonger la période de commentaires pour des fonctionnalités spécifiques.

## Quel est le niveau d'assistance fourni pour la chaîne de prévisualisation ?

Bien que nous vous encourageons à fournir des commentaires et des rapports de bogues directement sur le [problème de laGitHub feuille](#) de route d'App Mesh, nous sommes conscients que vous pouvez avoir des données sensibles à partager ou que vous pouvez trouver un problème que vous ne pouvez pas divulguer publiquement en toute sécurité. Pour ces problèmes, vous pouvez contacter AWS Support ou faire part de vos commentaires en [envoyant un e-mail](#) directement à l'équipe App Mesh.

## Mes données sont-elles sécurisées sur le terminal Preview Channel ?

Oui. Le canal de prévisualisation bénéficie du même niveau de sécurité que le point de terminaison de production standard.

## Pendant combien de temps ma configuration sera-t-elle disponible ?

Vous pouvez travailler avec un maillage dans le canal de prévisualisation pendant trente jours. Trente jours après la création d'un maillage, vous pouvez uniquement le répertorier, le lire ou le supprimer. Si vous tentez de créer ou de mettre à jour des ressources après trente jours, vous recevrez une `BadRequest` exception expliquant que le maillage est archivé.

## Quels outils puis-je utiliser pour travailler avec la chaîne de prévisualisation ?

Vous pouvez l'utiliser AWS CLI avec un fichier de modèle de service Preview Channel et des fichiers d'entrée de commandes. Pour plus d'informations sur la façon d'utiliser les fonctionnalités, consultez [Comment puis-je utiliser les fonctionnalités de la chaîne de prévisualisation ?](#). Vous ne pouvez pas utiliser les options de AWS CLI commande AWS Management Console, les SDK ou AWS CloudFormation utiliser les fonctionnalités de Preview Channel. Vous pouvez toutefois utiliser tous les outils une fois qu'une fonctionnalité est disponible pour le service de production.

## Les API Preview Channel seront-elles rétrocompatibles ?

Non. Les API peuvent changer en fonction des commentaires.

## Les fonctionnalités de Preview Channel sont-elles complètes ?

Non. Les nouveaux objets d'API peuvent ne pas être complètement intégrés dans leAWS Management ConsoleAWS CloudFormation, ouAWS CloudTrail. Au fur et à mesure que les fonctionnalités se consolideront dans le canal de prévisualisation et seront bientôt disponibles pour le grand public, les intégrations seront éventuellement disponibles.

## La documentation relative aux fonctionnalités de Preview Channel est-elle disponible ?

Oui. La documentation relative aux fonctionnalités de Preview Channel est incluse dans la documentation de production. Par exemple, si les fonctionnalités de la ressource d'itinéraire sont publiées sur le canal de prévisualisation, les informations relatives à l'utilisation des fonctionnalités figureront dans le document de ressource d'[itinéraire](#) existant. Les fonctionnalités de la chaîne de prévisualisation sont étiquetées comme étant uniquement disponibles dans la chaîne de prévisualisation.

## Comment savoir quand de nouvelles fonctionnalités seront disponibles dans la chaîne de prévisualisation ?

Lorsque de nouvelles fonctionnalités sont introduites dans le canal de prévisualisation, une entrée est ajoutée à l'[historique des documents App Mesh](#). Vous pouvez consulter la page régulièrement ou vous abonner au [flux RSS App Mesh Document History](#). En outre, vous pouvez consulter les [problèmes relatifs](#) au GitHub référentielAWS App Mesh de feuilles de route. Un lien de téléchargement pour le fichier JSON du modèle de service Preview Channel est ajouté à un problème lors de sa publication sur le Preview Channel. Pour plus d'informations sur l'utilisation du modèle et de la fonction, consultez [Comment puis-je utiliser les fonctionnalités de la chaîne de prévisualisation ?](#).

## Comment savoir si une fonctionnalité est passée au service de production ?

Le texte de la documentation App Mesh indiquant que la fonctionnalité n'est disponible que dans le canal de prévisualisation est supprimé et une entrée est ajoutée à l'[historique des documents App Mesh](#). Vous pouvez consulter la page régulièrement ou vous abonner au [flux RSS App Mesh Document History](#).

# App Mesh

AWS App Mesh est intégré à Service Quotas, un AWS service qui vous permet d'afficher et de gérer vos quotas à partir d'un emplacement central. Les quotas de service sont également appelés limites. Pour de plus amples informations, veuillez consulter [What Is Service Quotas?](#) dans le Guide de l'utilisateur Service Quotas.

## App Mesh

Pour afficher les App Mesh AWS Management Console

1. Ouvrez la console Service Quotas à l'adresse <https://console.aws.amazon.com/servicequotas/>.
2. Dans le panneau de navigation, choisissez services AWS.
3. Dans la liste des services AWS, recherchez et sélectionnez AWS App Mesh.

Dans la liste Service quotas (Quotas de service), vous pouvez voir le nom du Service Quota, la valeur appliquée (le cas échéant), le quota AWS par défaut et si la valeur du quota est réglable.

4. Pour afficher des informations supplémentaires sur un quota de service, notamment la description, choisissez le nom du quota.

Pour demander une augmentation de quota, consultez [Demander une augmentation de quota](#) dans le Guide de l'utilisateur de Service Quotas.

Pour afficher les App Mesh AWS CLI

Exécutez la commande suivante .

```
aws service-quotas list-aws-default-service-quotas \
 --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
 --service-code appmesh \
 --output table
```

Pour utiliser davantage les quotas de service à l'aide de l'AWS CLI, veuillez consulter la [Référence des commandes de la AWS CLI Services Quotas](#).

## Historique du document pour App Mesh

Le tableau suivant décrit les principales mises à jour et les nouvelles fonctions pour le Guide de l'utilisateur AWS App Mesh . Nous mettons aussi la documentation à jour régulièrement pour prendre en compte les commentaires qui nous sont envoyés.

| Modification                                                                                | Description                                                                                                                                 | Date            |
|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <a href="#">AWSAppMeshFullAccess Politique mise à jour</a>                                  | Mis AWSAppMeshFullAccess à jour pour permettre l'accès aux UntagResource API TagResource et.                                                | 24 avril 2024   |
| <a href="#">CloudTrail documentation d'intégration mise à jour</a>                          | La documentation décrivant l'intégration d'App Mesh CloudTrail pour enregistrer l'activité de l'API a été mise à jour.                      | 28 mars 2024    |
| <a href="#">Politiques mises à jour</a>                                                     | Mis à jour AWSServiceRoleForAppMesh et AWSAppMeshServiceRolePolicy pour permettre l'accès à l' AWS Cloud Map DiscoverInstancesRevision API. | 12 octobre 2023 |
| <a href="#">Prise en charge de la politique des points de terminaison VPC pour App Mesh</a> | App Mesh prend désormais en charge les politiques relatives aux points de terminaison VPC.                                                  | 11 mai 2023     |
| <a href="#">Plusieurs écouteurs pour App Mesh</a>                                           | App Mesh prend désormais en charge plusieurs auditeurs.                                                                                     | 18 août 2022    |
| <a href="#">IPv6 pour App Mesh</a>                                                          | App Mesh prend désormais en charge le protocole IPv6.                                                                                       | 18 mai 2022     |

|                                                                                                |                                                                                                                    |                  |
|------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">CloudTrail support de journalisation pour le service de gestion App Mesh Envoy</a> | App Mesh prend désormais en charge la CloudTrail journalisation pour le service de gestion App Mesh Envoy.         | 18 mars 2022     |
| <a href="#">App Mesh Agent pour Envoy</a>                                                      | App Mesh est désormais compatible avec Agent for Envoy.                                                            | 25 février 2022  |
| <a href="#">Plusieurs écouteurs pour App Mesh</a>                                              | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez implémenter plusieurs écouteurs pour App Mesh.  | 23 novembre 2021 |
| <a href="#">Support ARM64 pour App Mesh</a>                                                    | App Mesh est désormais compatible avec ARM64.                                                                      | 19 novembre 2021 |
| <a href="#">Extension de métriques pour App Mesh</a>                                           | Vous pouvez implémenter des extensions de métriques pour App Mesh.                                                 | 29 octobre 2021  |
| <a href="#">Mettre en œuvre des améliorations du trafic entrant</a>                            | Vous pouvez implémenter la correspondance entre le nom d'hôte et l'en-tête et réécrire le nom d'hôte et le chemin. | 14 juin 2021     |
| <a href="#">Mettre en œuvre l'authentification TLS mutuelle</a>                                | Vous pouvez implémenter l'authentification TLS mutuelle.                                                           | 4 février 2021   |
| <a href="#">Lancement régional dans af-south-1</a>                                             | App Mesh est désormais disponible dans la région af-south-1.                                                       | 22 janvier 2021  |
| <a href="#">Mettre en œuvre l'authentification TLS mutuelle</a>                                | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez implémenter l'authentification TLS mutuelle.    | 23 novembre 2020 |



|                                                                                                                                          |                                                                                                                                                                                          |                   |
|------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <a href="#">Mettre en œuvre le regroupement de connexions pour un écouteur de passerelle virtuelle</a>                                   | Vous pouvez implémenter le regroupement de connexions pour un écouteur de passerelle virtuelle.                                                                                          | 5 novembre 2020   |
| <a href="#">Mettre en œuvre le regroupement des connexions et la détection des valeurs aberrantes pour un écouteur de nœuds virtuels</a> | Vous pouvez implémenter le regroupement des connexions et la détection des valeurs aberrantes pour un écouteur de nœuds virtuels.                                                        | 5 novembre 2020   |
| <a href="#">Lancement de la région dans eu-south-1</a>                                                                                   | App Mesh est désormais disponible dans la région eu-south-1.                                                                                                                             | 21 octobre 2020   |
| <a href="#">Mettre en œuvre le regroupement de connexions pour un écouteur de passerelle virtuelle</a>                                   | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez implémenter le regroupement de connexions pour un écouteur de passerelle virtuelle.                                   | 28 septembre 2020 |
| <a href="#">Mettre en œuvre le regroupement des connexions et la détection des valeurs aberrantes pour un écouteur de nœuds virtuels</a> | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez implémenter le regroupement des connexions et la détection des valeurs aberrantes pour un écouteur de nœuds virtuels. | 28 septembre 2020 |
| <a href="#">Création d'une passerelle virtuelle et d'un itinéraire de passerelle pour le maillage entrant</a>                            | Permettez aux ressources situées en dehors d'un maillage de communiquer avec les ressources situées à l'intérieur d'un maillage.                                                         | 10 juillet 2020   |

|                                                                                                                     |                                                                                                                                                                                                                                                                                     |                 |
|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <a href="#">Créer et gérer des ressources App Mesh depuis Kubernetes avec le contrôleur App Mesh for Kubernetes</a> | Vous pouvez créer et gérer des ressources App Mesh depuis Kubernetes. Le contrôleur injecte également automatiquement le proxy Envoy et les conteneurs init dans les pods que vous déployez.                                                                                        | 18 juin 2020    |
| <a href="#">Ajouter une valeur de délai d'attente à un nœud virtuel, à un écouteur et à une route</a>               | <a href="#">Vous pouvez ajouter une valeur de délai d'attente à un écouteur et à une route de nœuds virtuels.</a>                                                                                                                                                                   | 18 juin 2020    |
| <a href="#">Ajouter une valeur de délai d'attente à un écouteur de nœud virtuel</a>                                 | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez ajouter une valeur de délai d'attente à un écouteur de nœuds virtuels.                                                                                                                                           | 29 mai 2020     |
| <a href="#">Création d'une passerelle virtuelle pour le maillage entrant</a>                                        | ( <a href="#">App Mesh Preview Channel</a> uniquement) Permettez aux ressources situées en dehors d'un maillage de communiquer avec les ressources situées à l'intérieur d'un maillage.                                                                                             | 8 avril 2020    |
| <a href="#">Chiffrement TLS</a>                                                                                     | ( <a href="#">App Mesh Preview Channel</a> uniquement) Utilisez les certificats d'une autorité de certification AWS Private Certificate Authority ou de votre propre autorité de certification pour chiffrer les communications entre les nœuds virtuels à l'aide du protocole TLS. | 17 janvier 2020 |

|                                                                                                     |                                                                                                                                                                                                                                                |                   |
|-----------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <a href="#">Partager un maillage avec un autre compte</a>                                           | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez partager un maillage avec un autre compte. Les ressources créées par n'importe quel compte peuvent communiquer avec les autres ressources du maillage.                      | 17 janvier 2020   |
| <a href="#">Ajouter une valeur de délai d'attente à un itinéraire</a>                               | ( <a href="#">App Mesh Preview Channel</a> uniquement) Vous pouvez ajouter une valeur de délai d'attente à un itinéraire.                                                                                                                      | 17 janvier 2020   |
| <a href="#">Création d'un proxy App Mesh sur un AWS Outpost</a>                                     | Vous pouvez créer un proxy App Mesh Envoy sur un AWS Outpost.                                                                                                                                                                                  | 3 décembre 2019   |
| <a href="#">Support HTTP/2 et gRPC pour les routes, les routeurs virtuels et les nœuds virtuels</a> | Vous pouvez acheminer le trafic qui utilise les protocoles HTTP/2 et gRPC. Vous pouvez également ajouter un écouteur pour ces protocoles aux <a href="#">nœuds virtuels</a> et aux <a href="#">routeurs virtuels</a> .                         | 25 octobre 2019   |
| <a href="#">Politique relative aux nouvelles tentatives</a>                                         | Une stratégie de nouvelle tentative permet aux clients de se protéger contre les défaillances intermittentes du réseau et les défaillances intermittentes côté serveur. Vous pouvez ajouter une logique de nouvelle tentative à un itinéraire. | 10 septembre 2019 |

---

|                                                             |                                                                                                                                                                                                                                                                                                                                                                                  |                  |
|-------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Chiffrement TLS</a>                             | <a href="#">(App Mesh Preview Channel uniquement)</a> Chiffrez les communications entre les nœuds virtuels à l'aide du protocole TLS.                                                                                                                                                                                                                                            | 6 septembre 2019 |
| <a href="#">Routage basé sur les en-têtes HTTP</a>          | Acheminez le trafic en fonction de la présence et des valeurs des en-têtes HTTP dans une requête.                                                                                                                                                                                                                                                                                | 15 août 2019     |
| <a href="#">Disponibilité de la chaîne App Mesh Preview</a> | L'App Mesh Preview Channel est une variante distincte du service App Mesh. La chaîne d'aperçu présente les fonctionnalités à venir que vous pourrez essayer au fur et à mesure de leur développement. Lorsque vous utilisez des fonctionnalités dans le canal de prévisualisation, vous pouvez fournir des commentaires GitHub pour définir le comportement des fonctionnalités. | 19 juillet 2019  |

[Points de terminaison VPC de l'interface App Mesh \(AWS PrivateLink\)](#)

Améliorez le niveau de sécurité de votre VPC en configurant App Mesh pour qu'il utilise un point de terminaison VPC d'interface. Les points de terminaison de l'interface sont alimentés par AWS PrivateLink une technologie qui vous permet d'accéder en privé aux API App Mesh en utilisant des adresses IP privées. PrivateLink restreint tout le trafic réseau entre votre VPC et App Mesh vers le réseau Amazon.

14 juin 2019

[Ajouté AWS Cloud Map en tant que méthode de découverte des services de nœuds virtuels](#)

Vous pouvez spécifier le DNS ou AWS Cloud Map en tant que méthode de découverte du service de nœud virtuel. Pour être utilisé AWS Cloud Map pour la découverte de services, votre compte doit avoir le rôle [lié au service](#) App Mesh.

13 juin 2019

[Créez automatiquement des ressources App Mesh dans Kubernetes](#)

Créez des ressources App Mesh et ajoutez automatiquement les images du conteneur App Mesh à vos déploiements Kubernetes lorsque vous créez des ressources dans Kubernetes.

11 juin 2019

---

|                                                   |                                                                                                                                                                                                                                                                                                  |                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Disponibilité générale d'App Mesh</a> | Le service App Mesh est désormais généralement disponible pour une utilisation en production.                                                                                                                                                                                                    | 27 mars 2019     |
| <a href="#">Mise à jour de l'API App Mesh</a>     | Les API App Mesh ont été mises à jour pour améliorer la convivialité. Pour plus d'informations, consultez <a href="#">[FEATURE] Ajouter des écouteurs aux routeurs virtuels</a> et <a href="#">[BUG] Routes pour cibler des nœuds virtuels dont les ports ne correspondent pas à Blackhole</a> . | 7 mars 2019      |
| <a href="#">Version initiale d'App Mesh</a>       | Documentation initiale pour la version préliminaire publique du service                                                                                                                                                                                                                          | 28 novembre 2018 |

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.