要件
Kubernetes clusters

NetApp
April 16, 2024

目次

要件

AWS での Kubernetes クラスタの要件

AWS上の管理対象のAmazon Elastic Kubernetes Service（EKS）クラスタまたは自己管理型のKubernetesクラスタをBlueXPに追加できます。BlueXPにクラスタを追加する前に’次の要件が満たされていることを確認する必要があります

> このトピックでは、 _Kubernetes cluster_where configuration is the same for EKS and selfmanaged Kubernetes clusters を使用します。クラスタタイプは設定が異なる場所で指定します。

要件

Astra Trident

最新バージョンの 4 つの Astra Trident が必要です。Astra Tridentは、BlueXPから直接インストールまたはアップグレードできます。お勧めします "前提条件を確認します" Astra Trident をインストールする前に、

Cloud Volumes ONTAP

Cloud Volumes ONTAP for AWS は、クラスタのバックエンドストレージとしてセットアップする必要があります。 "設定手順については、 Astra Trident のドキュメントを参照してください"。

BlueXPコネクタ

必要な権限を持つコネクタが AWS で実行されている必要があります。 詳細は以下をご覧ください。

ネットワーク接続

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタと Cloud Volumes ONTAP の間にはネットワーク接続が必要です。 詳細は以下をご覧ください。

RBAC 許可

BlueXP Connectorロールは、各Kubernetesクラスタで許可されている必要があります。 詳細は以下をご覧ください。

コネクタを準備します

Kubernetesクラスタを検出および管理するには、AWSにBlueXPコネクタが必要です。新しいコネクターを作成するか、必要な権限を持つ既存のコネクターを使用する必要があります。

新しいコネクターを作成します

次のリンクのいずれかの手順に従います。

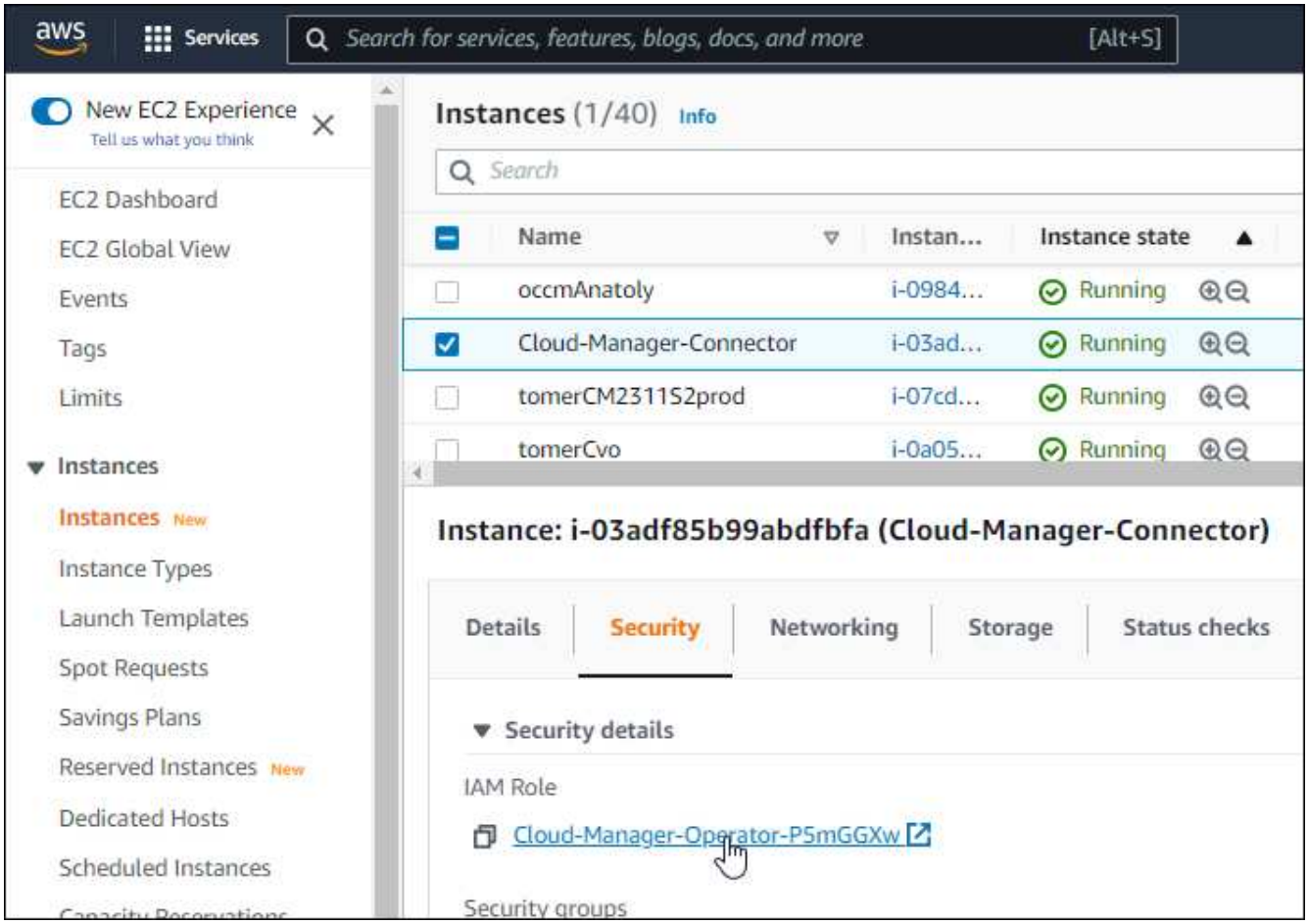
- "BlueXPからコネクタを作成します" （推奨）
- "AWS Marketplace からコネクタを作成します"
- "AWS の既存の Linux ホストにコネクタをインストールします"

必要な権限を既存のコネクタに追加します

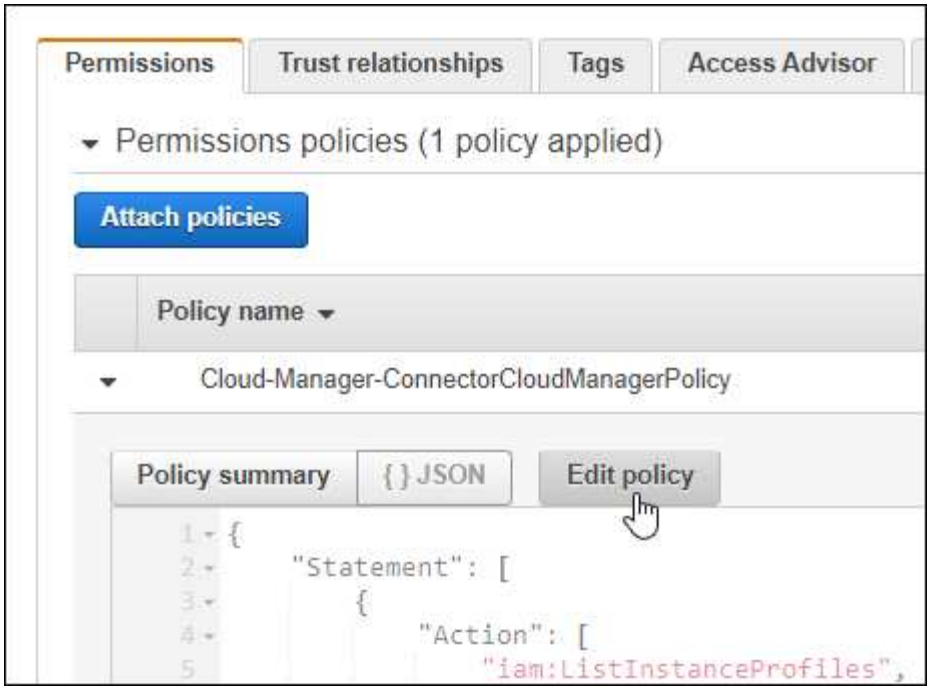
3.9.13 リリース以降、 new_newly で作成されたコネクタには、 Kubernetes クラスタの検出と管理を可能にする新しい AWS 権限が 3 つ含まれています。このリリースよりも前のリリースでコネクタを作成していた場合は、権限を付与するために、コネクタの IAM ロールの既存のポリシーを変更する必要があります。

手順

1. AWS コンソールにアクセスして EC2 サービスを開きます。
2. コネクタインスタンスを選択し、 * セキュリティ * をクリックして、 IAM ロールの名前をクリックし、 IAM サービスでロールを表示します。



3. [* アクセス許可 *] タブで、ポリシーを展開し、 [* ポリシーの編集 *] をクリックします。



4. JSON * をクリックして、最初のアクションセットに次の権限を追加します。
    - EC2: DescribeRegions (説明領域
    - EKS：リストクラスタ
    - EKS：DescribeCluster
    - IAM：GetInstanceProfile

    "ポリシーの完全な JSON 形式を表示します"

5. [ポリシーの確認] をクリックし、 [変更の保存] をクリックします。

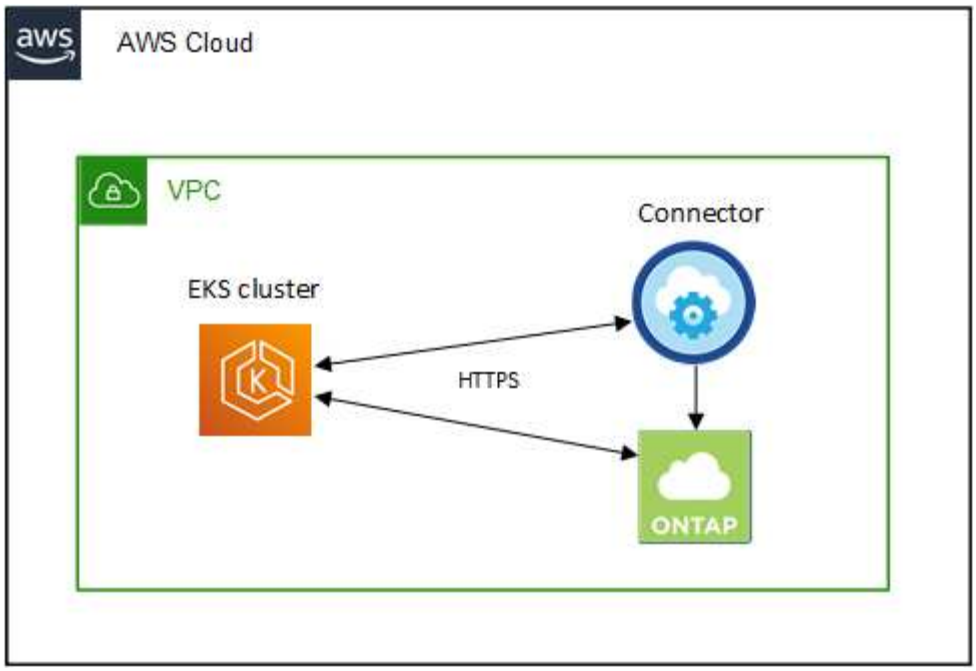
ネットワーク要件を確認します

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタとクラスタにバックエンドストレージを提供する Cloud Volumes ONTAP システムとの間にネットワーク接続を提供する必要があります。

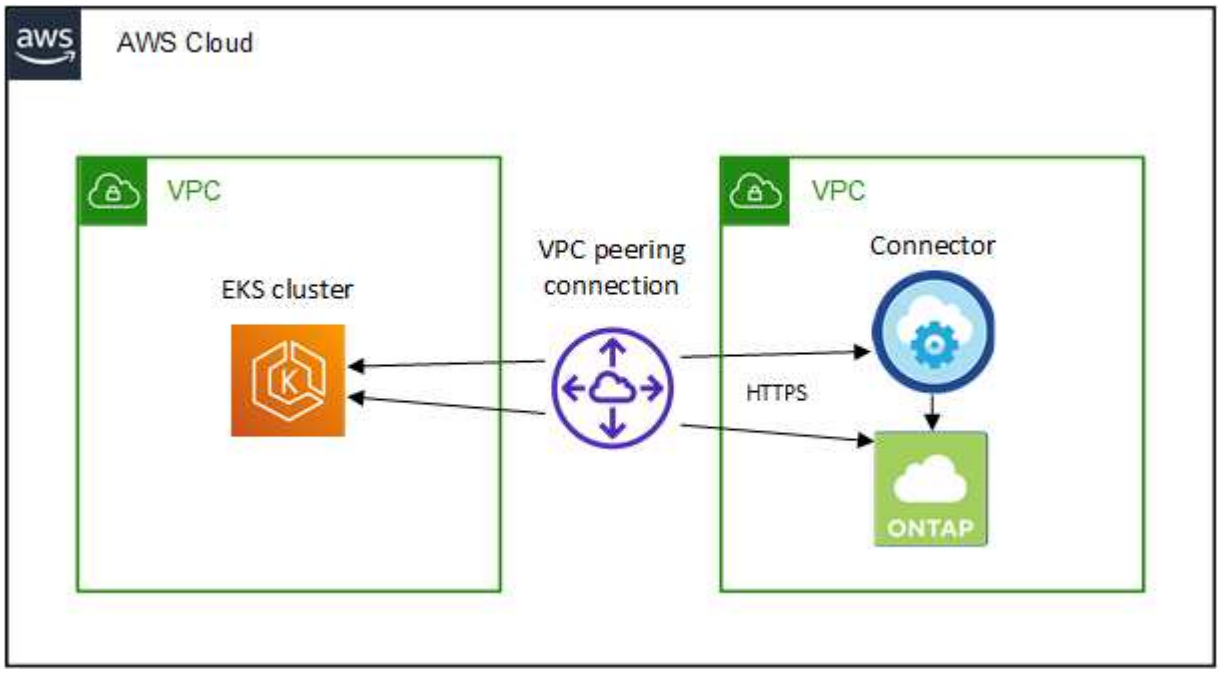
- 各 Kubernetes クラスタがコネクタからインバウンド接続を確立している必要があります
- コネクタには、ポート 443 経由で各 Kubernetes クラスタへのアウトバウンド接続が必要です

この接続を確立する最も簡単な方法は、 Kubernetes クラスタと同じ VPC にコネクタと Cloud Volumes ONTAP を導入することです。VPC が確立されていない場合は、 VPC 間に VPC ピアリング接続を設定する必要があります。

以下は、同じ VPC 内の各コンポーネントの例です。



別の VPC で実行されている EKS クラスタを次に示します。この例では、 VPC ピアリングによって、 EKS クラスタの VPC とコネクタおよび Cloud Volumes ONTAP の VPC 間の接続が確立されます。



RBAC 許可をセットアップします

コネクタがクラスタを検出して管理できるように、各 Kubernetes クラスタで Connector ロールを承認する必要があります。

異なる機能を有効にするには、異なる許可が必要です。

バックアップとリストア

バックアップとリストアに必要なのは、基本的な許可だけです。

ストレージクラスを追加する

BlueXPを使用してストレージクラスを追加し、バックエンドへの変更がないかクラスタを監視するには、拡張された許可が必要です。

Astra Trident をインストールします

BlueXPがAstra Tridentをインストールするためには、完全な権限を付与する必要があります。

> Astra Tridentをインストールすると、BlueXPはAstra Tridentバックエンドと、Astra Tridentのクレデンシャルを含むKubernetesシークレットをインストールして、ストレージクラスタと通信する必要があります。

手順

1. クラスタロールとロールバインドを作成します。
   a. 要件に基づいて承認をカスタマイズできます。

バックアップ / リストア

Kubernetes クラスタのバックアップとリストアを有効にするための基本的な許可を追加する。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
          - namespaces
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumes
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - pods
          - pods/exec
      verbs:
          - get
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumeclaims
      verbs:
          - list
          - create
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - list
```

```
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
      verbs:
          - list
          - watch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentorchestrators
      verbs:
          - get
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: Group
      name: cloudmanager-access-group
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

ストレージクラス

BlueXPを使用してストレージクラスを追加するには'拡張された認証を追加します

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
          - secrets
          - namespaces
          - persistentvolumeclaims
          - persistentvolumes
          - pods
          - pods/exec
```

```
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - get
          - create
          - list
          - watch
          - delete
          - patch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
          - tridentorchestrators
          - tridentbackendconfigs
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: Group
      name: cloudmanager-access-group
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

Tridentのインストール

コマンドラインを使用して完全な認証を行い、BlueXPでAstra Tridentをインストールできるようにします。

```
eksctl create iamidentitymapping --cluster < > --region < > --arn
< > --group "system:masters" --username
system:node:{{EC2PrivateDNSName}}
```

b. クラスタに構成を適用します。

```
kubectl apply -f <file-name>
```

2. 権限グループへの ID マッピングを作成します。

eksctl を使用します

eksctlを使用して、クラスタとBlueXPコネクタ用のIAMロールとの間にIAM IDマッピングを作成します。

"eksctl のマニュアルを参照してください"。

以下に例を示します。

```
eksctl create iamidentitymapping --cluster <eksCluster> --region
<us-east-2> --arn <ARN of the Connector IAM role> --group
cloudmanager-access-group --username
system:node:{{EC2PrivateDNSName}}
```

aws -auth を編集します

AWS- AUTH ConfigMapを直接編集して、BlueXPコネクタのIAMロールへのRBACアクセスを追加します。

"詳細な手順については、 AWS EKS のドキュメントを参照してください"。

以下に例を示します。

```
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - cloudmanager-access-group
      rolearn: <ARN of the Connector IAM role>
     username: system:node:{{EC2PrivateDNSName}}
kind: ConfigMap
metadata:
  creationTimestamp: "2021-09-30T21:09:18Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "1021"
  selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
  uid: dcc31de5-3838-11e8-af26-02e00430057c
```

Azure での Kubernetes クラスタの要件

Azureでは、BlueXPを使用して、マネージドAzure Kubernetesクラスタ（AKS）と自己管理Kubernetesクラスタを追加および管理できます。BlueXPにクラスタを追加する前に’次の要件が満たされていることを確認します

このトピックでは、 _Kubernetes cluster_where configuration is the same for AKS and selfmanaged Kubernetes clusters を使用します。クラスタタイプは設定が異なる場所で指定します。

要件

Astra Trident

最新バージョンの 4 つの Astra Trident が必要です。Astra Tridentは、BlueXPから直接インストールまたはアップグレードできます。お勧めします "前提条件を確認します" Astra Trident をインストールする前に、

Cloud Volumes ONTAP

クラスタのバックエンドストレージとして Cloud Volumes ONTAP が設定されている必要があります。 "設定手順については、 Astra Trident のドキュメントを参照してください"。

BlueXPコネクタ

必要な権限を持つコネクタが Azure で実行されている必要があります。 詳細は以下をご覧ください。

ネットワーク接続

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタと Cloud Volumes ONTAP の間にはネットワーク接続が必要です。 詳細は以下をご覧ください。

RBAC 許可

BlueXPは、Active Directoryを使用するかどうかに関係なく、RBAC対応のクラスタをサポートしています。BlueXP Connectorロールは、各Azureクラスタで許可されている必要があります。 詳細は以下をご覧ください。

コネクタを準備します

Kubernetesクラスタを検出および管理するには、AzureのBlueXPコネクタが必要です。新しいコネクターを作成するか、必要な権限を持つ既存のコネクターを使用する必要があります。

新しいコネクターを作成します

次のリンクのいずれかの手順に従います。

- "BlueXPからコネクタを作成します" （推奨）
- "Azure Marketplace からコネクタを作成します"
- "既存の Linux ホストにコネクタをインストールします"

既存のコネクタに必要な権限を追加する（管理対象の **AKS** クラスタを検出する）

管理対象の AKS クラスタを検出するには、コネクタのカスタムロールを変更して権限を提供しなければならない場合があります。

手順

1. Connector 仮想マシンに割り当てられているロールを特定します。
    a. Azure ポータルで、仮想マシンサービスを開きます。

b. Connector 仮想マシンを選択します。

c. [設定] で、 [**Identity**] を選択します。

d. Azure の役割の割り当て * をクリックします。

e. Connector 仮想マシンに割り当てられているカスタムロールをメモしておきます。

2. カスタムロールを更新します。

a. Azure ポータルで、 Azure サブスクリプションを開きます。

b. [* アクセス制御（ IAM ）]>[役割 *] をクリックします。

c. カスタムロールの省略記号 (...) をクリックし、 * 編集 * をクリックします。

d. JSON をクリックして、次の権限を追加します。

```
"Microsoft.ContainerService/managedClusters/listClusterUserCredential
/action"
"Microsoft.ContainerService/managedClusters/read"
```

e. [* Review + update *] をクリックし、 [* Update *] をクリックします。

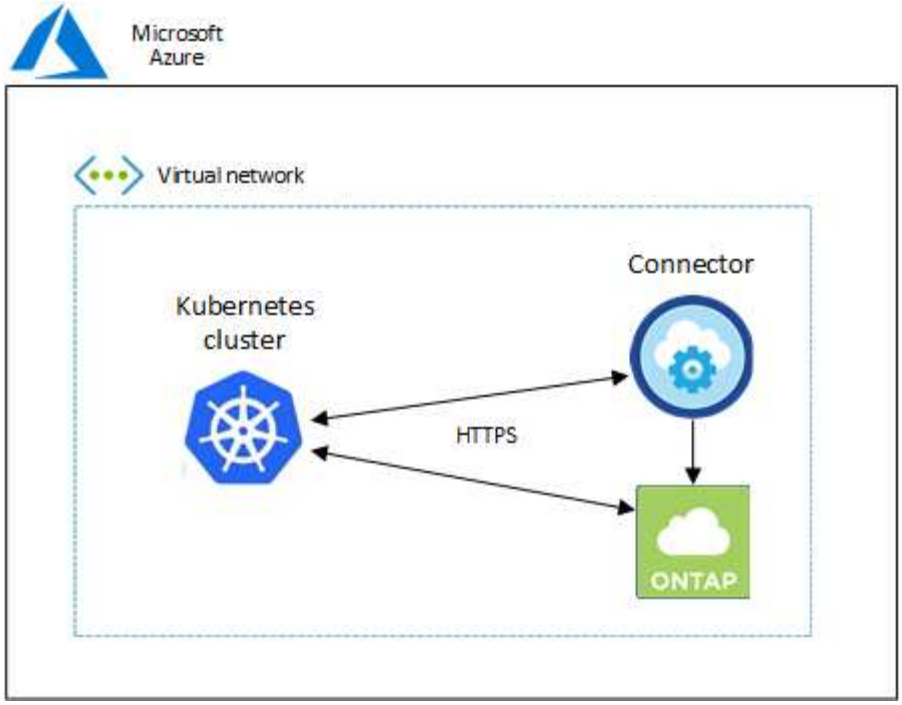
ネットワーク要件を確認します

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタとクラスタにバックエンドストレージを提供する Cloud Volumes ONTAP システムとの間にネットワーク接続を提供する必要があります。

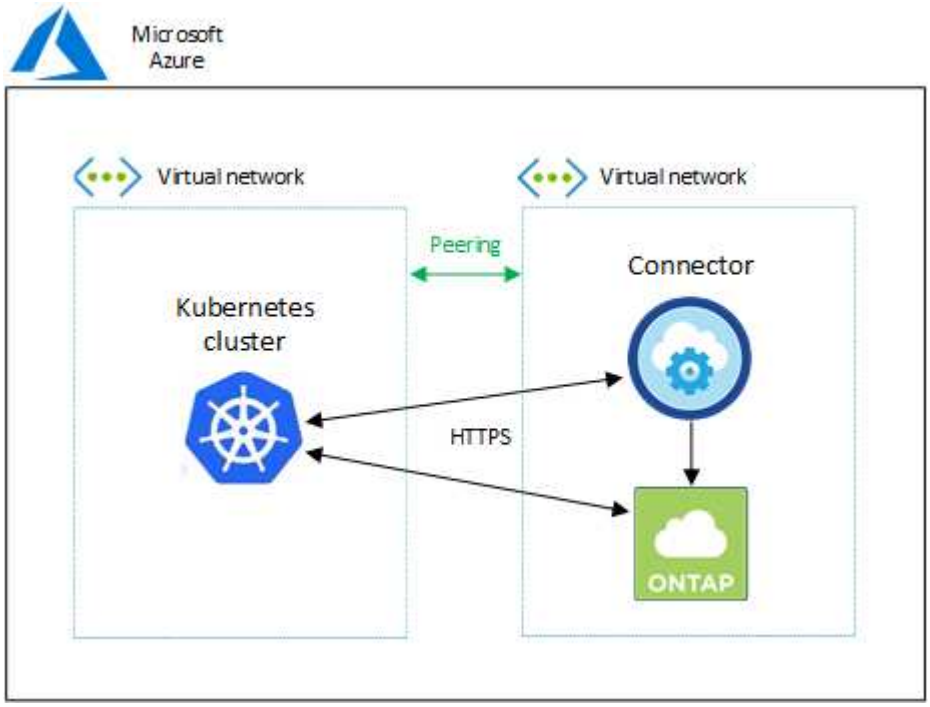
- 各 Kubernetes クラスタがコネクタからインバウンド接続を確立している必要があります
- コネクタには、ポート 443 経由で各 Kubernetes クラスタへのアウトバウンド接続が必要です

この接続を確立する最も簡単な方法は、 Kubernetes クラスタと同じ VNet にコネクタと Cloud Volumes ONTAP を導入することです。それ以外の場合は、異なる VNet 間のピアリング接続を設定する必要があります。

以下は、同じ VNet 内の各コンポーネントの例です。



別の VNet で実行される Kubernetes クラスタの例を次に示します。この例では、ピアリングによって Kubernetes クラスタの VNet とコネクタおよび Cloud Volumes ONTAP の VNet 間の接続が確立されます。



RBAC 許可をセットアップします

RBAC の検証は、 Active Directory （ AD ）が有効になっている Kubernetes クラスタでのみ実行されます。AD を使用しない Kubernetes クラスタは、検証に自動的に合格します。

コネクタがクラスタを検出して管理できるように、各 Kubernetes クラスタで Connector ロールを承認する必要があります。

バックアップとリストア

バックアップとリストアに必要なのは、基本的な許可だけです。

ストレージクラスを追加する

BlueXPを使用してストレージクラスを追加し、バックエンドへの変更がないかクラスタを監視するには、拡張された許可が必要です。

Astra Trident をインストールします

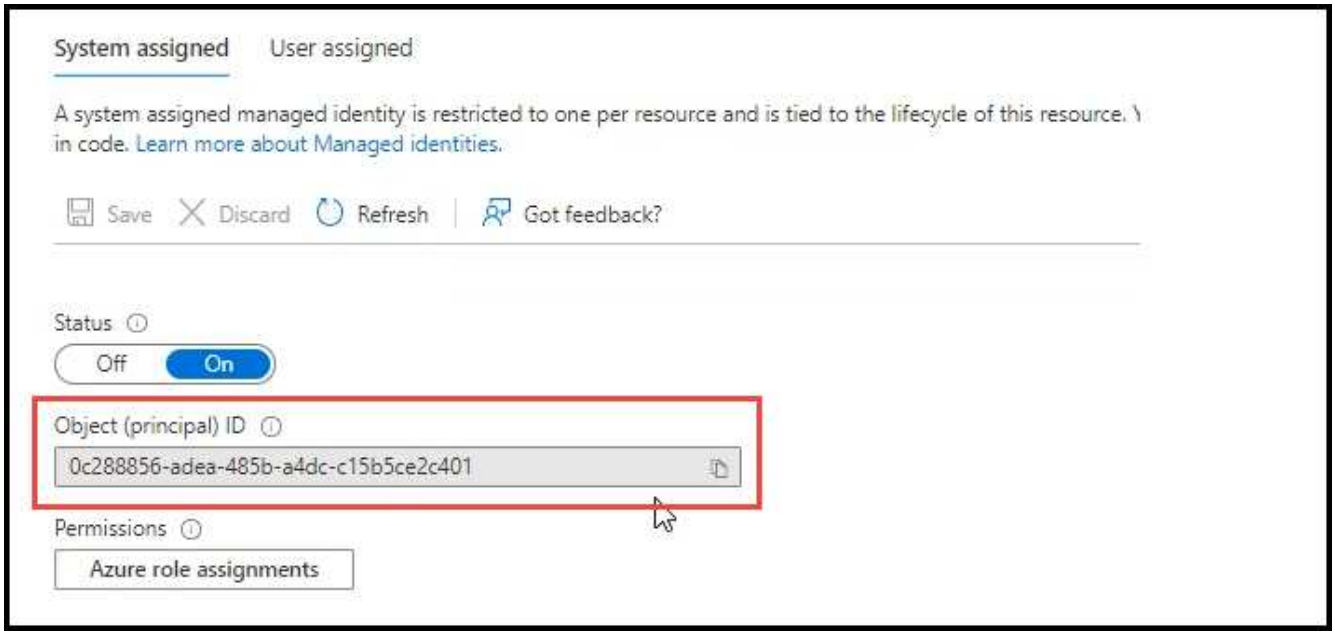
BlueXPがAstra Tridentをインストールするためには、完全な権限を付与する必要があります。

> Astra Tridentをインストールすると、BlueXPはAstra Tridentバックエンドと、Astra Tridentのクレデンシャルを含むKubernetesシークレットをインストールして、ストレージクラスタと通信する必要があります。

作業を開始する前に

RBAC のサブジェクト名 : name:` の構成は、 Kubernetes クラスタのタイプによって若干異なります。

- 管理対象 AKS クラスタ * を導入する場合、コネクターにシステムが割り当てた管理 ID のオブジェクト ID が必要です。この ID は Azure 管理ポータルで入手できます。



- 自己管理型の Kubernetes クラスタ * を導入する場合は、許可されたユーザのユーザ名が必要です。

手順

クラスタロールとロールバインドを作成します。

1. 要件に基づいて承認をカスタマイズできます。

バックアップ / リストア

Kubernetes クラスタのバックアップとリストアを有効にするための基本的な許可を追加する。

を交換します `subjects: kind:` 変数にユーザ名とを入力します `subjects: name:` システムが割り当てた管理IDのオブジェクトID、または上記のように許可されたユーザーのユーザー名のいずれかを使用します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
          - namespaces
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumes
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - pods
          - pods/exec
      verbs:
          - get
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumeclaims
      verbs:
          - list
          - create
          - watch
    - apiGroups:
          - storage.k8s.io
```

```
      resources:
          - storageclasses
      verbs:
          - list
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
      verbs:
          - list
          - watch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentorchestrators
      verbs:
          - get
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

ストレージクラス

BlueXPを使用してストレージクラスを追加するには'拡張された認証を追加します

を交換します `subjects: kind:` 変数にユーザ名とを入力します `subjects: user:` システムが割り当てた管理IDのオブジェクトID、または上記のように許可されたユーザーのユーザー名のいずれかを使用します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
```

```
      resources:
          - secrets
          - namespaces
          - persistentvolumeclaims
          - persistentvolumes
          - pods
          - pods/exec
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - get
          - create
          - list
          - watch
          - delete
          - patch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
          - tridentorchestrators
          - tridentbackendconfigs
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
```

```
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

Tridentのインストール

コマンドラインを使用して完全な認証を行い、BlueXPでAstra Tridentをインストールできるようにします。

```
eksctl create iamidentitymapping --cluster < > --region < > --arn <
> --group "system:masters" --username
system:node:{{EC2PrivateDNSName}}
```

2. クラスタに構成を適用します。

```
kubectl apply -f <file-name>
```

Google Cloud の Kubernetes クラスタの要件

管理対象のGoogle Kubernetes Engine（GKE）クラスタと、BlueXPを使用してGoogleで自己管理型Kubernetesクラスタを追加および管理できます。BlueXPにクラスタを追加する前に'次の要件が満たされていることを確認します

このトピックでは、 _Kubernetes cluster_where 構成は、 GKE クラスタと自己管理型 Kubernetes クラスタで同じです。クラスタタイプは設定が異なる場所で指定します。

要件

Astra Trident

最新バージョンの 4 つの Astra Trident が必要です。Astra Tridentは、BlueXPから直接インストールまたはアップグレードできます。お勧めします "前提条件を確認します" Astra Trident をインストールする前に

Cloud Volumes ONTAP

Cloud Volumes ONTAP は、Kubernetesクラスタと同じテナンシーアカウント、ワークスペース、コネクタを使用してBlueXPに配置する必要があります。 "設定手順については、 Astra Trident のドキュメントを参照してください"。

BlueXPコネクタ

必要な権限を持つ Connector が Google で実行されている必要があります。 詳細は以下をご覧ください。

ネットワーク接続

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタと Cloud Volumes ONTAP の間にはネットワーク接続が必要です。 詳細は以下をご覧ください。

RBAC 許可

BlueXPは、Active Directoryを使用するかどうかに関係なく、RBAC対応のクラスタをサポートしています。BlueXPコネクタの役割は、各GKEクラスタで許可されている必要があります。 詳細は以下をご覧ください。

コネクタを準備します

Kubernetesクラスタを検出および管理するには、GoogleのBlueXPコネクタが必要です。新しいコネクターを作成するか、必要な権限を持つ既存のコネクターを使用する必要があります。

新しいコネクターを作成します

次のリンクのいずれかの手順に従います。

- "BlueXPからコネクタを作成します" （推奨）
- "既存の Linux ホストにコネクタをインストールします"

既存のコネクタに必要な権限を追加する（管理対象の **GKE** クラスタを検出するため）

管理対象 GKE クラスタを検出する場合は、コネクタのカスタムロールを変更して権限を付与する必要があります。

手順

1. インチ "Cloud Console の略"をクリックし、 * Roles * ページに移動します。
2. ページ上部のドロップダウンリストを使用して、編集するロールを含むプロジェクトまたは組織を選択します。
3. カスタムロールをクリックします。
4. 役割の権限を更新するには、 * 役割の編集 * をクリックします。
5. [権限の追加 *] をクリックして、次の新しい権限を役割に追加します。

```
container.clusters.get
container.clusters.list
```

6. [更新（ Update ）] をクリックして、編集したロールを保存する。

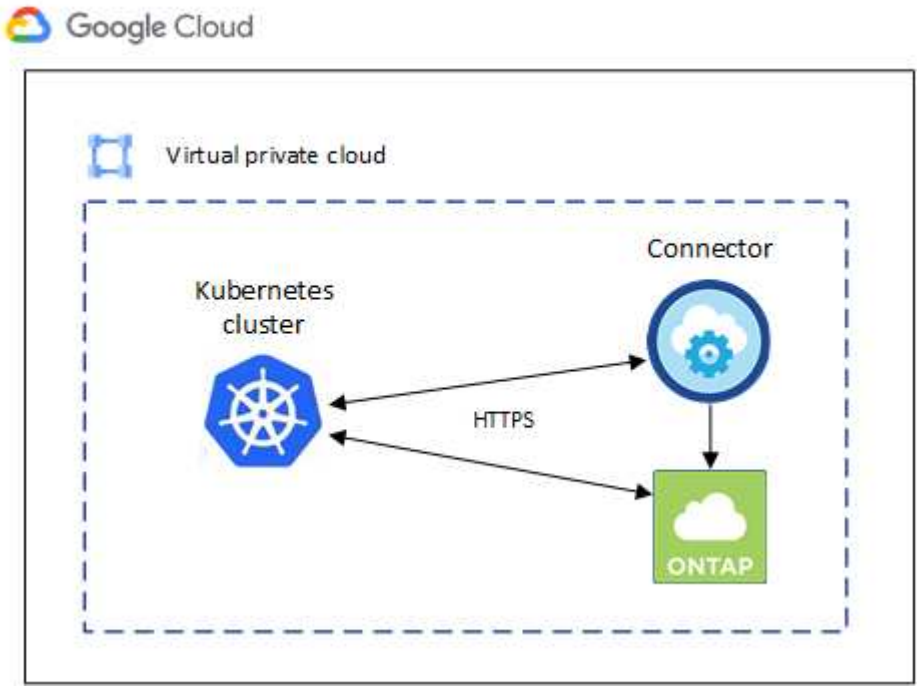
ネットワーク要件を確認します

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタとクラスタにバックエンドストレージを提供する Cloud Volumes ONTAP システムとの間にネットワーク接続を提供する必要があります。

- 各 Kubernetes クラスタがコネクタからインバウンド接続を確立している必要があります
- コネクタには、ポート 443 経由で各 Kubernetes クラスタへのアウトバウンド接続が必要です

この接続を確立する最も簡単な方法は、 Kubernetes クラスタと同じ VPC にコネクタと Cloud Volumes ONTAP を導入することです。それ以外の場合は、異なる VPC 間にピア接続を設定する必要があります。

以下は、同じ VPC 内の各コンポーネントの例です。



RBAC 許可をセットアップします

RBAC の検証は、 Active Directory （ AD ）が有効になっている Kubernetes クラスタでのみ実行されます。AD を使用しない Kubernetes クラスタは、検証に自動的に合格します。

コネクタがクラスタを検出して管理できるように、各 Kubernetes クラスタで Connector ロールを承認する必要があります。

バックアップとリストア

バックアップとリストアに必要なのは、基本的な許可だけです。

ストレージクラスを追加する

BlueXPを使用してストレージクラスを追加し、バックエンドへの変更がないかクラスタを監視するには、拡張された許可が必要です。

Astra Trident をインストールします

BlueXPがAstra Tridentをインストールするためには、完全な権限を付与する必要があります。

> Astra Tridentをインストールすると、BlueXPはAstra Tridentバックエンドと、Astra Tridentのクレデンシャルを含むKubernetesシークレットをインストールして、ストレージクラスタと通信する必要があります。

作業を開始する前に

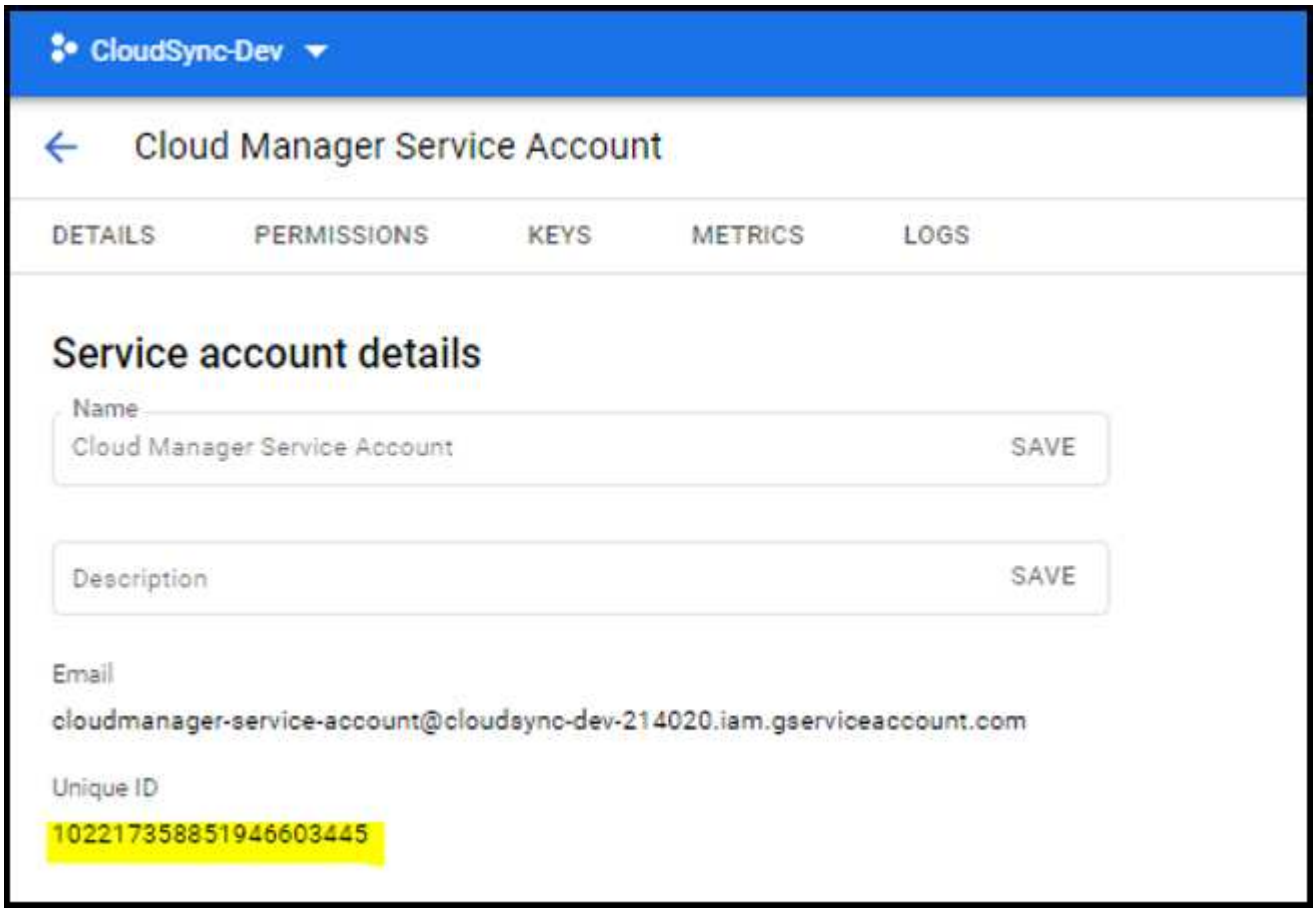
を設定します `subjects: name:` YAMLファイルで、BlueXPの一意のIDを知っている必要があります。

一意の ID は、次の 2 つの方法のいずれかで確認できます。

- コマンドを使用します。

```
gcloud iam service-accounts list
gcloud iam service-accounts describe <service-account-email>
```

- のサービスアカウントの詳細で確認します "Cloud Console の略"。



手順

クラスタロールとロールバインドを作成します。

1. 要件に基づいて承認をカスタマイズできます。

バックアップ / リストア

Kubernetes クラスタのバックアップとリストアを有効にするための基本的な許可を追加する。

を交換します `subjects: kind:` 変数にユーザ名とを入力します `subjects: name:` 承認されたサービスアカウントの一意のIDを使用します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
          - namespaces
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumes
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - pods
          - pods/exec
      verbs:
          - get
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumeclaims
      verbs:
          - list
          - create
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
```

```
            - storageclasses
        verbs:
            - list
    - apiGroups:
            - trident.netapp.io
        resources:
            - tridentbackends
        verbs:
            - list
            - watch
    - apiGroups:
            - trident.netapp.io
        resources:
            - tridentorchestrators
        verbs:
            - get
            - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
      apiGroup: rbac.authorization.k8s.io
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

ストレージクラス

BlueXPを使用してストレージクラスを追加するには'拡張された認証を追加します

を交換します `subjects: kind:` 変数にユーザ名とを入力します `subjects: user:` 承認されたサービスアカウントの一意のIDを使用します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
            - ''
      resources:
```

```yaml
            - secrets
            - namespaces
            - persistentvolumeclaims
            - persistentvolumes
            - pods
            - pods/exec
        verbs:
            - get
            - list
            - watch
            - create
            - delete
            - watch
    - apiGroups:
            - storage.k8s.io
        resources:
            - storageclasses
        verbs:
            - get
            - create
            - list
            - watch
            - delete
            - patch
    - apiGroups:
            - trident.netapp.io
        resources:
            - tridentbackends
            - tridentorchestrators
            - tridentbackendconfigs
        verbs:
            - get
            - list
            - watch
            - create
            - delete
            - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
subjects:
    - kind: User
      name:
      apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
    kind: ClusterRole
    name: cloudmanager-access-clusterrole
    apiGroup: rbac.authorization.k8s.io
```

Tridentのインストール

コマンドラインを使用して完全な認証を行い、BlueXPでAstra Tridentをインストールできるようにします。

```
kubectl create clusterrolebinding test --clusterrole cluster-admin
--user <Unique ID>
```

2. クラスタに構成を適用します。

```
kubectl apply -f <file-name>
```

OpenShiftでのKubernetesクラスタの要件

BlueXPを使用して、自己管理型OpenShift Kubernetesクラスタを追加および管理できます。BlueXPにクラスタを追加する前に'次の要件が満たされていることを確認します

要件

Astra Trident

最新バージョンの 4 つの Astra Trident が必要です。Astra Tridentは、BlueXPから直接インストールまたはアップグレードできます。お勧めします "前提条件を確認します" Astra Trident をインストールする前に、

Cloud Volumes ONTAP

クラスタのバックエンドストレージとして Cloud Volumes ONTAP が設定されている必要があります。 "設定手順については、 Astra Trident のドキュメントを参照してください"。

BlueXPコネクタ

Kubernetesクラスタをインポートおよび管理するには、BlueXPコネクタが必要です。クラウドプロバイダに必要な権限を持つ新しいコネクタを作成するか、既存のコネクタを使用する必要があります。

- "AWSコネクタ"
- "Azure Connectorの場合"
- "Google Cloud Connectorの場合"

ネットワーク接続

Kubernetes クラスタとコネクタの間、および Kubernetes クラスタと Cloud Volumes ONTAP の間にはネットワーク接続が必要です。

RBAC許可を使用したKubernetes構成ファイル（kubeconfig）

OpenShiftクラスタをインポートするには、さまざまな機能を有効にするために必要なRBAC許可を含むkubeconfigファイルが必要です。 [kubeconfigファイルを作成します]。

- バックアップとリストア：バックアップとリストアに必要なのは基本的な許可のみです。
- ストレージクラスの追加：BlueXPを使用してストレージクラスを追加し、バックエンドへの変更がないかクラスタを監視するには、拡張された許可が必要です。
- Install Astra Trident：BlueXPがAstra Tridentをインストールするための完全な権限を付与する必要があります。

> Astra Tridentをインストールすると、BlueXPはAstra Tridentバックエンドと、Astra Tridentのクレデンシャルを含むKubernetesシークレットをインストールして、ストレージクラスタと通信する必要があります。

kubeconfigファイルを作成します

OpenShift CLIを使用して、BlueXPにインポートするkubeconfigファイルを作成します。

手順

1. パブリックURLにある「OC login」を使用して、管理ユーザとともにOpenShift CLIにログインします。
2. 次の手順でサービスアカウントを作成します。

   a. 「OC-service-account.yaml」という名前のサービスアカウントファイルを作成します。

   名前と名前空間を必要に応じて調整します。ここで変更を行った場合は、以降の手順でも同じ変更を適用する必要があります。

```
oc-service-account.yaml
```

+

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: oc-service-account
  namespace: default
```

   a. サービスアカウントを適用します。

```
kubectl apply -f oc-service-account.yaml
```

3. 許可要件に基づいて、カスタムロールバインディングを作成します。

   a. 「OC-clusterrolebinding.yaml」という名前の「ClusterRoleBinding」ファイルを作成します。

```
oc-clusterrolebinding.yaml
```

b. クラスタに必要に応じてRBAC許可を設定します。

バックアップ / リストア

Kubernetes クラスタのバックアップとリストアを有効にするための基本的な許可を追加する。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
          - namespaces
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumes
      verbs:
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - pods
          - pods/exec
      verbs:
          - get
          - list
          - watch
    - apiGroups:
          - ''
      resources:
          - persistentvolumeclaims
      verbs:
          - list
          - create
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - list
```

```
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
      verbs:
          - list
          - watch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentorchestrators
      verbs:
          - get
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cloudmanager-access-clusterrole
subjects:
    - kind: ServiceAccount
      name: oc-service-account
      namespace: default
```

ストレージクラス

BlueXPを使用してストレージクラスを追加するには'拡張された認証を追加します

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    name: cloudmanager-access-clusterrole
rules:
    - apiGroups:
          - ''
      resources:
          - secrets
          - namespaces
          - persistentvolumeclaims
          - persistentvolumes
          - pods
          - pods/exec
```

```
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
    - apiGroups:
          - storage.k8s.io
      resources:
          - storageclasses
      verbs:
          - get
          - create
          - list
          - watch
          - delete
          - patch
    - apiGroups:
          - trident.netapp.io
      resources:
          - tridentbackends
          - tridentorchestrators
          - tridentbackendconfigs
      verbs:
          - get
          - list
          - watch
          - create
          - delete
          - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: k8s-access-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cloudmanager-access-clusterrole
subjects:
    - kind: ServiceAccount
      name: oc-service-account
      namespace: default
```

Tridentのインストール

フル管理者権限を付与し、BlueXPでAstra Tridentをインストールできるようにします。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cloudmanager-access-clusterrole
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: oc-service-account
  namespace: default
```

c. クラスタロールバインドを適用します。

```
kubectl apply -f oc-clusterrolebinding.yaml
```

4. 「 <context> 」をインストールの正しいコンテキストに置き換えて、サービスアカウントのシークレットをリストします。

```
kubectl get serviceaccount oc-service-account --context <context>
--namespace default -o json
```

出力の末尾は次のようになります。

```
"secrets": [
{ "name": "oc-service-account-dockercfg-vhz87"},
{ "name": "oc-service-account-token-r59kr"}
]
```

'ecsレット ' 配列内の各要素のインデックスは 0 から始まります上の例では、「OC-service-account-dockercfg-vhz87」のインデックスは0になり、「OC-service-account-token-r59kr」のインデックスは1になります。出力で、 "token" という単語が含まれるサービスアカウント名のインデックスをメモしてください。

5. 次のように kubeconfig を生成します。

a. 「 create-kubeconfig .sh` ファイル」を作成します。次のスクリプトの先頭にある「 token_index 」を正しい値に置き換えます。

```
create-kubeconfig.sh
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=oc-service-account
NAMESPACE=default
NEW_CONTEXT=oc
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
```

```
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. コマンドをソースにし、 Kubernetes クラスタに適用します。

```
source create-kubeconfig.sh
```

結果

結果を使用します `kubeconfig-sa` ファイルを使用して、OpenShiftクラスタをBlueXPに追加します。