Requirements

Kubernetes clusters

NetApp
June 10, 2024

Table of Contents

Requirements

Requirements for Kubernetes clusters in AWS

You can add managed Amazon Elastic Kubernetes Service (EKS) clusters or self-managed Kubernetes clusters on AWS to BlueXP. Before you can add the clusters to BlueXP, you need to ensure that the following requirements are met.

This topic uses *Kubernetes cluster* where configuration is the same for EKS and self-managed Kubernetes clusters. The cluster type is specified where configuration differs.

Requirements

Astra Trident

One of the four most recent versions of Astra Trident is required. You can install or upgrade Astra Trident directly from BlueXP. You should review the prerequisites prior to installing Astra Trident.

Cloud Volumes ONTAP

Cloud Volumes ONTAP for AWS must be set up as backend storage for the cluster. Go to the Astra Trident docs for configuration steps.

BlueXP Connector

A Connector must be running in AWS with the required permissions. Learn more below.

Network connectivity

Network connectivity is required between the Kubernetes cluster and the Connector and between the Kubernetes cluster and Cloud Volumes ONTAP. Learn more below.

RBAC authorization

The BlueXP Connector role must be authorized on each Kubernetes cluster. Learn more below.

Prepare a Connector

A BlueXP Connector is required in AWS to discover and manage Kubernetes clusters. You'll need to create a new Connector or use an existing Connector that has the required permissions.

Create a new Connector

Follow the steps in one of the links below.

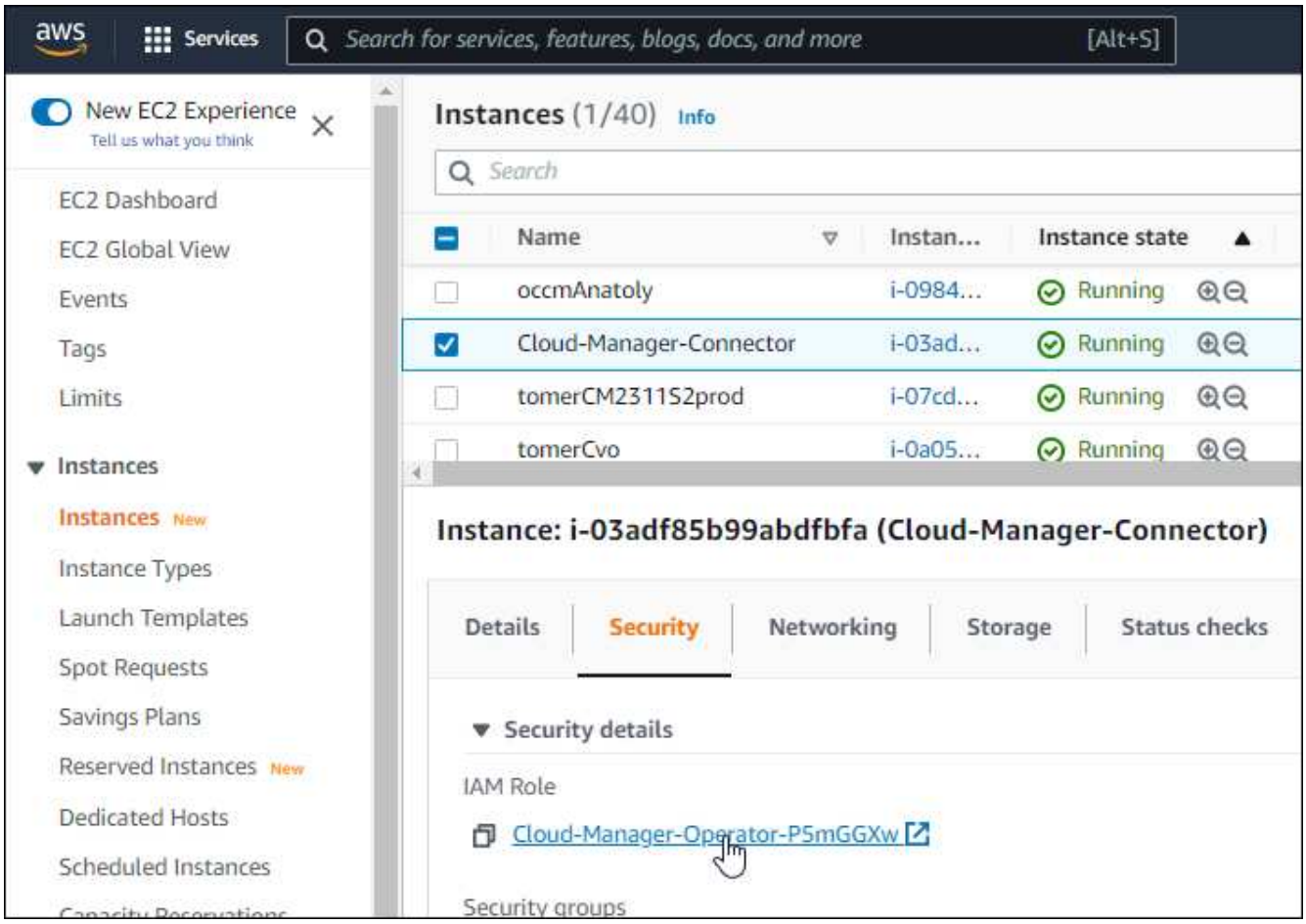
- Create a Connector from BlueXP (recommended)
- Create a Connector from the AWS Marketplace
- Install the Connector on an existing Linux host in AWS

Add the required permissions to an existing Connector

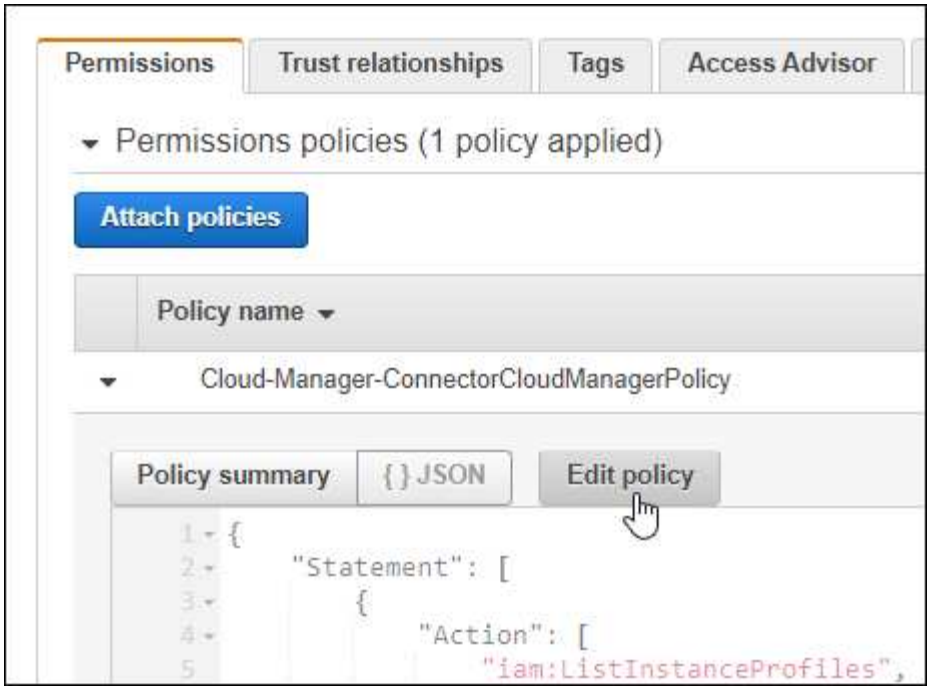
Starting in the 3.9.13 release, any *newly* created Connectors include three new AWS permissions that enable discovery and management of Kubernetes clusters. If you created a Connector prior to this release, then you'll need to modify the existing policy for the Connector's IAM role to provide the permissions.

Steps

1. Go the AWS console and open the EC2 service.
2. Select the Connector instance, click **Security**, and click the name of the IAM role to view the role in the IAM service.



3. In the **Permissions** tab, expand the policy and click **Edit policy**.



4. Click **JSON** and add the following permissions under the first set of actions:

   - ec2:DescribeRegions
   - eks:ListClusters
   - eks:DescribeCluster
   - iam:GetInstanceProfile

   View the full JSON format for the policy

5. Click **Review policy** and then click **Save changes**.

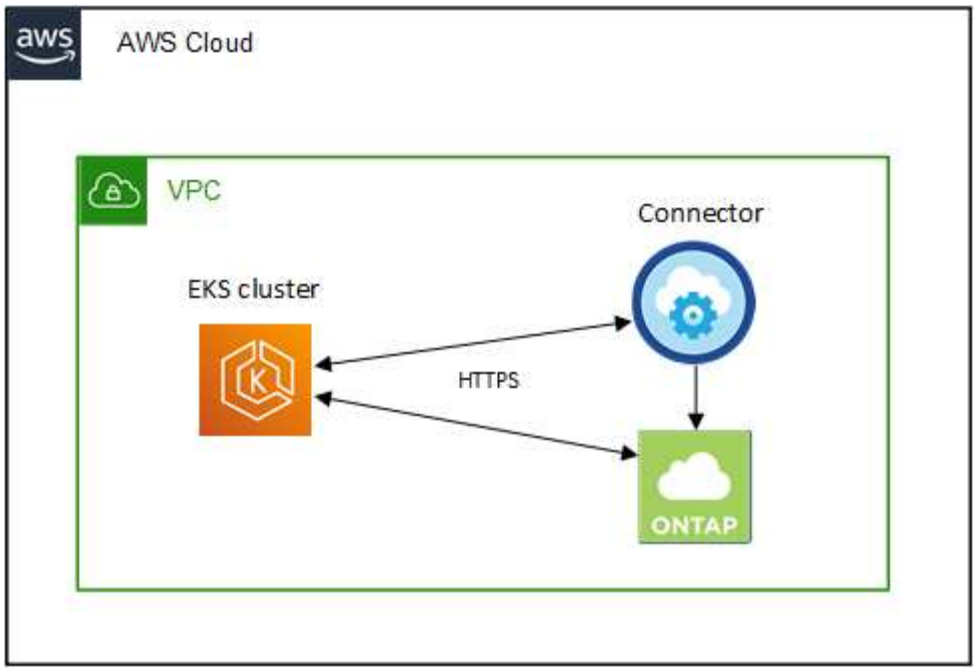
Review networking requirements

You need to provide network connectivity between the Kubernetes cluster and the Connector and between the Kubernetes cluster and the Cloud Volumes ONTAP system that provides backend storage to the cluster.

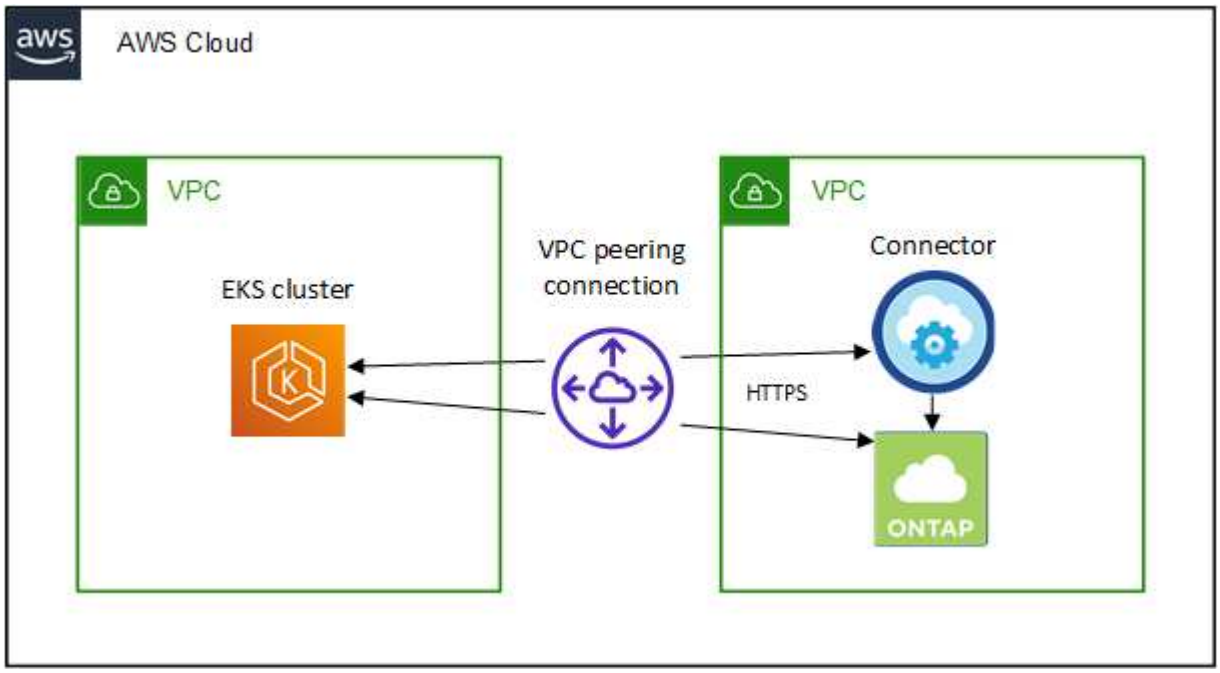
- Each Kubernetes cluster must have an inbound connection from the Connector
- The Connector must have an outbound connection to each Kubernetes cluster over port 443

The simplest way to provide this connectivity is to deploy the Connector and Cloud Volumes ONTAP in the same VPC as the Kubernetes cluster. Otherwise, you need to set up a VPC peering connection between the different VPCs.

Here's an example that shows each component in the same VPC.



And here's another example that shows an EKS cluster running in a different VPC. In this example, VPC peering provides a connection between the VPC for the EKS cluster and the VPC for the Connector and Cloud Volumes ONTAP.



Set up RBAC authorization

You need to authorize the Connector role on each Kubernetes cluster so the Connector can discover and manage a cluster.

Different authorization is required to enable different functionality.

Backup and restore

Backup and restore requires only basic authorization.

Add storage classes

Expanded authorization is required to add storage classes using BlueXP and monitor the cluster for changes to the backend.

Install Astra trident

You need to provide full authorization for BlueXP to install Astra Trident.

When installing Astra Trident, BlueXP installs the Astra Trident backend and Kubernetes secret that contains the credentials Astra Trident needs to communicate with the storage cluster.

Steps

1. Create a cluster role and role binding.
    a. You can customize authorization based on your requirements.

Backup/restore

Add basic authorization to enable backup and restore for Kubernetes clusters.

```
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
```

Storage classes

Add expanded authorization to add storage classes using BlueXP.

```
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 installation

Use the command line to provide full authorization and enable BlueXP to install Astra Trident.

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

b. Apply the configuration to a cluster.

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

2. Create an identity mapping to the permissions group.

Use eksctl

Use eksctl to create an IAM identity mapping between a cluster and the IAM role for the BlueXP Connector.

Go to the eksctl documentation for full instructions.

An example is provided below.

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

Edit aws-auth

Directly edit the aws-auth ConfigMap to add RBAC access to the IAM role for the BlueXP Connector.

Go to the AWS EKS documentation for full instructions.

An example is provided below.

```
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
```

Requirements for Kubernetes clusters in Azure

You can add and manage managed Azure Kubernetes clusters (AKS) and self-managed Kubernetes clusters in Azure using BlueXP. Before you can add the clusters to BlueXP, ensure the following requirements are met.

This topic uses *Kubernetes cluster* where configuration is the same for AKS and self-managed Kubernetes clusters. The cluster type is specified where configuration differs.

Requirements

Astra Trident

One of the four most recent versions of Astra Trident is required. You can install or upgrade Astra Trident directly from BlueXP. You should review the prerequisites prior to installing Astra Trident.

Cloud Volumes ONTAP

Cloud Volumes ONTAP must be set up as backend storage for the cluster. Go to the Astra Trident docs for configuration steps.

BlueXP Connector

A Connector must be running in Azure with the required permissions. Learn more below.

Network connectivity

Network connectivity is required between the Kubernetes cluster and the Connector and between the Kubernetes cluster and Cloud Volumes ONTAP. Learn more below.

RBAC authorization

BlueXP supports RBAC-enabled clusters with and without Active Directory. The BlueXP Connector role must be authorized on each Azure cluster. Learn more below.

Prepare a Connector

A BlueXP Connector in Azure is required to discover and manage Kubernetes clusters. You'll need to create a new Connector or use an existing Connector that has the required permissions.

Create a new Connector

Follow the steps in one of the links below.

- Create a Connector from BlueXP (recommended)
- Create a Connector from the Azure Marketplace
- Install the Connector on an existing Linux host

Add the required permissions to an existing Connector (to discover a managed AKS cluster)

If you want to discover a managed AKS cluster, you might need to modify the custom role for the Connector to provide the permissions.

Steps

1. Identify the role assigned to the Connector virtual machine:
    a. In the Azure portal, open the Virtual machines service.
    b. Select the Connector virtual machine.
    c. Under Settings, select **Identity**.
    d. Click **Azure role assignments**.
    e. Make note of the custom role assigned to the Connector virtual machine.

2. Update the custom role:
    a. In the Azure portal, open your Azure subscription.
    b. Click **Access control (IAM) > Roles**.
    c. Click the ellipsis (…) for the custom role and then click **Edit**.
    d. Click JSON and add the following permissions:

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

    e. Click **Review + update** and then click **Update**.

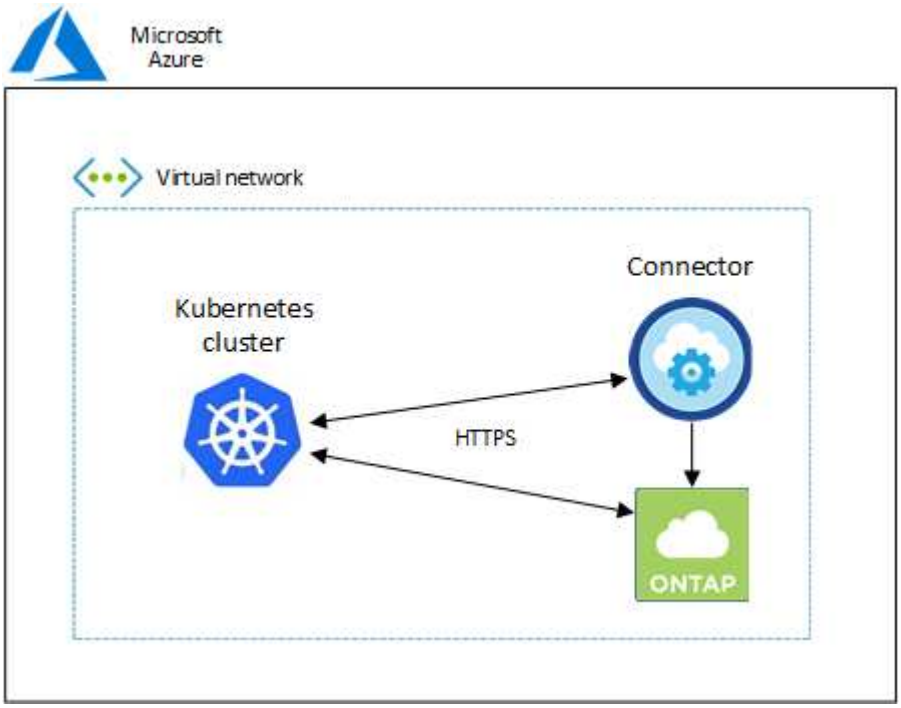
Review networking requirements

You need to provide network connectivity between the Kubernetes cluster and the Connector and between the Kubernetes cluster and the Cloud Volumes ONTAP system that provides backend storage to the cluster.

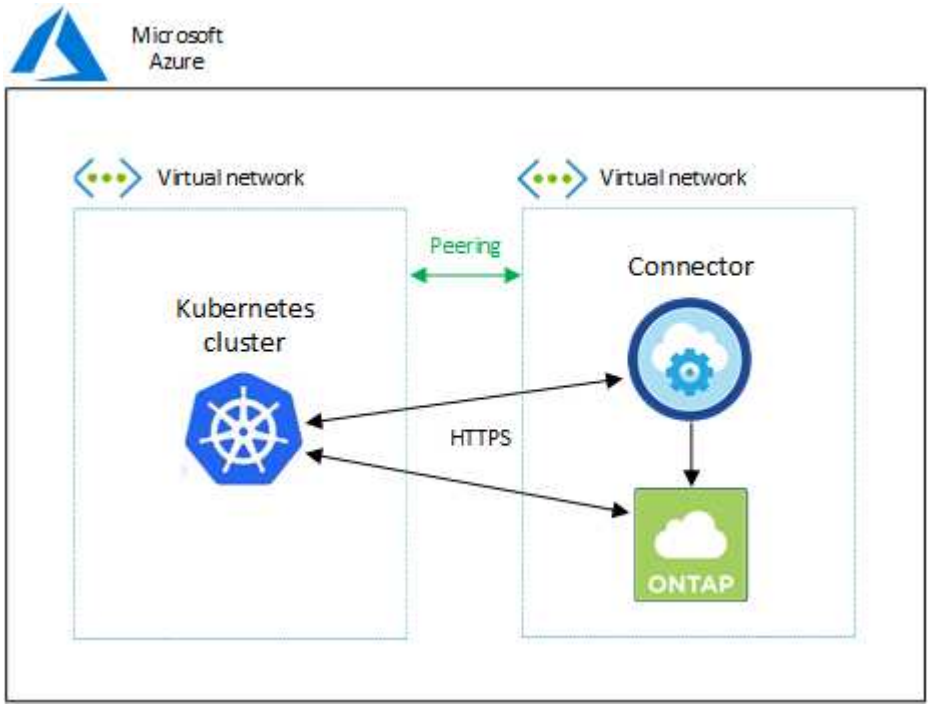
- Each Kubernetes cluster must have an inbound connection from the Connector
- The Connector must have an outbound connection to each Kubernetes cluster over port 443

The simplest way to provide this connectivity is to deploy the Connector and Cloud Volumes ONTAP in the same VNet as the Kubernetes cluster. Otherwise, you need to set up a peering connection between the different VNets.

Here's an example that shows each component in the same VNet.



And here's another example that shows a Kubernetes cluster running in a different VNet. In this example, peering provides a connection between the VNet for the Kubernetes cluster and the VNet for the Connector and Cloud Volumes ONTAP.



Set up RBAC authorization

RBAC validation occurs only on Kubernetes clusters with Active Directory (AD) enabled. Kubernetes clusters without AD will pass validation automatically.

You need authorize the Connector role on each Kubernetes cluster so the Connector can discover and manage a cluster.

Backup and restore

Backup and restore requires only basic authorization.

Add storage classes

Expanded authorization is required to add storage classes using BlueXP and monitor the cluster for changes to the backend.

Install Astra trident

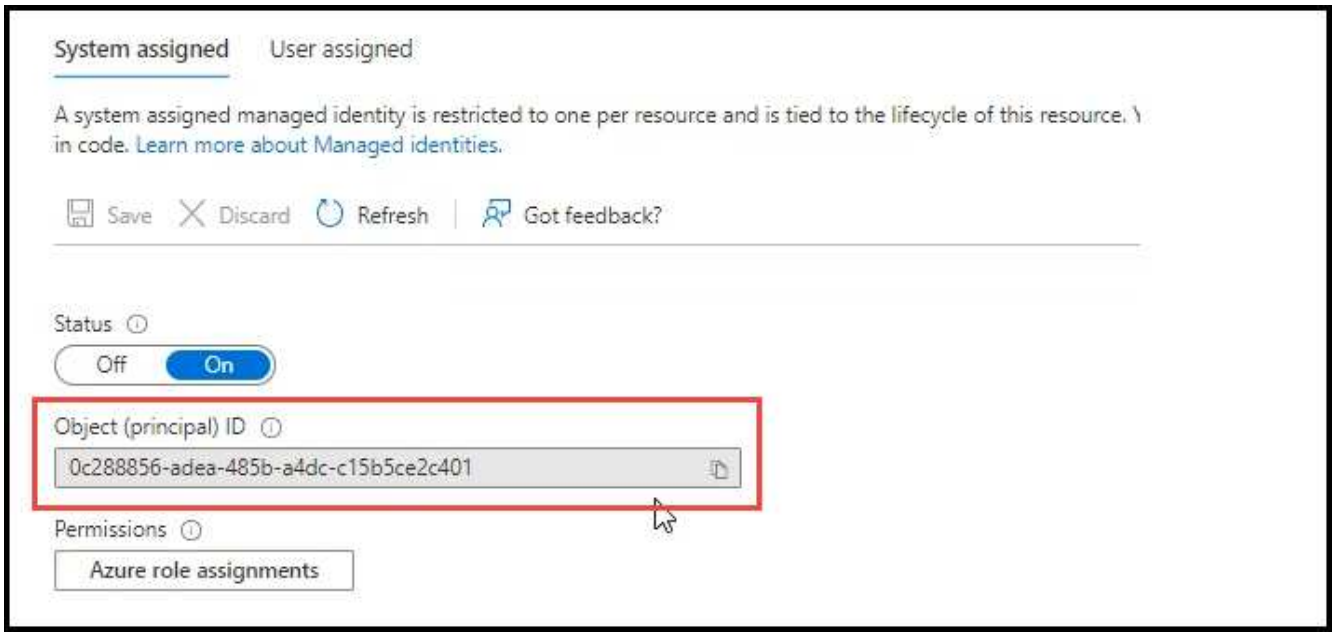
You need to provide full authorization for BlueXP to install Astra Trident.

When installing Astra Trident, BlueXP installs the Astra Trident backend and Kubernetes secret that contains the credentials Astra Trident needs to communicate with the storage cluster.

Before you begin

Your RBAC `subjects: name:` configuration varies slightly based on your Kubernetes cluster type.

- If you are deploying a **managed AKS cluster**, you need the Object ID for the system-assigned managed identity for the Connector. This ID is available in Azure management portal.



- If you are deploying a **self-managed Kubernetes cluster**, you need the username of any authorized user.

Steps

Create a cluster role and role binding.

1. You can customize authorization based on your requirements.

Backup/restore

Add basic authorization to enable backup and restore for Kubernetes clusters.

Replace the `subjects: kind:` variable with your username and `subjects: name:` with either the Object ID for the system-assigned managed identity or username of any authorized user as described above.

```
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
```

Storage classes

Add expanded authorization to add storage classes using BlueXP.

Replace the `subjects: kind:` variable with your username and `subjects: user:` with either the Object ID for the system-assigned managed identity or username of any authorized user as described above.

```
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 installation

Use the command line to provide full authorization and enable BlueXP to install Astra Trident.

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

2. Apply the configuration to a cluster.

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

Requirements for Kubernetes clusters in Google Cloud

You can add and manage managed Google Kubernetes Engine (GKE) clusters and self-managed Kubernetes clusters in Google using BlueXP. Before you can add the clusters to BlueXP, ensure the following requirements are met.

This topic uses *Kubernetes cluster* where configuration is the same for GKE and self-managed Kubernetes clusters. The cluster type is specified where configuration differs.

Requirements

Astra Trident

One of the four most recent versions of Astra Trident is required. You can install or upgrade Astra Trident directly from BlueXP. You should review the prerequisites prior to installing Astra Trident

Cloud Volumes ONTAP

Cloud Volumes ONTAP must be in BlueXP under the same tenancy account, workspace, and Connector as the Kubernetes cluster. Go to the Astra Trident docs for configuration steps.

BlueXP Connector

A Connector must be running in Google with the required permissions. Learn more below.

Network connectivity

Network connectivity is required between the Kubernetes cluster and the Connector and between the Kubernetes cluster and Cloud Volumes ONTAP. Learn more below.

RBAC authorization

BlueXP supports RBAC-enabled clusters with and without Active Directory. The BlueXP Connector role must be authorized on each GKE cluster. Learn more below.

Prepare a Connector

A BlueXP Connector in Google is required to discover and manage Kubernetes clusters. You'll need to create a new Connector or use an existing Connector that has the required permissions.

Create a new Connector

Follow the steps in one of the links below.

- Create a Connector from BlueXP (recommended)
- Install the Connector on an existing Linux host

Add the required permissions to an existing Connector (to discover a managed GKE cluster)

If you want to discover a managed GKE cluster, you might need to modify the custom role for the Connector to provide the permissions.

Steps

1. In Cloud Console, go to the **Roles** page.
2. Using the drop-down list at the top of the page, select the project or organization that contains the role that you want to edit.
3. Click a custom role.
4. Click **Edit Role** to update the role's permissions.
5. Click **Add Permissions** to add the following new permissions to the role.

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

6. Click **Update** to save the edited role.

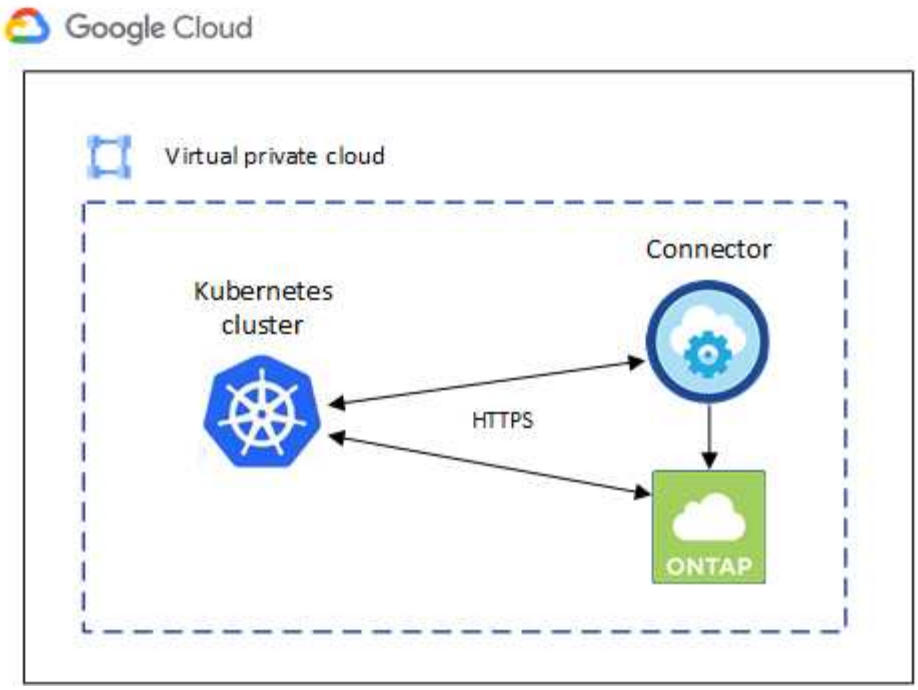
Review networking requirements

You need to provide network connectivity between the Kubernetes cluster and the Connector and between the Kubernetes cluster and the Cloud Volumes ONTAP system that provides backend storage to the cluster.

- Each Kubernetes cluster must have an inbound connection from the Connector
- The Connector must have an outbound connection to each Kubernetes cluster over port 443

The simplest way to provide this connectivity is to deploy the Connector and Cloud Volumes ONTAP in the same VPC as the Kubernetes cluster. Otherwise, you need to set up a peering connection between the different VPC.

Here's an example that shows each component in the same VPC.



Set up RBAC authorization

RBAC validation occurs only on Kubernetes clusters with Active Directory (AD) enabled. Kubernetes clusters without AD will pass validation automatically.

You need authorize the Connector role on each Kubernetes cluster so the Connector can discover and manage a cluster.

Backup and restore

Backup and restore requires only basic authorization.

Add storage classes

Expanded authorization is required to add storage classes using BlueXP and monitor the cluster for changes to the backend.

Install Astra trident

You need to provide full authorization for BlueXP to install Astra Trident.

When installing Astra Trident, BlueXP installs the Astra Trident backend and Kubernetes secret that contains the credentials Astra Trident needs to communicate with the storage cluster.

Before you begin

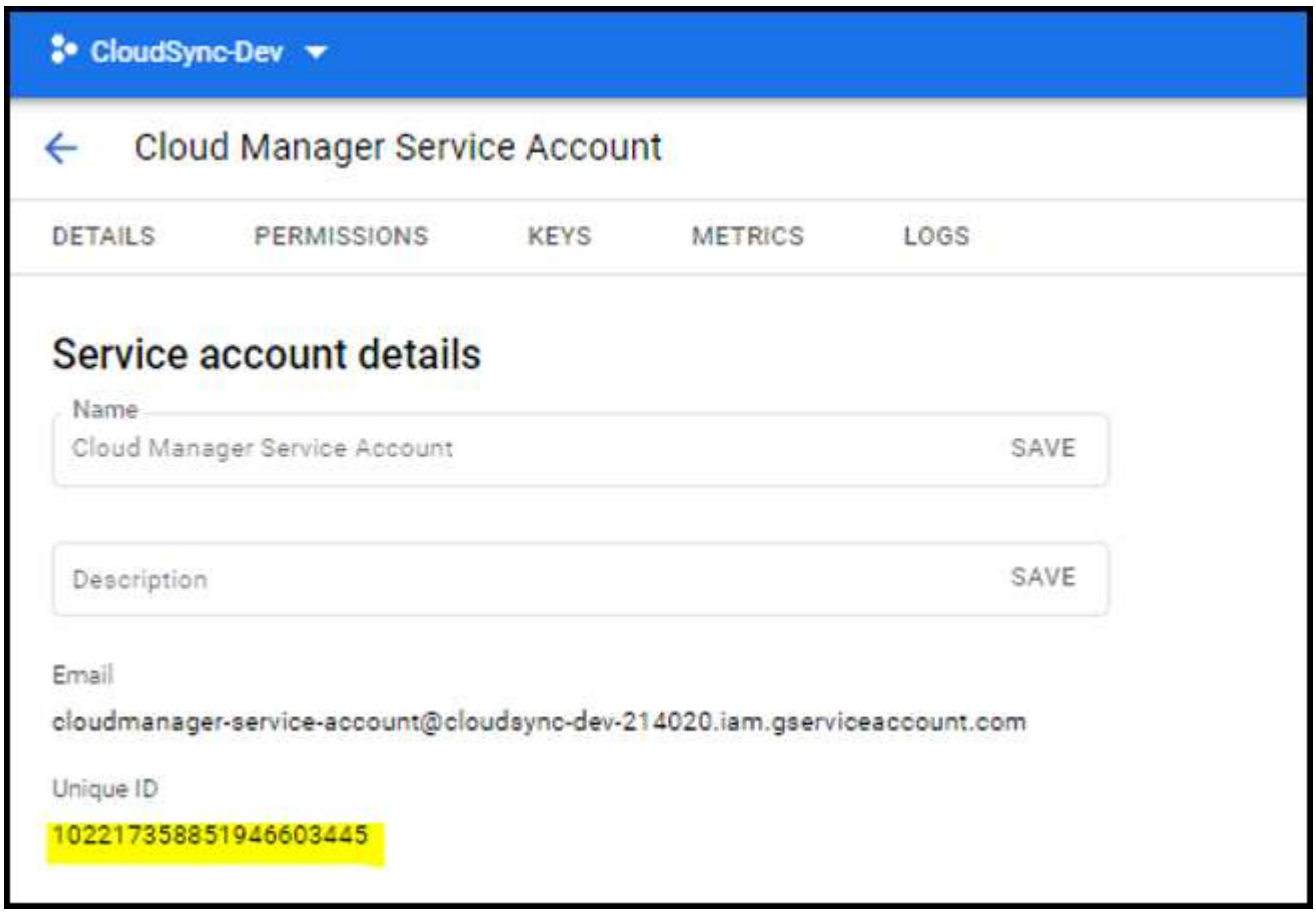
To configure `subjects: name:` in the YAML file, you need to know the BlueXP Unique ID.

You can find the unique ID one of two ways:

- Using the command:

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

- In the Service Account Details on the Cloud Console.



Steps

Create a cluster role and role binding.

1. You can customize authorization based on your requirements.

Backup/restore

Add basic authorization to enable backup and restore for Kubernetes clusters.

Replace the `subjects: kind:` variable with your username and `subjects: name:` with the unique ID for the authorized service account.

```
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
```

Storage classes

Add expanded authorization to add storage classes using BlueXP.

Replace the `subjects: kind:` variable with your username and `subjects: user:` with the unique ID for the authorized service account.

```
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 installation

Use the command line to provide full authorization and enable BlueXP to install Astra Trident.

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

2. Apply the configuration to a cluster.

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

Requirements for Kubernetes clusters in OpenShift

You can add and manage self-managed OpenShift Kubernetes clusters using BlueXP. Before you can add the clusters to BlueXP, ensure the following requirements are met.

Requirements

Astra Trident

One of the four most recent versions of Astra Trident is required. You can install or upgrade Astra Trident directly from BlueXP. You should review the prerequisites prior to installing Astra Trident.

Cloud Volumes ONTAP

Cloud Volumes ONTAP must be set up as backend storage for the cluster. Go to the Astra Trident docs for configuration steps.

BlueXP Connector

A BlueXP Connector is required to import and manage Kubernetes clusters. You'll need to create a new Connector or use an existing Connector that has the required permissions for your Cloud provider:

- AWS Connector
- Azure Connector
- Google Cloud Connector

Network connectivity

Network connectivity is required between the Kubernetes cluster and the Connector and between the Kubernetes cluster and Cloud Volumes ONTAP.

Kubernetes configuration file (kubeconfig) with RBAC authorization

To import OpenShift clusters, you need a kubeconfig file with the RBAC authorization required to enable different functionality. Create a kubeconfig file.

- Backup and restore: Backup and restore requires only basic authorization.
- Add storage classes: Expanded authorization is required to add storage classes using BlueXP and monitor the cluster for changes to the backend.
- Install Astra Trident: You need to provide full authorization for BlueXP to install Astra Trident.

> When installing Astra Trident, BlueXP installs the Astra Trident backend and Kubernetes secret that contains the credentials Astra Trident needs to communicate with the storage cluster.

Create a kubeconfig file

Using the OpenShift CLI, create a kubeconfig file to import to BlueXP.

Steps

1. Log in to the OpenShift CLI using `oc login` on a public URL with an administrative user.
2. Create a service account as follows:
   a. Create a service account file called `oc-service-account.yaml`.

      Adjust the name and namespace as needed. If changes are made here, you should apply the same changes in the following steps.

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

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

   b. Apply the service account:

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

3. Create a custom role binding based on your authorization requirements.
   a. Create a `ClusterRoleBinding` file called `oc-clusterrolebinding.yaml`.

      ```
      oc-clusterrolebinding.yaml
      ```

   b. Configure RBAC authorization as needed for your cluster.

Backup/restore

Add basic authorization to enable backup and restore for Kubernetes clusters.

```
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
```

Storage classes

Add expanded authorization to add storage classes using BlueXP.

```
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 installation

Grant full admin authorization and enable BlueXP to install Astra Trident.

```
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. Apply the cluster role binding:

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

4. List the service account secrets, replacing `<context>` with the correct context for your installation:

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

The end of the output should look similar to the following:

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

The indices for each element in the `secrets` array begin with 0. In the above example, the index for `oc-service-account-dockercfg-vhz87` would be 0 and the index for `oc-service-account-token-r59kr` would be 1. In your output, make note of the index for the service account name that has the word "token" in it.

5. Generate the kubeconfig as follows:

a. Create a `create-kubeconfig.sh` file. Replace `TOKEN_INDEX` in the beginning of the following script with the correct value.

```
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. Source the commands to apply them to your Kubernetes cluster.

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

Result

You will use the resulting `kubeconfig-sa` file to add an OpenShift cluster to BlueXP.

Copyright information

Trademark information