Prerequisites for adding a cluster

Astra Control Center

Michael Wallis, amitha, Dave Bagwell
September 22, 2021

Table of Contents

Prerequisites for adding a cluster

You should ensure that the prerequisite conditions are met before you add a cluster. You should also run the eligibility checks to ensure that your cluster is ready to be added to Astra Control Center.

What you'll need before you add a cluster

- A cluster running OpenShift 4.6 or 4.7, which has Trident StorageClasses backed by ONTAP 9.5 or later.
  - One or more worker nodes with at least 1GB RAM available for running telemetry services.

> If you plan to add a second OpenShift 4.6 or 4.7 cluster as a managed compute resource, you should ensure that the Trident Volume Snapshot feature is enabled. See the official Trident instructions to enable and test Volume Snapshots with Trident.

- The superuser and user ID set on the backing ONTAP system to back up and restore apps with Astra Control Center (ACC). Run the following commands in the ONTAP command line:
  `export policy rule modify -vserver svm0 -policyname default -ruleindex 1 -superuser sys`
  `export-policy rule modify -policyname default -ruleindex 1 -anon 65534` (this is the default value)

Run eligibility checks

Run the following eligibility checks to ensure that your cluster is ready to be added to Astra Control Center.

Steps

1. Check the Trident version.

   ```
   kubectl get tridentversions -n trident
   ```

   If Trident exists, you see output similar to the following:

   ```
   NAME      VERSION
   trident   21.04.0
   ```

   If Trident does not exist, you see output similar to the following:

   ```
   error: the server doesn't have a resource type "tridentversions"
   ```

   > If Trident is not installed or the installed version is not the latest, you need to install the latest version of Trident before proceeding. See the Trident documentation for instructions.

2. Check if the storage classes are using the supported Trident drivers. The provisioner name should be `csi.trident.netapp.io`. See the following example:

```
kubectl get storageClass -A
NAME                   PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io          Delete
Immediate           true                   5d23h
thin                   kubernetes.io/vsphere-volume   Delete
Immediate           false                  6d
```

Create an admin-role kubeconfig

Ensure that you have the following on your machine before you do the steps:

- `kubectl` v1.19 or later installed
- An active kubeconfig with cluster admin rights for the active context

Steps

1. Create a service account as follows:
   a. Create a service account file called `astracontrol-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.

      ```
      <strong>astracontrol-service-account.yaml</strong>
      ```

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

   b. Apply the service account:

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

2. Grant cluster admin permissions as follows:
   a. Create a `ClusterRoleBinding` file called `astracontrol-clusterrolebinding.yaml`.

      Adjust any names and namespaces modified when creating the service account as needed.

      ```
      <strong>astracontrol-clusterrolebinding.yaml</strong>
      ```

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

b. Apply the cluster role binding:

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

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

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

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

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

The indices for each element in the `secrets` array begin with 0. In the above example, the index for `astracontrol-service-account-dockercfg-vhz87` would be 0 and the index for `astracontrol-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.

4. Generate the kubeconfig as follows:

a. Create a `create-kubeconfig.sh` file. If the token index you noted in the previous step was not 0, replace the value for `TOKEN_INDEX` in the beginning of the following script with the correct value.

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment. Replace the value for
```

```
TOKEN_INDEX from
# the output in the previous step if it was not 0. If you didn't
change anything
# else above, don't change anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'
TOKEN_INDEX=0

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
```

5. (**Optional**) Rename the kubeconfig to a meaningful name for your cluster. Protect your cluster credential.

```
chmod 700 create-kubeconfig.sh
mv kubeconfig-sa.txt YOUR_CLUSTER_NAME_kubeconfig
```

What's next?

Now that you've verified that the prerequisites are met, you're ready to add a cluster.

Find more information

- Trident documentation
- Use the Astra API