ArgoCD: Creating a connection to a remote cluster

This page contains information about how to get ArgoCD (a.k.a OpenShift GitOps) to deploy to another (read remote) OpenShift /Kubernetes cluster.

On this page

Handy Links

ArgoCD Documentation

- Clusters Declarative Setup - Argo CD

Overview

When installed on a OpenShift/Kubernetes cluster ArgoCD (a.k.a. OpenShift GitOps) is given sufficient privileges to deploy to the cluster it is running in. This cluster is often referred to as '*in-cluster*' and uses the URL/Server of https://kubernetes.default.svc

To enable ArgoCD to be able to manage resources in another cluster (which is the pattern used in ACIC's implementation where the Hub clusters manage resources in the Workload clusters) additional configuration is required.

Process of creating an ArgoCD cluster connection

Creating a connection to another cluster involves:

1. On the target cluster:
   a. Create the Namespace - to store the resources
   b. Create the ClusterRole - with sufficient privileges to manage all of the required resources
   c. Create the ServiceAccount (and Secret) - which will be used by ArgoCD to access the target cluster
   d. Create the ClusterRoleBinding - to assign the ServiceAccount the ClusterRole
   e. Populate the template with the ServiceAccount credentials - the completed template is used in an ArgoCD Secret to obtain the ServiceAccount credentials
2. On the cluster with ArgoCD:
   a. Create a Secret - which contains the details of the clusters' ServiceAccount, URL/Server, and Name
   b. Use the cluster name in your ArgoCD Application manifest
   c. (If using a protected repository) Create an ArgoCD repository Secret - so that ArgoCD can access the source files

Prerequisits

For Step 1 (target cluster) - you will need:

- cluster-administrator access to the cluster
- the *oc* (or *kubectl*) command line tool
- a linux machine with *jq* installed

For Step 2 (ArgoCD cluster) - you will need:

- the linux machine used in Step 1 **or** a copy of the populated template (*config* file)
- cluster-administrator access to the cluster
- the *oc* (or *kubectl*) command line tool

On the target cluster

Create the Namespace

Create the '*cluster-config-gitops*' namespace which will be used to store the resources.

Below is an example of a YAML file which can be used to create the namespace:

00-cluster-config-gitops-namespace.yaml

```
#
# Description: The namespace which contains the resources related to remote cluster management
#
# To apply/create:
#   oc apply -f ./00-cluster-config-gitops-namespace.yaml
#
---
apiVersion: v1
kind: Namespace
metadata:
  name: cluster-config-gitops
  annotations:
    # Prevent the namespace from being deleted if the (ArgoCD) application is deleted
    argocd.argoproj.io/sync-options: Delete=false
```

Create the ClusterRole

Create a ClusterRole with sufficient privileges to manage all of the required resources.

Below is an example of a YAML file which can be used to create the ClusterRole:

10-cluster-config-gitops-clusterrole.yaml

```
#
# Description: This ClusterRole is used by ArgoCD on the Hub clusters (via a ServiceAccount)
#               to deploy cluster configuration to a target cluster
#
# Note: You may need to add to this ClusterRole if you want ArgoCD to manage additional resources or
CustomResourceDefinitions (CRDs)
#
# To apply/create:
#    oc apply -f ./10-cluster-config-gitops-clusterrole.yaml
#
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-config-gitops
rules:
  - apiGroups:
      - '*'
    resources:
      - '*'
    verbs:
      - get
      - list
      - watch
  - nonResourceURLs:
      - '*'
    verbs:
      - get
      - list
```

```
- apiGroups:
    - operators.coreos.com
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - operator.openshift.io
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - user.openshift.io
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - config.openshift.io
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - console.openshift.io
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - ""
  resources:
    - namespaces
    - persistentvolumeclaims
    - persistentvolumes
    - configmaps
    - services
    - pods
  verbs:
    - '*'
- apiGroups:
    - "apps"
  resources:
    - deployments
  verbs:
    - '*'
- apiGroups:
    - "networking.k8s.io"
  resources:
    - ingresses
  verbs:
    - '*'
- apiGroups:
    - rbac.authorization.k8s.io
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - storage.k8s.io
  resources:
    - '*'
  verbs:
    - '*'
- apiGroups:
    - machine.openshift.io
  resources:
    - '*'
  verbs:
    - '*'
```

```
  - apiGroups:
      - machineconfiguration.openshift.io
    resources:
      - '*'
    verbs:
      - '*'
  - apiGroups:
      - compliance.openshift.io
    resources:
      - scansettingbindings
    verbs:
      - '*'
  - apiGroups:
      - cert-manager.io
    resources:
      - '*'
    verbs:
      - '*'
  - apiGroups:
      - ""
    resources:
      - 'secrets'
      - 'resourcequotas'
      - 'serviceaccounts'
    verbs:
      - '*'
  - apiGroups:
      - batch
    resources:
      - jobs
      - cronjobs
      - cronjobs/finalizers
    verbs:
      - '*'
  - apiGroups:
      - platform.stackrox.io
    resources:
      - securedclusters
    verbs:
      - '*'
  - apiGroups:
      - route.openshift.io
    resources:
      - routes
    verbs:
      - '*'
---
```

Create the ServiceAccount (and Secret)

1. Create the ServiceAccount which will be used by ArgoCD to access the target cluster
2. Create the Secret which (upon creation) will be populated with the Certificate and Bearer Token for the ServiceAccount.

Below is an example of a YAML file which can be used to create the ServiceAccount and Secret:

20-cluster-config-gitops-serviceaccount.yaml

```
#
# Description: This ServiceAccount is used by ArgoCD on the Hub clusters to deploy cluster configuration and
applications
#               to this cluster
#
# To apply/create:
#   oc apply -f ./20-cluster-config-gitops-serviceaccount.yaml
#
---

kind: ServiceAccount
apiVersion: v1
metadata:
  name: cluster-config-gitops-sa
  namespace: cluster-config-gitops
---
kind: Secret
apiVersion: v1
metadata:
  name: cluster-config-gitops-sa-token
  namespace: cluster-config-gitops
  annotations:
    kubernetes.io/service-account.name: cluster-config-gitops-sa
type: kubernetes.io/service-account-token
---
```

Create the ClusterRoleBinding

Create the ClusterRoleBinding to assign the ServiceAccount the '*cluster-config-gitops*' ClusterRole.

Below is an example of a YAML file which can be used to create the ClusterRoleBinding:

21-cluster-config-gitops-crb.yaml

```
#
# Description: Binds the 'cluster-config-gitops-sa' ServiceAccount (used by the Hub cluster to externally
configure the cluster)
#               to the 'cluster-config-gitops' ClusterRole to give the ServiceAccount the permissions it
needs.
#
# To apply/create:
#   oc apply -f ./cluster-config/cluster-gitops/21-cluster-config-gitops-crb.yaml
#
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-config-gitops-crb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-config-gitops
subjects:
- kind: ServiceAccount
  name: cluster-config-gitops-sa
  namespace: cluster-config-gitops
```

Populate the (ArgoCD cluster secret) template with the ServiceAccount credentials

This step extracts the ServiceAccount credentials from the *service-account-token* Secret on the target cluster (created above) and uses a template along with some other details to create a Secret in the ArgoCD cluster.

Complete this step on a linux machine!

This step uses several linux command line tools including:

- oc
- jq
- base64

It is possible to populate the template using a Windows machine, but for simplicity a linux machine is preferred.

1. Create the template file (`argo-cd-cluster-auth-template.json`) with the following contents:
    a. **argo-cd-cluster-auth-template.json**

```
{
    "bearerToken": "<SERVICE_ACCOUNNT_BEARER_TOKEN>",
    "tlsClientConfig": {
      "insecure": false,
      "caData": "<SERVICE_ACCOUNT_TOKEN_ca.crt>"
    }
}
```

2. With your *oc* context set to the **target** cluster, populate the template by running:
    a.
```
SERVICE_ACCOUNT=$(oc -n cluster-config-gitops get secret cluster-config-gitops-sa-token \
    -o "jsonpath={.data['ca\.crt']}") && \

BEARER_TOKEN=$(oc -n cluster-config-gitops get secret cluster-config-gitops-sa-token \
    -o "jsonpath={.data['token']}" | base64 -d) && \

jq --arg bearerToken "$BEARER_TOKEN" --arg caData "$SERVICE_ACCOUNT" \
    '.bearerToken = $bearerToken | .tlsClientConfig.caData = $caData' \
    ./argo-cd-cluster-auth-template.json > /tmp/config
```

3. Switch context to the ArgoCD cluster:
    a.
```
oc config use-context <YOUR_ARGOCD_CLUSTER_CONTEXT_NAME>
```

4. On the ArgoCD cluster, create the Secret (see below for details).

Delete the 'config' file!

Be sure to DELETE the 'config' file when you are done!

This file contains credentials that have a lot of privileges within the target cluster.

On the cluster with ArgoCD

Create a Secret

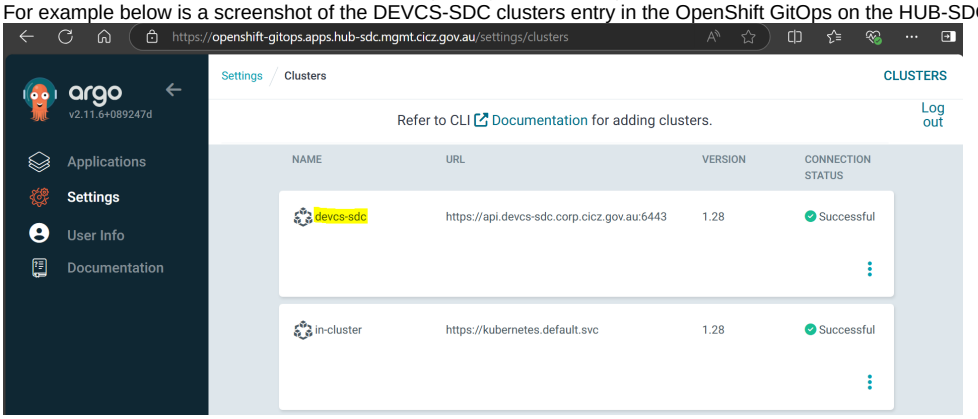
On the ArgoCD cluster create a Secret which contains the details of the clusters' ServiceAccount, URL/Server, and Name

Complete this step on the same linux machine used above!

The instructions below assume that you are using the same linux machine used to populate the template (*config* file) with the ServiceAccount credentials used above.

If you are using a different machine you will need a copy of the populated template (*config* file) to complete these instructions.

1. Using the contents of the populated template (*config* file), cluster name and the URL of the OpenShift/Kubernetes API create a Secret, label it and delete the *config* file:
    a.
```
oc -n openshift-gitops create secret generic <CLUSTER_NAME>-cluster \
    --from-file=/tmp/config \
        --from-literal=name=<CLUSTER_NAME> \
    --from-literal=server=https://<CLUSTER_API_URL>:6443 && \

oc -n openshift-gitops label secret <CLUSTER_NAME>-cluster argocd.argoproj.io/secret-
type=cluster && \
rm /tmp/config
```
2. The target cluster should now be listed in the ArgoCD UI (Settings --> Clusters)
    a. For example below is a screenshot of the DEVCS-SDC clusters entry in the OpenShift GitOps on the HUB-SDC cluster:
    b. 

3. Extract and store the cluster Secret in Keypass
    a. TODO
4. Delete the *config* file and Secret from the machine!

Delete the 'config' file and Secret!

Be sure to DELETE the 'config' file and Secret when you are done!

These files contains credentials that have a lot of privileges within the target cluster.

Use the cluster name in your ArgoCD Application manifest

The ArgoCD Application needs to reference the target clusters' name (preferred) in the destination stansa.

For example, the ArgoCD Application below will deploy the resources to the `*devcs-sdc*` cluster:

Example ArgoCD Application

```
#
# Description: This partial ArgoCD Application is an example of how to set the destination cluster for the
deployment resources.
#
---
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: devcs-example-app
  namespace: openshift-gitops
spec:
  # The details of the 'target' cluster (i.e. where to deploy the resources)
  destination:
        # The name of the cluster. This needs to match name value in the ArgoCD cluster Secret
    name: devcs-sdc
    # The namespace within the 'devcs-sdc' cluster to deploy the resources
    namespace: example-app
  source:
        # The source of the deployment resources
    repoURL: 'http://psdas-vpd-pst01.crimtracagency.corporate:7990/scm/dev/dev-argocd-applications.git'
        ...
```

For more information see:

- Application Specification Reference - Argo CD - Declarative GitOps CD for Kubernetes
- DevOps - Confluence Page

(If using a protected repository) Create an ArgoCD repository Secret

If the deployment resources reside within a protected repository (i.e. need credentials to access), your ArgoCD instance will need a Repository Secret.

For more information refer to: ArgoCD: Using Protected git Repositories