Red Hat OpenShift GitOps 1.12

Access control and user management

Configuring user authentication and access controls for users and namespaces

Last Updated: 2024-05-14

Red Hat OpenShift GitOps 1.12 Access control and user management

Configuring user authentication and access controls for users and namespaces

Legal Notice

Abstract

This document provides instructions for changing and managing user level access and resource requests. It also discusses how to configure role-based access control and single sign-on authentication providers to manage multiple users, permissions, Argo CD resources, and instances in the cluster.

Table of Contents

CHAPTER 1. CONFIGURING ARGO CD RBAC 3
1.1. CONFIGURING USER LEVEL ACCESS 3
1.2. MODIFYING RHSSO RESOURCE REQUESTS/LIMITS 3

CHAPTER 2. CONFIGURING SSO FOR ARGO CD USING DEX 5
2.1. CONFIGURATION TO ENABLE THE DEX OPENSHIFT OAUTH CONNECTOR 5
2.1.1. Mapping users to specific roles 5
2.2. DISABLING DEX BY REPLACING .SPEC.SSO 6

CHAPTER 3. CONFIGURING SSO FOR ARGO CD USING KEYCLOAK 7
3.1. PREREQUISITES 7
3.2. CONFIGURING A NEW CLIENT IN KEYCLOAK 7
3.3. LOGGING IN TO KEYCLOAK 8
3.4. UNINSTALLING KEYCLOAK 9

CHAPTER 1. CONFIGURING ARGO CD RBAC

By default, if you are logged in to Argo CD using Red Hat SSO (RH SSO), you are a read-only user. You can change and manage the user level access.

1.1. CONFIGURING USER LEVEL ACCESS

To manage and modify the user level access, configure the role-based access control (RBAC) section in the Argo CD custom resource (CR).

Procedure

1. Edit the **argocd** CR:

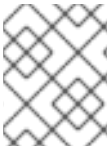
```
$ oc edit argocd [argocd-instance-name] -n [namespace]
```

Output

```
metadata
...
...
  rbac:
    policy: 'g, rbacsystem:cluster-admins, role:admin'
    scopes: '[groups]'
```

2. Add the **policy** configuration to the **rbac** section and add the **name** and the desired **role** to be applied to the user:

```
metadata
...
...
rbac:
    policy: g, <name>, role:<admin>
    scopes: '[groups]'
```



NOTE

Currently, RHSSO cannot read the group information of Red Hat OpenShift GitOps users. Therefore, configure the RBAC at the user level.

1.2. MODIFYING RHSSO RESOURCE REQUESTS/LIMITS

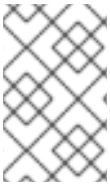
By default, the RHSSO container is created with resource requests and limitations. You can change and manage the resource requests.

Resource	Requests	Limits
CPU	500	1000m
Memory	512 Mi	1024 Mi

Procedure

- Modify the default resource requirements patching the Argo CD custom resource (CR):

```
$ oc -n openshift-gitops patch argocd openshift-gitops --type='json' -p='[{"op": "add", "path": "/spec/sso", "value": {"provider": "keycloak", "resources": {"requests": {"cpu": "512m", "memory": "512Mi"}, "limits": {"cpu": "1024m", "memory": "1024Mi"}} }}]'
```



NOTE

RHSSO created by the Red Hat OpenShift GitOps only persists the changes that are made by the operator. If the RHSSO restarts, any additional configuration created by the Admin in RHSSO is deleted.

CHAPTER 2. CONFIGURING SSO FOR ARGO CD USING DEX

After the Red Hat OpenShift GitOps Operator is installed, Argo CD automatically creates a user with **admin** permissions. To manage multiple users, cluster administrators can use Argo CD to configure Single Sign-On (SSO).

NOTE

The **spec.dex** parameter in the ArgoCD CR is no longer supported from Red Hat OpenShift GitOps v1.10.0 onwards. Consider using the **.spec.sso** parameter instead.

2.1. CONFIGURATION TO ENABLE THE DEX OPENSHIFT OAUTH CONNECTOR

Dex is installed by default for all the Argo CD instances created by the Operator. You can configure Red Hat OpenShift GitOps to use Dex as the SSO authentication provider by setting the **.spec.sso** parameter.

Dex uses the users and groups defined within OpenShift Container Platform by checking the **OAuth** server provided by the platform.

Procedure

- To enable Dex, set the **.spec.sso.provider** parameter to **dex** in the YAML resource of the Operator:

```
# ...
spec:
  sso:
    provider: dex
    dex:
      openShiftOAuth: true 1
# ...
```

1 The **openShiftOAuth** property triggers the Operator to automatically configure the built-in OpenShift Container Platform **OAuth** server when the value is set to **true**.

2.1.1. Mapping users to specific roles

Argo CD cannot map users to specific roles if they have a direct **ClusterRoleBinding** role. You can manually change the role as **role:admin** on SSO through OpenShift.

Procedure

1. Create a group named **cluster-admins**.

```
$ oc adm groups new cluster-admins
```

2. Add the user to the group.

```
$ oc adm groups add-users cluster-admins USER
```

3. Apply the **cluster-admin ClusterRole** to the group:

```
$ oc adm policy add-cluster-role-to-group cluster-admin cluster-admins
```

2.2. DISABLING DEX BY REPLACING .SPEC.SSO

- To disable dex, either remove the **spec.sso** element from the Argo CD custom resource or specify a different SSO provider.

CHAPTER 3. CONFIGURING SSO FOR ARGO CD USING KEYCLOAK

After the Red Hat OpenShift GitOps Operator is installed, Argo CD automatically creates a user with **admin** permissions. To manage multiple users, cluster administrators can use Argo CD to configure Single Sign-On (SSO).

3.1. PREREQUISITES

- Red Hat SSO is installed on the cluster.
- The Red Hat OpenShift GitOps Operator is installed on your OpenShift Container Platform cluster.
- Argo CD is installed on the cluster.

3.2. CONFIGURING A NEW CLIENT IN KEYCLOAK

Dex is installed by default for all the Argo CD instances created by the Operator. However, you can delete the Dex configuration and add Keycloak instead to log in to Argo CD using your OpenShift credentials. Keycloak acts as an identity broker between Argo CD and OpenShift.

Procedure

To configure Keycloak, follow these steps:

1. Delete the Dex configuration by removing the **.spec.sso.dex** parameter from the Argo CD custom resource (CR), and save the CR:

```
dex:
   openShiftOAuth: true
   resources:
    limits:
      cpu:
      memory:
    requests:
      cpu:
      memory:
```

2. Set the value of the **provider** parameter to **keycloak** in the Argo CD CR.
3. Configure Keycloak by performing one of the following steps:
   - For a secure connection, set the value of the **rootCA** parameter as shown in the following example:

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec:
  sso:
```

```
    provider: keycloak
    keycloak:
      rootCA: "<PEM-encoded-root-certificate>" (1)
  server:
    route:
      enabled: true
```

(1) A custom certificate used to verify the Keycloak's TLS certificate.

The Operator reconciles changes in the **.spec.sso.keycloak.rootCA** parameter and updates the **oidc.config** parameter with the PEM encoded root certificate in the **argocd-cm** configuration map.

- For an insecure connection, leave the value of the **rootCA** parameter empty and use the **oidc.tls.insecure.skip.verify** parameter as shown below:

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec:
  extraConfig:
    oidc.tls.insecure.skip.verify: "true"
  sso:
    provider: keycloak
    keycloak:
      rootCA: ""
```

NOTE

The Keycloak instance takes 2-3 minutes to install and run.

3.3. LOGGING IN TO KEYCLOAK

Log in to the Keycloak console to manage identities or roles and define the permissions assigned to the various roles.

Prerequisites

- The default configuration of Dex is removed.
- Your Argo CD CR must be configured to use the Keycloak SSO provider.

Procedure

1. Get the Keycloak route URL for login:

```
$ oc -n argocd get route keycloak

NAME       HOST/PORT                                   PATH   SERVICES   PORT
```

```
TERMINATION   WILDCARD
keycloak    keycloak-default.apps.ci-ln-******.origin-ci-int-aws.dev.**.com          keycloak   <all>
reencrypt     None
```

2. Get the Keycloak pod name that stores the user name and password as environment variables:

```
$ oc -n argocd get pods

NAME                  READY   STATUS         RESTARTS   AGE
keycloak-1-2sjcl          1/1    Running          0        45m
```

   a. Get the Keycloak user name:

```
$ oc -n argocd exec keycloak-1-2sjcl -- "env" | grep SSO_ADMIN_USERNAME

SSO_ADMIN_USERNAME=Cqid54Ih
```

   b. Get the Keycloak password:

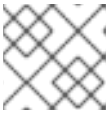
```
$ oc -n argocd exec keycloak-1-2sjcl -- "env" | grep SSO_ADMIN_PASSWORD

SSO_ADMIN_PASSWORD=GVXxHifH
```

3. On the login page, click **LOG IN VIA KEYCLOAK**

NOTE

You only see the option **LOGIN VIA KEYCLOAK** after the Keycloak instance is ready.

4. Click **Login with OpenShift**.



NOTE

Login using **kubeadmin** is not supported.

5. Enter the OpenShift credentials to log in.

6. Optional: By default, any user logged in to Argo CD has read-only access. You can manage the user level access by updating the **argocd-rbac-cm** config map:

```
policy.csv:
<name>, <email>, role:admin
```

3.4. UNINSTALLING KEYCLOAK

You can delete the Keycloak resources and their relevant configurations by removing the **SSO** field from the Argo CD Custom Resource (CR) file. After you remove the **SSO** field, the values in the file look similar to the following:

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
```

```
metadata:
  name: example-argocd
  labels:
    example: basic
spec:
  server:
    route:
      enabled: true
```

NOTE

A Keycloak application created by using this method is currently not persistent. Additional configurations created in the Argo CD Keycloak realm are deleted when the server restarts.