Configuring External Authentication Providers

6

OpenShift supports a number of different authentication providers, and you can find the complete list in the authentication documentation. One of the most commonly used authentication providers is LDAP, whether provided by Microsoft Active Directory or by other sources.

OpenShift can perform user authentication against an LDAP server, and can also configure group membership and certain RBAC attributes based on LDAP group membership.

Background: LDAP Structure

In this environment we are providing LDAP with a Red Hat Identity Management server running on Red Hat Enterprise Linux 7 running in a VM. IDM is configured with the following user groups:

- `ose-user`: Users with OpenShift access
    - Any users who should be able to log-in to OpenShift must be members of this group
    - All of the below mentioned users are in this group
- `ose-normal-dev`: Normal OpenShift users

- Regular users of OpenShift without special permissions
- Contains: `normaluser1`, `teamuser1`, `teamuser2`

- `ose-fancy-dev`: Fancy OpenShift users
  - Users of OpenShift that are granted some special privileges
  - Contains: `fancyuser1`, `fancyuser2`
- `ose-teamed-app`: Teamed app users
  - A group of users that will have access to the same OpenShift **Project**
  - Contains: `teamuser1`, `teamuser2`

Examine the master configuration

The installation of OpenShift already included setting up LDAP authentication in the inventory file. As the `root` user on the master host, examine the master config with `cat`, `less`, or your favorite editor:

```
cat /etc/origin/master/master-config.yaml
```

Look for one of the `oauthConfig` sections that looks like the following:

```
oauthConfig:
  assetPublicURL: https://master.193183262213.aws.testdriv
  grantConfig:
    method: auto
```

```
    identityProviders:
    - challenge: true
      login: true
      mappingMethod: claim (1)
      name: idm (2)
      provider:
        apiVersion: v1
        attributes: (3)
          email:
          - mail
          id:
          - dn
          name:
          - cn
          preferredUsername:
          - uid
        bindDN: uid=admin,cn=users,cn=accounts,dc=unset,dc=o
        bindPassword: ldapadmin (5)
        ca: /etc/origin/master/ipa-ca.crt (6)
        insecure: false
        kind: LDAPPasswordIdentityProvider
        url: <7>ldap://idm.internal.aws.testdrive.openshift.
...
```

Some notable fields under `identityProviders:`:

(1) `name`: The unique ID of the identity provider. It is possible to have multiple authentication providers in an OpenShift environment, and OpenShift is able to distinguish between them.

2. `mappingMethod: claim`: This section has to do with how usernames are assigned within an OpenShift cluster when multiple providers are configured. See the mapping identities to users section for more information.

3. `attributes`: This section defines the LDAP fields to iterate over and assign to the fields in the OpenShift user's "account". If any attributes are not found / not populated when searching through the list, the entire authentication fails. In this case we are creating an identity that is assocuated with the LDAP `dn`, an email address from the LDAP `mail`, a name from the LDAP `cn`, and a username from the LDAP `uid`.

4. `bindDN`: When searching LDAP, bind to the server as this user.

5. `bindPassword`: The password to use when binding for searching.

6. `ca`: The CA certificate to use for validating the SSL certificate of the LDAP server.

7. `url`: Identifies the LDAP server and the search to perform.

For more information on the specific details of LDAP authentication in OpenShift you can refer to the LDAP provider documentation.

You can find the installer configuration line that sets up LDAP

authentication by looking in `/etc/ansible/hosts` at the line that begins with `openshift_master_identity_providers=`

Syncing LDAP Groups to OpenShift Groups

In OpenShift, groups can be used to manage users and control permissions for multiple users at once. There is a section in the documentation on how to sync groups with LDAP. Syncing groups involves running a program called `groupsync` when logged into OpenShift as a user with `cluster-admin` privileges, and using a configuration file that tells OpenShift what to do with the users it finds in the various groups.

We have provided a `groupsync` configuration file for you already. Using `cat`, `less`, or your favorite editor, look at the following file:

```
cat /opt/lab/support/groupsync.yaml
```

Without going into too much detail (you can look at the documentation), the `groupsync` config file does the following:

- searches LDAP using the specified bind user and password
- queries for any LDAP groups whose name begins with `ose-`
- creates OpenShift groups with a name from the `cn` of the LDAP group
- finds the members of the LDAP group and then puts them into the created OpenShift group
- uses the `dn` and `uid` as the UID and name attributes, respectively, in OpenShift

Now that you understand how the `groupsync` is configured, first make sure you are logged in as the `root` user. Then, make sure you are logged in as a cluster administrator:

```
oc login -u system:admin
```

And then execute the `groupsync`:

```
oc adm groups sync --sync-config=/opt/lab/support/groupsync.yaml -
```

You will see output like the following:

```
group/ose-user
group/ose-normal-dev
group/ose-fancy-dev
group/ose-teamed-app
```

What you are seeing is the **Group** objects that have been created by the `groupsync` command. If you are curious about the `--confirm` flag, check the output of the help with `oc adm groups sync -h`.

If you want to see the **Groups** that were created, execute the following:

```
oc get groups
```

You will see output like the following:

```
NAME             USERS
ose-fancy-dev    fancyuser1, fancyuser2
ose-normal-dev   normaluser1, teamuser1, teamuser2
ose-teamed-app   teamuser1, teamuser2
ose-user         normaluser1, fancyuser1, fancyuser2, teamuser1, t
```

Take a look at a specific group in YAML:

```
oc get group ose-fancy-dev -o yaml
```

The YAML looks like:

```
apiVersion: v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2017-08-29T14:29:33Z
    openshift.io/ldap.uid: cn=ose-fancy-dev,cn=groups,cn=a
    openshift.io/ldap.url: idm.internal.aws.testdrive.open
  creationTimestamp: 2017-08-29T14:29:33Z
  labels:
    openshift.io/ldap.host: idm.internal.aws.testdrive.ope
  name: ose-fancy-dev
  resourceVersion: "5242"
  selfLink: /oapi/v1/groups/ose-fancy-dev
  uid: 7a353ec7-8cc6-11e7-b355-0ee4d6c98466
```

```
    users:
    - fancyuser1
    - fancyuser2
```

OpenShift has automatically associated some LDAP metadata with the **Group**, and has listed the users who are in the group.

What happens if you list the **Users**?

```
oc get user
```

If you logged into the web console as `fancyuser1` previously, you will see:

```
NAME         UID                                    FULL NAME
fancyuser1   d0f11476-a9c3-11e7-a5b2-029466ad0702   OpenShift User
```

Or, if you did not login with the UI before, you will get:

```
No resources found.
```

Why would there be no **Users** found? They are clearly listed in the **Group** definition.

Users are not actually created until the first time they try to log in. What you are seeing in the **Group** definition is simply a placeholder telling OpenShift that, if it encounters a **User** with that specific ID, that it should be associated with the **Group**.

Change Group Policy

In your environment, there are a special group of super developers called *ose-fancy-dev* who should have special `cluster-reader` privileges. This is a role that allows a user to view administrative-level information about the cluster. For example, they can see the list of all **Projects** in the cluster.

Change the policy for the `ose-fancy-dev` **Group**:

```
oc adm policy add-cluster-role-to-group cluster-reader ose-fancy-d
```

If you are interested in the different roles that come with OpenShift, you can learn more about them in the authorization policies documentation.

Examine `cluster-reader` policy

Go ahead and login as a regular user:

```
oc login -u normaluser1 -p openshift
```

Then, try to list **Projects**:

```
oc get projects
```

You will see:

```
No resources found.
```

Now, login as a member of `ose-fancy-dev`:

```
oc login -u fancyuser1 -p openshift
```

And then perform the same `oc get projects` and you will now see the list of all of the projects in the cluster:

```
NAME              DISPLAY NAME   STATUS
default                          Active
kube-system                      Active
logging                          Active
management-infra                 Active
openshift                        Active
openshift-infra                  Active
```

You should now be starting to understand how RBAC in OpenShift Container Platform can work.

Create Projects for Collaboration

Make sure you login as the cluster administrator:

```
oc login -u system:admin
```

Then, create several **Projects** for people to collaborate:

```
oc adm new-project app-dev --display-name="Application Development
oc adm new-project app-test --display-name="Application Testing"
oc adm new-project app-prod --display-name="Application Production
```

You have now created several **Projects** that represent a typical Software Development Lifecycle setup. Next, you will configure **Groups** to grant collaborative access to these projects.

Creating projects with `oc adm new-project` does **not** use the project request process or the project request template. These projects will not have quotas or limitranges applied by default. A cluster administrator can "impersonate" other users, so there are several options if you wanted these projects to get quotas/limit ranges:

1. use `--as` to specify impersonating a regular user with `oc new-project`
2. use `oc process` and provide values for the project request template, piping into create (eg:

`oc process … | oc create -f -`). This will create all of the objects in the project request template, which would include the quota and limit range.

3. manually create/define the quota and limit ranges after creating the projects.

For these exercises it is not important to have quotas or limit ranges on these projects.

Map Groups to Projects

As you saw earlier, there are several roles within OpenShift that are preconfigured. When it comes to **Projects**, you similarly can grant view, edit, or administrative access. Let's give our `ose-teamed-app` users access to edit the development and testing projects:

```
oc adm policy add-role-to-group edit ose-teamed-app -n app-dev
oc adm policy add-role-to-group edit ose-teamed-app -n app-test
```

And then give them access to view production:

```
oc adm policy add-role-to-group view ose-teamed-app -n app-prod
```

Now, give the `ose-fancy-dev` group edit access to the production project:

```
oc adm policy add-role-to-group edit ose-fancy-dev -n app-prod
```

Examine Group Access

Log in as `normaluser1` and see what **Projects** you can see:

```
oc login -u normaluser1 -p openshift
oc get projects
```

Then, try `teamuser1` from the `ose-teamed-app` group:

```
oc login -u teamuser1 -p openshift
oc get projects
```

You did not grant the team users edit access to the production project. Go ahead and try to create something in the production project as `teamuser1`:

```
oc project app-prod
oc new-app docker.io/siamaksade/mapit
```

You will see that it will not work:

```
--> Found Docker image 338a303 (3 weeks old) from docker.io for "d
```

```
      * An image stream will be created as "mapit:latest" that will
      * This image will be deployed in deployment config "mapit"
      * Ports 8080/tcp, 8778/tcp, 9779/tcp will be load balanced by
        * Other containers can access this service through the hostna

--> Creating resources ...
    error: User "teamuser1" cannot create imagestreams in project
    error: User "teamuser1" cannot create deploymentconfigs in proj
    error: User "teamuser1" cannot create services in project "app
--> Failed
```

This failure is exactly what we wanted to see.

Go to previous module | Go to next module