Lab: Implementing GitOps with Jenkins

In this lab, you will implement a GitOps workflow to automate the Samples cluster operator configuration and remove some of the standard templates from your cluster.

**Outcomes**

You should be able to:

* Deploy a Jenkins instance with cluster administration rights on OpenShift using an Ansible Playbook.
* Complete and deploy a Jenkins pipeline that applies settings for the Samples cluster operator from YAML files in a Git repository.
* Use YAML resource files to exclude standard templates for Ruby on Rails applications from your cluster and configure the *Samples cluster operator* so these templates are not recreated.
* Generate a Jenkins API token.

The Samples cluster operator manages standard templates and image streams in a cluster. It usually acts only during installation and upgrades of OpenShift, and it also watches the openshift namespace to prevent accidental deletion of these standard resources.

**NOTE**

To make this review lab shorter, you will not deploy a Drift pipeline; therefore you will implement an incomplete GitOps workflow.

To perform this exercise, ensure you have access to:

* A running OpenShift cluster.
* The Jenkins and Node.js agent container images from Red Hat.
* Initial files for the Samples operator configuration resources, the Apply pipeline, and the Ansible Playbook.

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and downloads sources of the sample application.

**[student@workstation ~]$ lab gitops-review start**

**Procedure 4.4. Instructions**

1. Complete an Ansible Playbook that deploys Jenkins on the gitops-review project.

An incomplete playbook is available in the deploy-jenkins.yaml file in the /home/student/DO380/labs/gitops-review folder. That playbook uses YAML resource files in the /home/student/DO380/labs/gitops-review/resources folder.

Complete the playbook and its resource files to use the correct OpenShift template for a Jenkins instance with persistent storage, and also to grant cluster administration rights to the Jenkins service account.

If you are unsure about the indentation in your YAML files, you can review the solution files in the /home/student/DO380/solutions/gitops-review folder.

**NOTE**

The playbook uses the oc command to process a template because that functionality is not available from the k8s Ansible module.

* 1. Inspect the incomplete playbook.

Open the deploy-jenkins.yaml file in the ~/DO380/labs/gitops-review folder with a text editor and verify that it contains five Ansible tasks.

You will change only the first, second, and fourth tasks, which are highlighted in the following listing.

- name: Deploy Jenkins for GitOps

hosts: localhost

gather\_facts: no

tasks:

**- name: Create project for Jenkins**

*...output omitted...*

**- name: Proccess the Jenkins standard template**

*...output omitted...*

- name: Deploy Jenkins using resources from the standard template

*...output omitted...*

**- name: Grant cluster-administrator privileges to the Jenkins service account**

*...output omitted...*

- name: Wait until Jenkins is available

*...output omitted...*

Leave the playbook file open in the text editor for the following steps.

* 1. Complete the resource file for the first task, Create project for Jenkins, to create the gitops-review project.

This task uses the project.yaml file to define a project request resource. Make no changes to the task.

*...output omitted...*

- name: Create project for Jenkins

k8s:

state: present

src: "{{ playbook\_dir + **'/resources/project.yaml'** }}"

*...output omitted...*

Open a second terminal, and then open the project.yaml file in the ~/DO380/labs/gitops-review/resources folder with a text editor to specify gitops-review as the name of the project in the project request resource.

apiVersion: project.openshift.io/v1

kind: ProjectRequest

metadata:

name: **gitops-review**

description: Project to host a Jenkins instance with cluster admin rights

Save your changes and leave the second terminal open for the following steps.

* 1. Complete the second task, Process the Jenkins standard template, to use the `jenkins-persistent`standard template.

Switch to your first terminal and change the oc process command inside the task as follows:

*...output omitted...*

- name: Proccess the Jenkins standard template

shell: "oc process **jenkins-persistent** -n openshift -o yaml"

register: template\_resources

*...output omitted...*

* 1. Complete the fourth task, Grant cluster administrator privileges to Jenkins, to assign the cluster-admin cluster role to the jenkins service account inside the gitops-review project.

That task uses the role-binding.yaml file to define a project request resource. Make no changes to the task.

...output omitted...

- name: Grant cluster administrator privileges to the Jenkins service account

k8s:

state: present

src: "{{ playbook\_dir + '/resources/role-binding.yaml' }}"

...output omitted...

Switch to your second terminal and open the role-binding.yaml file in the ~/DO380/labs/gitops-review/resources folder with a text editor. Specify the name of the cluster role and of the service account in the cluster role binding resource as follows:

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: jenkins-cluster-admin

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: **cluster-admin**

subjects:

- kind: ServiceAccount

name: **jenkins**

namespace: gitops-review

* 1. Switch to your first terminal, save changes to your playbook, and close your text editor.

You can compare your edits with the solution files in the /home/student/DO380/solutions/gitops-review folder.

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Run the playbook and verify that Jenkins is ready and running, and that its service account has cluster administator privileges.

Also, log in to the Jenkins web UI as the admin user to verify that your Jenkins instance is fully operational.

* 1. Log in to OpenShift as a cluster administrator and navigate to the /home/student/DO380/labs/gitops-review folder.
  2. **[student@workstation ~]$ oc login -u admin -p redhat \**
  3. **> https://api.ocp4.example.com:6443**
  4. Login successful.
  5. *...output omitted...*

**[student@workstation ~]$ cd ~/DO380/labs/gitops-review**

* 1. Run the deploy-jenkins.yaml Ansible Playbook.
  2. **[student@workstation gitops-review]$ ansible-playbook deploy-jenkins.yaml**
  3. PLAY [Deploy Jenkins for GitOps]
  4. *...output omitted...*
  5. PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  6. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=5 changed=4 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

**NOTE**

The task [Wait until Jenkins is available] will take 1-2 minutes to complete.

**IMPORTANT**

The lab script deploys a global Ansible configuration file to /etc/ansible/ansible.cfg. This global Ansible configuration file sets become: true. The become: true option makes playbook tasks execute as the root user. The lab script runs as the root user and logs in to OpenShift as a user with administrator privileges.

The previous ansible-playbook command interacts with OpenShift as a user with administrator privileges.

* 1. Enter the gitops-review project and verify that the Jenkins master pod is ready and running.

Also confirm that Jenkins is fully initialized by looking for the message "Jenkins is fully up and running" in the container logs.

**[student@workstation gitops-review]$ oc project gitops-review**

Now using project "gitops-review" on server "https://api.ocp4.example.com:6443".

**[student@workstation gitops-review]$ oc get pod**

NAME READY STATUS RESTARTS AGE

jenkins-1-5f46k **1/1** **Running** 0 3m17s

jenkins-1-deploy 0/1 Completed 0 3m20s

**[student@workstation gitops-review]$ oc logs jenkins-1-*5f46k* | grep \**

**> 'up and running'**

2020-06-13 15:02:47.679+0000 [id=19] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running

* 1. Verify that the Jenkins service account has cluster administrator privileges.

One way to check these privileges is to verify if the service account is among the subjects allowed to list nodes.

**[student@workstation gitops-review]$ oc policy who-can get node | grep jenkins**

system:serviceaccount:gitops-review:jenkins

* 1. Find the host name of your Jenkins instance. Do not close your terminal, you will come back to it a few times during this exercise.
  2. **[student@workstation gitops-review]$ oc get route**
  3. NAME HOST/PORT ...

jenkins **jenkins-gitops-review.apps.ocp4.example.com** ...

* 1. Open a web browser on the workstation machine, and then enter the Jenkins web UI using the URL obtained in the previous step.

If prompted, accept the TLS certificate for the Jenkins web UI, and then click **Log in with OpenShift**.

Also accept the TLS certificate of the OpenShift OAuth server, if prompted, and then click **htpasswd\_provider** to enter the OpenShift log in credentials page.

Log in as the admin user with the redhat password, and then click **Allow selected permissions**. OpenShift redirects your browser to the Jenkins **Dashboard** page.

Do not close your web browser, you will come back to the Jenkins web UI later in this exercise.

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Create a public GitHub repository named gitops-review for your GitOps workflow, and then clone that repository to your home folder.

From the /home/student/DO380/labs/gitops-review/ folder, copy the Jenkinsfile-apply file and the samples and delete folders to your repository, and then commit and push these files to GitHub.

**NOTE**

The stub resource files must be present in the delete folder because the oc apply command and Kustomize cannot delete Kubernetes resources. The incomplete Jenkinsfile already contains an oc delete command to handle these.

* 1. Open a web browser and access [https://github.com](https://github.com/).

If you do not have a GitHub account, then click **Sign Up** and follow the instructions to create a personal account.

If you do have a GitHub account, then click **Sign In** and follow the instructions to log in using your personal account.

* 1. Create a new empty repository named gitops-review.

On the **Repositories** page, click **New** to enter the **Create a new repository** page. Type gitops-review in the **Repository name** field. Leave all other fields at their default values and click **Create repository**.

On the **Quick setup** page, click the clipboard icon to copy the HTTPS URL of your repository.

* 1. Switch to your terminal and clone the repository into your home folder.
  2. **[student@workstation gitops-review]$ cd ~**
  3. **[student@workstation ~]$ git clone \**
  4. **> https://github.com/*youraccount*/gitops-review.git**
  5. Cloning into 'gitops-review'...

warning: You appear to have cloned an empty repository.

* 1. Copy the incomplete Jenkinsfile, the configuration folder of the Samples cluster operator, and the folder with stub files from the ~/DO380/labs/gitops-review folder into your local clone of the git repository.
  2. **[student@workstation ~]$ cp ~/DO380/labs/gitops-review/Jenkinsfile-apply \**
  3. **> ~/gitops-review**
  4. **[student@workstation ~]$ cp -r ~/DO380/labs/gitops-review/samples ~/gitops-review**
  5. **[student@workstation ~]$ cp -r ~/DO380/labs/gitops-review/delete ~/gitops-review**
  6. **[student@workstation ~]$ ls ~/gitops-review**

delete Jenkinsfile-apply samples

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Create a Kustomize configuration file in the samples folder of your Git repository that generates configuration resources for the Samples cluster operator.

You already copied the samples.yaml file with the configuration for the Samples cluster operator to that folder. Do not make any change to that file and use it as input to Kustomize.

**NOTE**

If you want to understand the contents of the samples.yaml file, use the oc explain command, for example:

**[user@demo ~]$ oc explain config.spec \**

**> --api-version samples.operator.openshift.io/v1**

**NOTE**

Using Kustomize is not required by the scenario of this review lab. It uses Kustomize as a recommended practice for generic GitOps workflows.

* 1. Enter your Git repository clone folder and inspect its samples folder.

It contains a single resource file named samples.yaml.

**[student@workstation ~]$ cd ~/gitops-review**

**[student@workstation gitops-review]$ ls samples**

samples.yaml

* 1. Inspect the samples/samples.yaml resource file.

The skippedTemplates attribute lists all standard templates related to Ruby on Rails applications and makes the Samples cluster operator delete them from the openshift namespace. Do not make any change to this file.

**[student@workstation gitops-review]$ cat samples/samples.yaml**

apiVersion: samples.operator.openshift.io/v1

kind: Config

metadata:

name: cluster

spec:

architectures:

- x86\_64

managementState: Managed

**skippedTemplates:**

**- rails-pgsql-persistent**

**- rails-postgresql-example**

* 1. Create the Kustomize configuration file that lists all resource files in the samples folder.

Use a text editor to create the kustomization.yaml file in the samples folder and add the following lines.

**resources:**

**- samples.yaml**

* 1. Validate your Kustomize configurations.
  2. **[student@workstation gitops-review]$ oc kustomize samples**
  3. apiVersion: samples.operator.openshift.io/v1
  4. kind: Config
  5. *...output omitted...*
  6. skippedTemplates:
  7. - rails-pgsql-persistent

- rails-postgresql-example

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Use your YAML files to change the configuration of the Samples cluster operator and also delete the standard templates for Ruby on Rails applications.

After you verify that your YAML files work as expected, commit and push them to GitHub.

Then, restore the default configuration for the Samples operator using the Kustomize folder at ~/DO380/labs/gitops-review/default. Do not make any change to this folder. Also, verify that the Samples cluster operator restored the templates you deleted.

* 1. List all templates in your cluster related to Ruby on Rails.
  2. **[student@workstation gitops-review]$ oc get template -n openshift | grep rails**
  3. rails-pgsql-persistent An example Rails application ...

rails-postgresql-example An example Rails application ...

* 1. Apply the Samples cluster operator configuration from the samples folder.
  2. **[student@workstation gitops-review]$ oc apply -k samples**

config.samples.operator.openshift.io/cluster configured

* 1. Delete the templates for Ruby on Rails applications using the stub files from the delete folder.
  2. **[student@workstation gitops-review]$ oc delete -f delete**
  3. template.template.openshift.io "rails-pgsql-persistent" deleted

template.template.openshift.io "rails-postgresql-example" deleted

* 1. Once again, list all templates in your cluster related to Ruby on Rails. There should be none.
  2. **[student@workstation gitops-review]$ oc get template -n openshift | grep rails**

**[student@workstation gitops-review]$**

* 1. Commit your samples and delete folders and then push to GitHub.
  2. **[student@workstation gitops-review]$ git add samples**
  3. **[student@workstation gitops-review]$ git add delete**
  4. **[student@workstation gitops-review]$ git commit -m 'Exclude templates for Ruby'**
  5. *...output omitted...*
  6. create mode 100644 delete/templates.yaml
  7. create mode 100644 samples/kustomization.yaml
  8. create mode 100644 samples/samples.yaml
  9. **[student@workstation gitops-deploy]$ git push**
  10. *...output omitted...*

\* [new branch] master -> master

* 1. Restore the default configuration for the Samples cluster operator.
  2. **[student@workstation gitops-review]$ oc apply -k \**
  3. **> ~/DO380/labs/gitops-review/default**

config.samples.operator.openshift.io/cluster configured

* 1. Make sure that the templates related to Ruby on Rails are back to your cluster before moving to the next step.
  2. **[student@workstation gitops-review]$ oc get template -n openshift | grep rails**
  3. rails-pgsql-persistent An example Rails application ...

rails-postgresql-example An example Rails application ...

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Complete the Jenkinsfile for the Apply pipeline.

You already copied an incomplete Jenkinsfile named Jenkinsfile-apply.yaml to your Git repository.

Complete the Jenkinsfile to apply YAML resource files from the samples folder of your Git repository. Also, make sure that your pipeline only applies and tests changes for builds running from the master branch.

The pipeline also deletes resources using stub files in the delete folder of your Git repository, and verifies that no template for Ruby on Rails applications exists in the openshift namespace. Do not make changes to the steps that perform these actions.

If you are unsure about your changes to the Jenkinsfile, you can review the solution files in the /home/student/DO380/solutions/gitops-review folder.

* 1. Inspect the incomplete Jenkinsfile.

Open the Jenkinsfile-apply file in the /home/student/gitops-review folder with a text editor and verify that its pipeline contains three stages. You will change only the Apply and Verify stages, which are highlighted in the following listing.

The pipeline also contains a few directives. You will only change the triggers directive, as highlighted in the following listing. The following steps describe each change in detail.

pipeline {

**triggers {**

*...output omitted...*

options {

*...output omitted...*

agent {

*...output omitted...*

stages {

stage ('Validate configuration resources') {

*...output omitted...*

**stage ('Apply resources') {**

*...output omitted...*

**stage ('Verify deleted templates') {**

*...output omitted...*

}

}

* 1. Configure your pipeline to poll its Git repository every five minutes.

Change your triggers directive as follows.

pipeline {

triggers {

**pollSCM ('H/5 \* \* \* \*')**

}

options {

disableConcurrentBuilds()

}

agent {

node {

label 'nodejs'

}

}

stages {

stage ('Validate configuration resources') {

*...output omitted...*

* 1. Complete your Apply stage to run from the master branch only, and to apply the configurations from the samples folder using Kustomize.
  2. *...output omitted...*
  3. stage ('Validate configuration resources') {
  4. *...output omitted...*
  5. stage ('Apply resources') {
  6. when {
  7. **branch 'master'**
  8. }
  9. steps {
  10. **sh 'oc apply -k samples'**
  11. // ignore delete errors caused by templates deleted by previous runs
  12. sh 'oc delete -f delete --wait || true'
  13. }
  14. }
  15. stage ('Verify deleted templates') {

*...output omitted...*

* 1. Complete your Verify stage to only run from the master branch.
  2. *...output omitted...*
  3. stage ('Apply resources') {
  4. *...output omitted...*
  5. stage ('Verify deleted templates') {
  6. when {
  7. **branch 'master'**
  8. }
  9. steps {
  10. sh '! oc get templates -n openshift | grep rails'
  11. }
  12. }
  13. }

}

* 1. Commit your Jenkinsfile and then push to GitHub.
  2. **[student@workstation gitops-review]$ git add Jenkinsfile-apply**
  3. **[student@workstation gitops-review]$ git commit -m 'Completed Apply pipeline'**
  4. *...output omitted...*
  5. create mode 100644 samples/Jenkinsfile-apply
  6. *...output omitted...*
  7. **[student@workstation gitops-deploy]$ git push**
  8. *...output omitted...*

\* [new branch] master -> master

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Create a Jenkins multibranch pipeline project named apply for your Apply pipeline. Configure your Jenkins project to take the Jenkinsfile from your GitHub repository.

After Jenkins successfully runs a build from your project, verify that the openshift namespace contains no template related to Ruby on Rails applications.

* 1. Create a new multibranch pipeline project for the Apply pipeline.

Click **New Item** on the Jenkins **Dashboard** page, and then type apply as the item name. Select **Multibranch Pipeline**, and then click **Ok**.

Under the **Branch Sources** heading, select **Add Source** → **Git** and type the HTTPS URL of your GitHub repository, https://github.com/*youraccount*/gitops-review.git, in the **Project Repository** field. Because your Git repository is public, you do not provide any credentials.

Under the **Build Configuration** heading, type Jenkinsfile-apply in the **Script Path** field.

Do not set any other fields and click **Save**.

* 1. Verify that Jenkins finds the master branch.

Jenkins automatically starts a **Scan Multibranch Pipeline** job and displays its logs. Wait until it finishes and verifies that the logs of the job indicate that Jenkins found a master branch.

* 1. Wait until Jenkins builds your master branch successfully.

Click either **Up** or **Jenkins** to go to the Jenkins **Dashboard** page. Click **apply** to enter the Apply project, then click **master** to list builds of the **master** branch of your Git repository.

It might take a few moments until a build starts and its stages are displayed. After the Jenkins build starts, wait until it finishes successfully.

* 1. Verify that your cluster now contains no standard templates related to Ruby on Rails.

**[student@workstation gitops-review]$ oc get template -n openshift | grep rails**

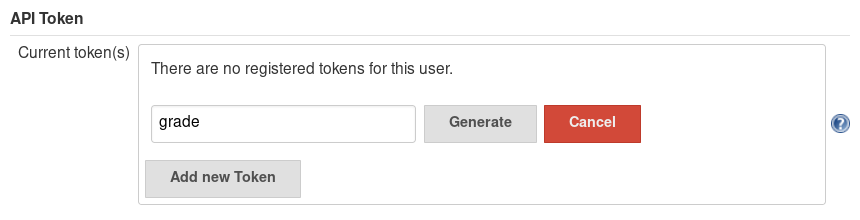
[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

1. Generate a Jenkins API token.

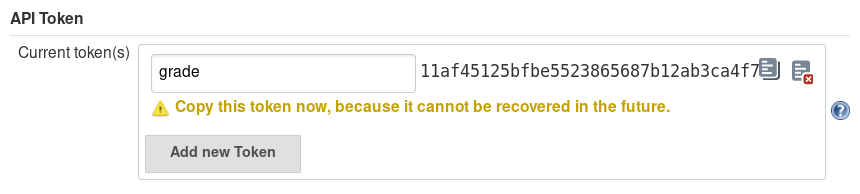
Use the Jenkins web UI to access the configuration for the admin user and save the token to the /home/student/jenkins-api-token.txt file.

The grade command requires that token to invoke the Jenkins API and start a build of your Apply pipeline.

* 1. Switch to the open web browser displaying the Jenkins web UI and click **admin**, found near the **logout** button.
  2. On the **admin** Jenkins user page, click **Configure** and then click **Add new Token**. Type a name for the token, such as 'grade', and click **Generate**.



* 1. Select and copy the long hexadecimal string, which is the value of the token.



* 1. Switch back to your terminal and create the /home/student/jenkins-api-token.txt file using a text editor. Paste the value of the token you got from the previous step into this file.
  2. Change to the /home/student/ directory.

**[student@workstation operators-review]$ cd**

[HIDE SOLUTION](https://rol.redhat.com/rol/app/)

**Evaluation**

As the student user on the workstation machine, use the lab command to grade your work.

Correct any reported failures and rerun the command until successfully.

**[student@workstation ~]$ lab gitops-review grade**

**Finish**

On the workstation machine, use the lab command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

**[student@workstation ~]$ lab gitops-review finish**

The lab command deletes the OpenShift project you created and restores default cluster settings. It also deletes the clone of your gitops-review Git repository from your home folder and the Jenkins API token file.

If you wish, you can also delete the gitops-review repository from your GitHub account.

This concludes the section.

s