Guided Exercise: Deploying Jenkins on OpenShift

In this exercise you will deploy and validate a simple Continuous Integration/Continuous Deployment (CI/CD) pipeline for a sample application.

**Outcomes**

You should be able to:

* Deploy a Jenkins master on OpenShift.
* Create a CI/CD pipeline project using a Jenkinsfile in GitHub from the Jenkins web UI.
* Start and monitor a pipeline build from the Jenkins web UI and the OpenShift CLI.
* Generate a Jenkins API token.

To perform this exercise, ensure you have access to:

* A running OpenShift cluster.
* The Jenkins master and Node.js agent container images from Red Hat.
* A personal free account at GitHub. If you need to register with GitHub, see the instructions in the [the section called “Creating a GitHub Account”](https://rol.redhat.com/rol/app/courses/do380-4.5/pages/apb) section of Appendix B.
* The source code for the "hello, world" sample application.
* The Jenkinsfile to build and deploy the sample application.

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-deploy start**

**Procedure 4.1. Instructions**

1. Create a GitHub project to host the sample application and also the Jenkinsfile.
   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. Refer to the instructions in the [the section called “Creating a GitHub Account”](https://rol.redhat.com/rol/app/apb.html#appendix-git) section of Appendix B for details about creating a GitHub 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-deploy.

On the **Repositories** page, click **New** to enter the **Create a new repository** page. Type gitops-deploy 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.

Do not close your web browser, you will come back to GitHub a few times during this exercise.

* 1. Open a terminal on the workstation machine, and then clone the repository into your home folder.
  2. **[student@workstation ~]$ git clone \**
  3. **> https://github.com/*youraccount*/gitops-deploy.git**
  4. Cloning into 'gitops-deploy'...

warning: You appear to have cloned an empty repository.

* 1. Copy the sample application sources and the Jenkinsfile from the ~/DO380/labs/gitops-deploy/hello folder into your local clone of the git repository.
  2. **[student@workstation ~]$ cp ~/DO380/labs/gitops-deploy/hello/\* ~/gitops-deploy**
  3. **[student@workstation ~]$ cd ~/gitops-deploy**
  4. **[student@workstation gitops-deploy]$ ls**

app.js Jenkinsfile package.json

1. Review the Jenkinsfile for a sample CI/CD pipeline.
   1. Inspect the Jenkinsfile and review its stages.

You do not need to understand all steps inside the Jenkinsfile. These stages work as-is and are ready to build and deploy the sample application.

The pipeline uses the OpenShift Client plug-in DSL to create a new OpenShift project that generates its name from the branch name and build number. Then, the pipeline builds and deploys the sample application from its source code, emulating the steps a developer would perform using the oc new-app command.

Observe that there are multiple echo statements inside each step of each stage. These statements can help you to relate each step to console output from the Jenkins Master.

**[student@workstation gitops-deploy]$ less Jenkinsfile**

pipeline {

environment {

*...output omitted...*

stages {

stage('create') {

*...output omitted...*

stage('build') {

*...output omitted...*

stage('deploy') {

*...output omitted...*

stage('test') {

*...output omitted...*

}

post {

*...output omitted...*

}

}

* 1. Review the test stage.

It waits for an interactive prompt before proceeding, giving you time to inspect OpenShift resources in the middle of a Jenkins build, and then tests the application using the curl command.

**[student@workstation gitops-deploy]$ less Jenkinsfile**

*...output omitted...*

stage('test') {

input {

message 'About to test the application'

ok 'Ok'

}

steps {

echo "Check that '${env.APP}.${env.DOMAIN}' returns HTTP 200"

sh "curl -s --fail ${env.APP}.${env.DOMAIN}"

}

}

*...output omitted...*

* 1. Commit the sources and the Jenkinsfile, and then push to GitHub.
  2. **[student@workstation gitops-deploy]$ git add \***
  3. **[student@workstation gitops-deploy]$ git commit -m 'sample app and pipeline'**
  4. *...output omitted...*
  5. create mode 100644 Jenkinsfile
  6. *...output omitted...*
  7. **[student@workstation gitops-deploy]$ git push**
  8. *...output omitted...*

\* [new branch] master -> master

* 1. Switch to your web browser and refresh the GitHub page that shows your repository. Verify that you see your Jenkinsfile and sample application files there.

1. As a developer, create a project in OpenShift and deploy Jenkins using the persistent template.
   1. Switch to your terminal and log in on OpenShift as the developer user.
   2. **[student@workstation gitops-deploy]$ oc login -u developer -p developer \**
   3. **> https://api.ocp4.example.com:6443**
   4. Login successful.

*...output omitted...*

* 1. Create the gitops-deploy project.
  2. **[student@workstation gitops-deploy]$ oc new-project gitops-deploy**
  3. Now using project "gitops-deploy" on server "https://api.ocp4.example.com:6443".

*...output omitted...*

* 1. List all standard Jenkins templates provided with OpenShift.
  2. **[student@workstation gitops-deploy]$ oc get template -n openshift | grep jenkins**
  3. jenkins-ephemeral Jenkins service, without persistent storage....
  4. jenkins-ephemeral-monitored Jenkins service, without persistent storage....
  5. jenkins-persistent Jenkins service, with persistent storage....

jenkins-persistent-monitored Jenkins service, with persistent storage....

* 1. List the parameters of the jenkins-persistent standard template. Note that it provides parameters that allow you to customize such things as storage size and memory requests for larger environments. Note also that all parameters have default values.
  2. **[student@workstation gitops-deploy]$ oc process jenkins-persistent --parameters \**
  3. **> -n openshift**
  4. NAME DESCRIPTION ... VALUE
  5. *...output omitted...*
  6. MEMORY\_LIMIT Maximum amount of memory the container can use. ... 1Gi
  7. VOLUME\_CAPACITY Volume space available for data, e.g. 512Mi, 2Gi. ... 1Gi

*...output omitted...*

* 1. Deploy Jenkins using the jenkins-persistent template. Accept the default values for all of the template parameters.
  2. **[student@workstation gitops-deploy]$ oc new-app --template jenkins-persistent**
  3. *...output omitted...*
  4. --> Creating resources ...
  5. route.route.openshift.io "jenkins" created
  6. persistentvolumeclaim "jenkins" created
  7. deploymentconfig.apps.openshift.io "jenkins" created
  8. serviceaccount "jenkins" created
  9. rolebinding.authorization.openshift.io "jenkins\_edit" created
  10. service "jenkins-jnlp" created
  11. service "jenkins" created
  12. --> Success

*...output omitted...*

1. As a cluster administrator, grant to the Jenkins service account permission to create OpenShift projects.
   1. Log in to OpenShift as the admin user.
   2. **[student@workstation gitops-deploy]$ oc login -u admin -p redhat**
   3. Login successful.

*...output omitted...*

* 1. Grant the Jenkins service account permission to create new projects.
  2. **[student@workstation gitops-deploy]$ oc adm policy add-cluster-role-to-user \**
  3. **> self-provisioner -z jenkins -n gitops-deploy**

clusterrole.rbac.authorization.k8s.io/self-provisioner added: "jenkins"

**NOTE**

The oc adm policy command creates a cluster role binding resource, which is not namespaced, and references a service account, which is namespaced. Thus, the -n option to specify the project of the service account is required.

1. As a developer, use the Jenkins web UI to create a Jenkins project that takes its Jenkinsfile from a Git repository.
   1. Log in on OpenShift as the developer user and enter the gitops-deploy project in OpenShift.
   2. **[student@workstation gitops-deploy]$ oc login -u developer -p developer**
   3. Login successful.
   4. *...output omitted...*
   5. **[student@workstation gitops-deploy]$ oc project gitops-deploy**
   6. Already on project "gitops-deploy" on server "https://api.ocp4.example.com:6443".

*...output omitted...*

* 1. Wait until the Jenkins master pod is ready and running. It takes time for the pod to make its single container ready because Jenkins needs time to initialize.

To confirm that Jenkins is fully initialized, look for the message "Jenkins is fully up and running" in the container logs.

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

NAME READY STATUS RESTARTS AGE

jenkins-1-deploy 0/1 Completed 0 2m57s

jenkins-1-**xvwwr** **1/1** **Running** 0 2m54s

**[student@workstation gitops-deploy]$ oc logs jenkins-1-*xvwwr* | grep \**

**> 'up and running'**

2020-06-12 14:00:31.798+0000 [id=19] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running

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

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

* 1. Open a new tab in the 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 developer with the developer password, and then click **Allow selected permissions**. OpenShift redirects your browser to the Jenkins **Dashboard** page.

* 1. Create a new multibranch pipeline project for the sample application.

Click **New Item** on the Jenkins **Dashboard** page, and then type hello as the item name. Select **Multibranch Pipeline** and 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-deploy.git, in the **Project Repository** field. Because your Git repository is public you do not provide any credentials.

Leave all other fields unchanged and click **Save**.

* 1. Verify that Jenkins finds the master branch.

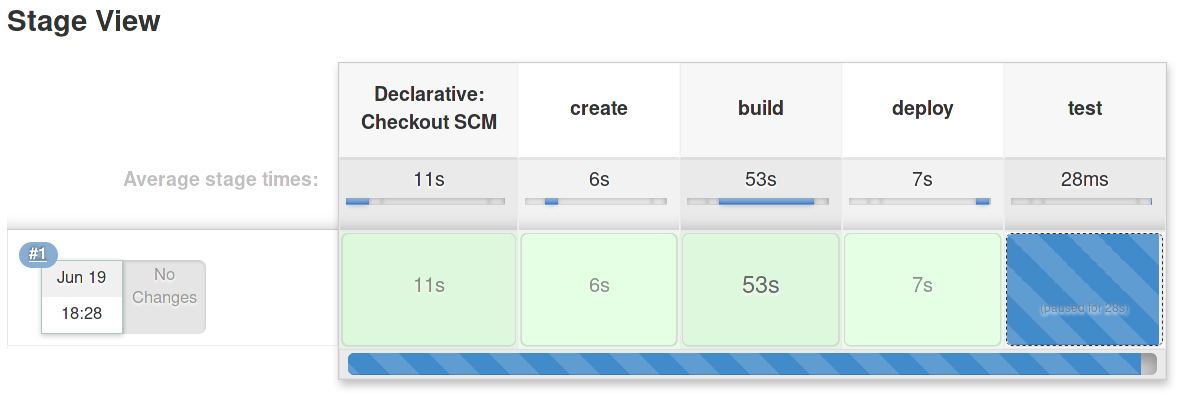
Jenkins automatically starts a **Scan Multibranch Pipeline** job and displays its logs. Wait until it finishes and verifies that the job logs indicate that Jenkins found a master branch. If a master branch is not found, click **Configure** and fix your Jenkins project settings, and then click **Scan Multibranch Pipeline Now**.

* 1. Verify that there is a build running for the master branch.

Click either **Up** or **Jenkins** to go to the Jenkins **Dashboard** page. Click **hello** to enter the new project and click **master** to enter the project’s **Stage View** page.

If you do not see any build, then click **Build Now** to start a build. If you see a build that is in-progress, wait until it reaches the **test** stage.

After the build reaches the test stage, it is paused while waiting for user interaction. Leave the build paused at the test stage for now.



Do not close your web browser. You will come back to the Jenkins web UI a few times during this exercise.

**NOTE**

If you see errors during the build, then hover the mouse over each stage and click **Logs** to find clues about what caused the errors and review your previous steps.

To change your Jenkins project settings, click **hello** and then click **Configure**.

To change your Git repository, commit and push your changes and then click **Build Now** to start a new build.

1. As a developer, monitor a build from the Jenkins web UI and also from the OpenShift CLI.
   1. Click **Jenkins** to enter the Jenkins **Dashboard** page. Notice that there is a representation of an agent running a build under the heading **Build executor status**.

From the information displayed, you can identify the name of the agent, the name of the project, the name of the branch, and the number of the build.

|  |
| --- |
|  |

* 1. Switch to your terminal and verify that there is a pod with the same name as the agent you saw in the previous step.
  2. **[student@workstation gitops-gitops]$ oc get pod**
  3. NAME READY STATUS RESTARTS AGE
  4. jenkins-1-deploy 0/1 Completed 0 18m
  5. jenkins-1-xvwwr 1/1 Running 0 18m

**nodejs-56b7s** 1/1 Running 0 14m

* 1. Verify that there is no controller resource for the Jenkins agent pod. It runs as an unmanaged, stand-alone pod.
  2. **[student@workstation gitops-gitops]$ oc status**
  3. In project gitops-deploy on server https://api.ocp4.example.com:6443
  4. svc/jenkins-jnlp - 172.30.249.140:50000
  5. https://jenkins-gitops-deploy.apps.ocp4.example.com (redirects) (svc/jenkins)
  6. dc/jenkins deploys openshift/jenkins:2
  7. deployment #1 deployed 12 minutes ago - 1 pod
  8. **pod/nodejs-56b7s runs .../jenkins-agent-nodejs:latest**

*...output omitted...*

* 1. Verify that there is a project named hello-*branch*-*build* where the sample application was built and deployed:
  2. **[student@workstation gitops-gitops]$ oc status -n hello-master-1**
  3. In project hello-master-1 on server https://api.ocp4.example.com:6443
  4. http://hello-master-1.apps.ocp4.example.com to pod port 8080-tcp (svc/nodeapp)
  5. deployment/nodeapp deploys istag/nodeapp:latest <-
  6. bc/nodeapp source builds https://github.com/youraccount/gitops-deploy.git#master on openshift/nodejs:12
  7. deployment #2 running for 2 minutes - 1 pod
  8. deployment #1 deployed 3 minutes ago

*...output omitted...*

* 1. Switch to your web browser and inspect the build logs in the Jenkins web UI.

From the Jenkins **Dashboard** page, click **hello** and then click **master** to return to your project’s **Stage View** page.

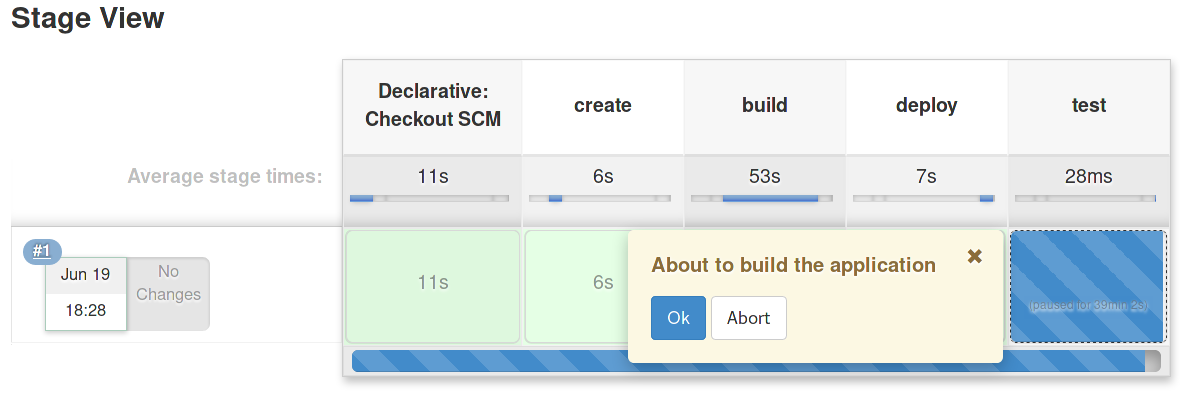
Click the number of the build, **#1**, to enter the **Build** page, and then click **Console Output** to view the build logs. Search for messages from the echo steps in your pipeline.

At the end of the page, just before a spinning disk, there is the message from the input directive of the test stage. Do not click any of the **Ok** or **Abort** links.

|  |
| --- |
|  |

* 1. Return to the **Stage View** page and allow the build to continue.

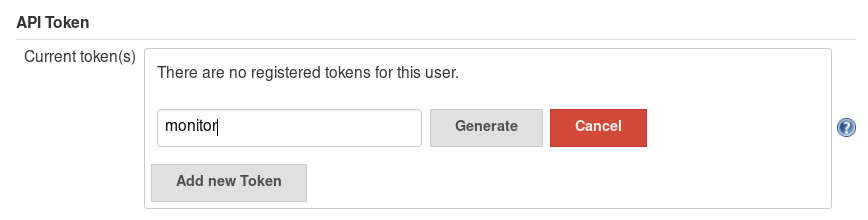
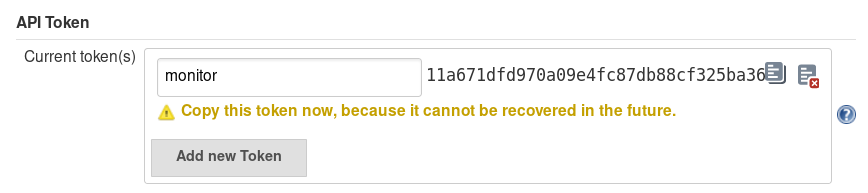
Click either **Back to project** or **master** to return to the **Stage View** page. Hover the mouse over the **test** stage box until you see the input directive window, and then click **Ok**.



The build finishes successfully after a few moments.

* 1. Switch to your terminal and verify that both the agent pod and the hello-main-1 project are no longer displayed. Note that the project might be available for a few moments before OpenShift deletes it.
  2. **[student@workstation gitops-gitops]$ oc get pod**
  3. NAME READY STATUS RESTARTS AGE
  4. jenkins-1-deploy 0/1 Completed 0 26m
  5. jenkins-1-xvwwr 1/1 Running 0 26m
  6. **[student@workstation gitops-gitops]$ oc get project**
  7. NAME DISPLAY NAME STATUS

gitops-deploy Active

1. Monitor your Jenkins build using the Jenkins API.
   1. 
   2. 

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

1. As the developer user, delete the Jenkins project and your GitHub repository.

**NOTE**

**Do not delete** the OpenShift project. You will reuse this project and the Jenkins server you just deployed in a subsequent exercise.

* 1. As the developer user, delete the Jenkins project.

Switch to the Jenkins web UI and click **Jenkins** to return to the Jenkins **Dashboard** page. Click the down arrow near **hello**, and then click **Delete Multibranch Pipeline**.

|  |
| --- |
|  |

Click **Yes** to confirm the operation. It might be necessary to scroll down the page to find the **Yes** button.

* 1. Switch to your web browser and from the GitHub repository page, click **Settings**, and then scroll down until you find the **Danger Zone** header. Click **Delete this repository** and follow the instructions to confirm deleting the repository.
  2. Remove your local clone of the sample application repository.

**[student@workstation ~]$ rm -rf gitops-deploy**

* 1. Remove your Jenkins API token.

**[student@workstation ~]$ rm developer-token.txt**

**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-deploy finish**

This concludes the section.