CI/CD with OpenShift

1. Connect to OpenShift Cluster

Once you are connected to your client VM, you can log in to the OpenShift cluster.

1. Use the **oc login** command to log in to the cluster, making sure to replace the URL with your cluster’s URL:

oc login -u <Your OPENTLC UserID> https://master.<$REGION>.openshift.opentlc.com

|  |  |
| --- | --- |
|  | If you see a warning about an insecure certificate, type **y** to connect insecurely. |

**Sample Output**

Login successful.

You don't have any projects. You can try to create a new project, by running

oc new-project <projectname>

2. Create Jenkins Container

In this section, you create a Jenkins container to serve as your CI/CD server. OpenShift comes with a fully supported Jenkins image that you deploy now.

1. Log in to the web console.
2. On the **Service Catalog**, click the **CI/CD** filter to show the CI/CD templates.
3. Click **Jenkins** to select a Jenkins template that automatically sets up persistent storage.
4. Read the description on the **Information** page and then click **Next**.
5. On the **Configuration** page, enter or select the following values:
   * **Add to Project**: **Create Project**
   * **Project Name**: **xyz-cicd** (replacing **xyz** with your initials to create a unique project name)
   * **Project Display Name**: **My CICD Project**
   * **Memory Limit**: **2Gi**
   * **Disable memory intensive administrative monitors**: **true**
6. Click **Next** to move to the next page.
7. On the **Bindings** page, leave **Do not bind at this time** selected and click **Create**.
8. On the **Results** page, click the blue link to **Continue to project overview**.
9. Open the **jenkins** deployment and wait until Jenkins is fully up and running.
   * You may need to click the Jenkins deployment to see details.
   * The pod turns from a light blue circle to a dark blue circle when Jenkins is ready.
   * Alternatively, click the pod (in the circle) and then observe the pod logs on the **Logs** tab of the pod properties. When you see "INFO: Jenkins is fully up and running" (this will *not* be the last line in the log) you are ready to proceed. At this point the blue circle turned from light blue to dark blue.
10. Once Jenkins is up and running, go back to your project overview.

|  |  |
| --- | --- |
|  | It may take a few minutes for Jenkins to be fully up and running. |

1. Click the route—similar to **https://jenkins-xyz-cicd.apps.na311.openshift.opentlc.com/**—to open Jenkins.
   * You may need to accept the certificate to proceed. You are then presented with the Jenkins login screen.
2. Because Jenkins is OAuth enabled, simply click the **Login with OpenShift** button.
   * It presents the OpenShift login screen where you use your OPENTLC credentials again to log in to Jenkins.
   * When the authentication succeeds, you are asked to allow Jenkins to access OpenShift resources.
3. Leave the default values checked and click **Allow selected permissions**.
   * After a few moments, expect to see the Jenkins home screen.

2.1. Prepare Development Project

Now that Jenkins is ready, you set up a project in OpenShift to hold the application to be built using the command line.

1. On the client VM, create a project (replacing **xyz** with your initials):

oc new-project xyz-tasks

**Sample Output**

Now using project `xyz-tasks` on server "https://master.na39.openshift.opentlc.com:443".

You can add applications to this project with the 'new-app' command. For example, try:

oc new-app centos/ruby-25-centos7~https://github.com/openshift/ruby-ex.git

to build a new example application in Ruby.

1. Create the application that you want to build using Jenkins:

oc new-app jboss-eap71-openshift:1.3~https://github.com/redhat-gpte-devopsautomation/openshift-tasks

**Sample Output**

*--> Found image f02a32b (3 months old) in image stream "openshift/jboss-eap71-openshift" under tag "1.3" for "jboss-eap71-openshift:1.3"*

JBoss EAP 7.1

*-------------*

Platform for building and running JavaEE applications on JBoss EAP 7.1

Tags: builder, javaee, eap, eap7

\* A source build using source code from https://github.com/redhat-gpte-devopsautomation/openshift-tasks will be created

\* The resulting image will be pushed to image stream tag "openshift-tasks:latest"

\* Use 'start-build' to trigger a new build

\* This image will be deployed in deployment config "openshift-tasks"

\* Ports 8080/tcp, 8443/tcp, 8778/tcp will be load balanced by service "openshift-tasks"

\* Other containers can access this service through the hostname "openshift-tasks"

*--> Creating resources ...*

imagestream.image.openshift.io "openshift-tasks" created

buildconfig.build.openshift.io "openshift-tasks" created

deploymentconfig.apps.openshift.io "openshift-tasks" created

service "openshift-tasks" created

*--> Success*

Build scheduled, use 'oc logs -f bc/openshift-tasks' to track its progress.

Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:

'oc expose svc/openshift-tasks'

Run 'oc status' to view your app.

* + This command creates a build configuration, image stream, deployment configuration, and service for your application.

1. Create a route so that you can test the application:

oc expose svc openshift-tasks

**Sample Output**

route.route.openshift.io/openshift-tasks exposed

1. Set the **MAVEN\_MIRROR\_URL** environment variable for the **openshift-tasks** build configuration:

oc set env bc openshift-tasks MAVEN\_MIRROR\_URL=http://nexus.opentlc-shared.svc.cluster.local:8081/repository/maven-all-public

**Sample Output**

buildconfig.build.openshift.io/openshift-tasks updated

* + You specify the Nexus path where the proxy repositories for all Red Hat and JBoss repositories are located (**/repository/maven-all-public**).
  + Because you are using the service directly and not through a router, you must also specify the port at **8081**.

|  |
| --- |
| * + - To speed up the build you set a Maven Mirror. This **MAVEN\_MIRROR\_URL** is an environment variable that is supported by all Red Hat xPaaS builder images and can be used to point to a Maven proxy repository. You already have a shared Nexus repository running as a pod in a common project on the cluster.     - Also note that rather than using the route you are actually pointing to the fully qualified service name. You can always construct the fully qualified service name like this:   <Service Name>.<Project Name>.svc.cluster.local |

1. Turn off all automatic triggers:

oc set triggers dc openshift-tasks --manual

**Sample Output**

deploymentconfig "openshift-tasks" updated

* + Because you are building this application using a Jenkins pipeline, it is Jenkins that must have full control over what happens in this project. By default, the application gets redeployed every time a new image is available. However, if you rebuild the image via Jenkins, you may want to run a few tests before you actually redeploy the application.

1. Give the service account the correct permissions to *edit* objects in this project to allow Jenkins to build and deploy the application.
   * Make sure to replace **xyz-cicd** and **xyz-tasks** with your specific project names.

oc policy add-role-to-user edit system:serviceaccount:xyz-cicd:jenkins -n xyz-tasks

**Sample Output**

role "edit" added: "system:serviceaccount:xyz-cicd:jenkins"

* + Because the Jenkins server is in a different project than the application, the **jenkins** service account used to run the Jenkins pod needs to be given access to the application’s project. By default, this account, does not have access to the project with your application.

|  |  |
| --- | --- |
|  | When granting permissions it is always necessary to specify the fully qualified path to the service account. For a single service account, this is always **system:serviceaccount:<project name>:<service account name>**. |

You are now ready to start building this application using Jenkins.

2.2. Create Simple Pipeline in Jenkins

Now that everything is prepared, you can build the pipeline in Jenkins.

1. Log in to your Jenkins server again.
2. In the navigator on the left, click **New Item**.
3. Type **Tasks** for the item name, select **Pipeline** for the job type, and click **OK**.
   * At the bottom of the screen expect to see a multiline text box to write your pipeline in.
4. Paste the pipeline shown below into the text box.
   * Make sure that all of the **xyz-tasks** project/namespace names point to your actual project name.
   * There are three occurrences overall in this pipeline.
   * node {
   * stage('Build Tasks') {
   * openshift.withCluster() {
   * openshift.withProject("xyz-tasks") {
   * openshift.selector("bc", "openshift-tasks").startBuild("--wait=true")
   * }
   * }
   * }
   * stage('Tag Image') {
   * openshift.withCluster() {
   * openshift.withProject("xyz-tasks") {
   * openshift.tag("openshift-tasks:latest", "openshift-tasks:${BUILD\_NUMBER}")
   * }
   * }
   * }
   * stage('Deploy new image') {
   * openshift.withCluster() {
   * openshift.withProject("xyz-tasks") {
   * openshift.selector("dc", "openshift-tasks").rollout().latest();
   * }
   * }
   * }

}

1. Click **Save** to save this pipeline job.
   * The pipeline job performs the following:
     + Executes a build by calling the OpenShift build configuration.
     + Tags the image to use the Jenkins build number as the latest tag. This ensures that every build creates a unique tag.
     + Deploys the new image by invoking the OpenShift deploy operation on the deployment configuration.
2. On the Jenkins job page, click **Build now**.
   * Expect to see your pipeline progress through the three stages until it finishes successfully.

|  |  |
| --- | --- |
|  | If your pipeline fails, double-check the names of your project in the pipeline. You can also click the small triangle next to your build number under **Build History**. (To see the triangle, hover over the build number.) Select **Console Log** for more details about the build process. |

1. From your job overview, click **Open Blue Ocean** to observe the pipeline.
   * Blue Ocean is Jenkins' new, more modern user interface for observing pipelines.
2. On the Blue Ocean pipeline view, click **Run** to kick off another pipeline run.
   * Expect to see a small box in the bottom right corner indicating that a new pipeline run is started.
3. Click the **Open** link in the notification box to show the details of the pipeline run. If you miss it you may select your build in the list of builds to open the pipeline view.
4. Observe how the build progresses through the stages and how the logs are shown directly under the stage view.
   * When your pipeline is finished, the pipeline view switches from blue to green.
5. Open each stage to see logs of any step in the pipeline.

2.3. Confirm Running Application

In this section, you verify that your application is running.

1. Return to the OpenShift web console and switch to your **xyz-tasks** application project.
2. Click the route to your application (similar to **http://openshift-tasks-xyz-tasks.apps.na311.openshift.opentlc.com**).
   * Expect to see the tasks application that was just deployed using your Jenkins pipeline.

3. Clean Up Environment

In this section, you clean up the lab environment by simply deleting the two projects that you created. You can do this from either the web console or the command line, replacing **xyz** with your initials.

1. Delete the projects using the command line:

oc delete project xyz-tasks xyz-cicd