Table of Contents

* [Demonstrate OpenShift Pipelines Integration Lab](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_demonstrate_openshift_pipelines_integration_lab)
* [1. Prepare for Demonstration](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_prepare_for_demonstration)
  + [1.1. Install Command Line Tools](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_install_command_line_tools)
  + [1.2. Create Projects for Demonstration](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_create_projects_for_demonstration)
  + [1.3. Deploy the CICD Environment](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_deploy_the_cicd_environment)
  + [1.4. Deploy Mock Applications](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_deploy_mock_applications)
  + [1.5. Create Initial Build Config Pipeline](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_create_initial_build_config_pipeline)
  + [1.6. Test the New Pipeline](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_test_the_new_pipeline)
* [2. Demonstrate OpenShift Pipelines Integration](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_demonstrate_openshift_pipelines_integration)
* [3. Additional Demonstrations](https://cloud.scorm.com/content/courses/A9KI96X2QE/OpenShift_3_Foundations/8/04_CICD_and_Pipelines_OpenShift/04_02_OpenShift_Pipelines_Integration_Lab.html" \l "_additional_demonstrations)

## Demonstrate OpenShift Pipelines Integration Lab

Scenario

In the scenario for this demonstration lab, a developer wants to deploy a new CICD pipeline to manage and monitor the application development life cycle.

* In this scenario, the application goes through different stages and moves between them up to the production stage.
* This demonstration requires preparation before the demonstration can start.

## 1. Prepare for Demonstration

In this section, you set up the prerequisites for the demonstration. Please note the following:

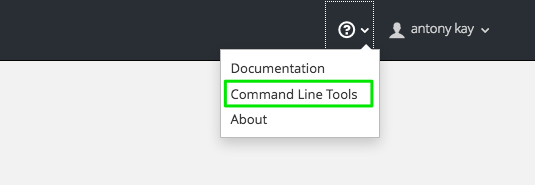
* The instructions are brief and not meant to be done in front of the demo audience.
* At the end of the section, a simple script to create the demo is made available.

### 1.1. Install Command Line Tools

1. Log in to the OPENTLC shared OpenShift web console: [https://master.na1.openshift.opentlc.com](https://master.na1.openshift.opentlc.com/).

|  |  |
| --- | --- |
|  | The link for the North American region is [https://master.na1.openshift.opentlc.com](https://master.na1.openshift.opentlc.com/). |

1. Select Help (small question mark icon) at the top right corner of the screen.
2. Select Command Line Tools and then latest release to access the official Red Hat OpenShift download page:



|  |  |
| --- | --- |
|  | You can also download the appropriate OpenShift Origin client from this link if you do not have a Red Hat Network account: <https://github.com/openshift/origin/releases/tag/v1.3.1>. |

1. Download and unpack the file on your computer.
2. Copy the extracted oc file to /usr/bin or an appropriate location for your operating system.
3. Open a command line shell and test your ability to authenticate to OpenShift:

LocalMachine$ export REGION="na" ; export USERNAME=youropentlc-account.com

LocalMachine$ oc login https://master.${REGION}.openshift.opentlc.com:443 --username ${USERNAME}

Username: shacharb-redhat.com

Password: \*\*\*\*\*\*\*\*\*\* [ This is your OPENTLC Account password ]

Logged into "https://master.${REGION}.openshift.opentlc.com:443" as "shacharb-redhat.com" using existing credentials.

### 1.2. Create Projects for Demonstration

1. Create a few projects for your demonstration:

|  |  |
| --- | --- |
|  | You must select a unique identifier (GUID) so your project names and routes do not conflict with other users in the OpenShift cluster. |

#GUID=youruniquename

GUID=mydemo

oc new-project pipeline-${GUID}-dev --description="Cat of the Day Development Environment" --display-name="Cat Of The Day - Dev"

oc new-project pipeline-${GUID}-test --description="Cat of the Day Testing Environment" --display-name="Cat Of The Day - Test"

oc new-project pipeline-${GUID}-prod --description="Cat of the Day Production Environment" --display-name="Cat Of The Day - Prod"

1. Display your created projects:

oc get projects

Sample Output

oc get projects

NAME DISPLAY NAME STATUS

pipeline-mydemo-dev Cat Of The Day - Dev Active

pipeline-mydemo-prod Cat Of The Day - Prod Active

pipeline-mydemo-test Cat Of The Day - Test Active

1. Switch back to your dev project where most of the work needs to be done:

oc project pipeline-${GUID}-dev

### 1.3. Deploy the CICD Environment

1. Deploy Jenkins to control your builds and deployment pipeline:

oc new-app jenkins-persistent -p ENABLE\_OAUTH=false -p MEMORY\_LIMIT=1.5Gi -n pipeline-${GUID}-dev

Sample Output

--> Deploying template "openshift/jenkins-persistent" to project pipeline-${GUID}-dev

Jenkins (Persistent)

---------

Jenkins service, with persistent storage.

NOTE: You must have persistent volumes available in your cluster to use this template.

A Jenkins service has been created in your project. Log into Jenkins with your OpenShift account. The tutorial at https://github.com/openshift/origin/blob/master/examples/jenkins/README.md contains more information about using this template.

\* With parameters:

\* Jenkins Service Name=jenkins

\* Jenkins JNLP Service Name=jenkins-jnlp

\* Enable OAuth in Jenkins=false

\* Memory Limit=1.5Gi

\* Volume Capacity=1Gi

\* Jenkins ImageStream Namespace=openshift

\* Jenkins ImageStreamTag=jenkins:latest

--> Creating resources ...

route "jenkins" created

persistentvolumeclaim "jenkins" created

deploymentconfig "jenkins" created

serviceaccount "jenkins" created

rolebinding "jenkins\_edit" created

service "jenkins-jnlp" created

service "jenkins" created

--> Success

Run 'oc status' to view your app.

1. The Jenkins login and password is admin:password.
   * This password is set to that default as a result of setting ENABLE\_OAUTH=false earlier
2. Enable the Jenkins service account to manage resources in the pipeline-${GUID}-test and pipeline-${GUID}-prod projects:

oc policy add-role-to-user edit system:serviceaccount:pipeline-${GUID}-dev:jenkins -n pipeline-${GUID}-test

oc policy add-role-to-user edit system:serviceaccount:pipeline-${GUID}-dev:jenkins -n pipeline-${GUID}-prod

1. Enable the pulling of images from the pipeline-${GUID}-dev project to the pipeline-${GUID}-test and pipeline-${GUID}-prod projects:

oc policy add-role-to-group system:image-puller system:serviceaccounts:pipeline-${GUID}-test -n pipeline-${GUID}-dev

oc policy add-role-to-group system:image-puller system:serviceaccounts:pipeline-${GUID}-prod -n pipeline-${GUID}-dev

### 1.4. Deploy Mock Applications

1. Deploy the "Cat of The Day" (cotd) application in the dev project:

oc new-app https://github.com/StefanoPicozzi/cotd.git -n pipeline-${GUID}-dev

1. Watch the build logs until the build is complete:

sleep 5 # give OpenShift a chance to start the build

oc logs -f build/cotd-1 -n pipeline-${GUID}-dev

1. Check that the build has completed and tag the image:

oc tag cotd:latest cotd:testready -n pipeline-${GUID}-dev

oc tag cotd:testready cotd:prodready -n pipeline-${GUID}-dev

1. Check the image stream to see that the tags were created:

oc describe is cotd -n pipeline-${GUID}-dev

Name: cotd

Created: About an hour ago

Labels: app=cotd

Annotations: openshift.io/generated-by=OpenShiftNewApp

Docker Pull Spec: 172.30.99.85:5000/pipeline-mydemo-dev/cotd

Tag Spec Created PullSpec Image

latest <pushed> About an hour ago 172.30.99.85:5000/pipeline-mydemo-dev/cotd@sha256:21c16f04309942... <same>

prodready cotd@sha256:21c16f04309942... About an hour ago 172.30.99.85:5000/pipeline-mydemo-dev/cotd@sha256:21c16f04309942... <same>

testready cotd@sha256:21c16f04309942... About an hour ago 172.30.99.85:5000/pipeline-mydemo-dev/cotd@sha256:21c16f04309942... <same>

1. Deploy the cotd application in the test and prod projects:

oc new-app pipeline-${GUID}-dev/cotd:testready --name=cotd -n pipeline-${GUID}-test

oc new-app pipeline-${GUID}-dev/cotd:prodready --name=cotd -n pipeline-${GUID}-prod

1. Create routes for all three applications:

oc expose service cotd -n pipeline-${GUID}-dev

oc expose service cotd -n pipeline-${GUID}-test

oc expose service cotd -n pipeline-${GUID}-prod

1. Disable automatic deployment for all deployment configurations in your demonstration:

oc get dc cotd -o yaml -n pipeline-${GUID}-dev | sed 's/automatic: true/automatic: false/g' | oc replace -f -

oc get dc cotd -o yaml -n pipeline-${GUID}-test| sed 's/automatic: true/automatic: false/g' | oc replace -f -

oc get dc cotd -o yaml -n pipeline-${GUID}-prod | sed 's/automatic: true/automatic: false/g' | oc replace -f -

### 1.5. Create Initial Build Config Pipeline

In this section, you create the initial Build Config pipeline using one of the two methods listed below.

Sample Build Config Pipeline

apiVersion: v1

items:

- kind: "BuildConfig"

apiVersion: "v1"

metadata:

name: "pipeline-demo"

spec:

triggers:

- github:

secret: 5Mlic4Le

type: GitHub

- generic:

secret: FiArdDBH

type: Generic

strategy:

type: "JenkinsPipeline"

jenkinsPipelineStrategy:

jenkinsfile: |

node {

stage ("Build") {

echo '\*\*\* Build Starting \*\*\*'

openshiftBuild bldCfg: 'cotd', buildName: '', checkForTriggeredDeployments: 'false', commitID: '', namespace: '', showBuildLogs: 'true', verbose: 'true'

openshiftVerifyBuild bldCfg: 'cotd', checkForTriggeredDeployments: 'false', namespace: '', verbose: 'false'

echo '\*\*\* Build Complete \*\*\*'

}

stage ("Deploy and Verify in Development Env") {

echo '\*\*\* Deployment Starting \*\*\*'

openshiftDeploy depCfg: 'cotd', namespace: '', verbose: 'false', waitTime: ''

openshiftVerifyDeployment authToken: '', depCfg: 'cotd', namespace: '', replicaCount: '1', verbose: 'false', verifyReplicaCount: 'false', waitTime: ''

echo '\*\*\* Deployment Complete \*\*\*'

}

}

kind: List

metadata: {}

1. Before you continue, verify that Jenkins is deployed and make sure you can connect to the Jenkins Web Interface (user: "admin", password:"password"), to find out the route use:

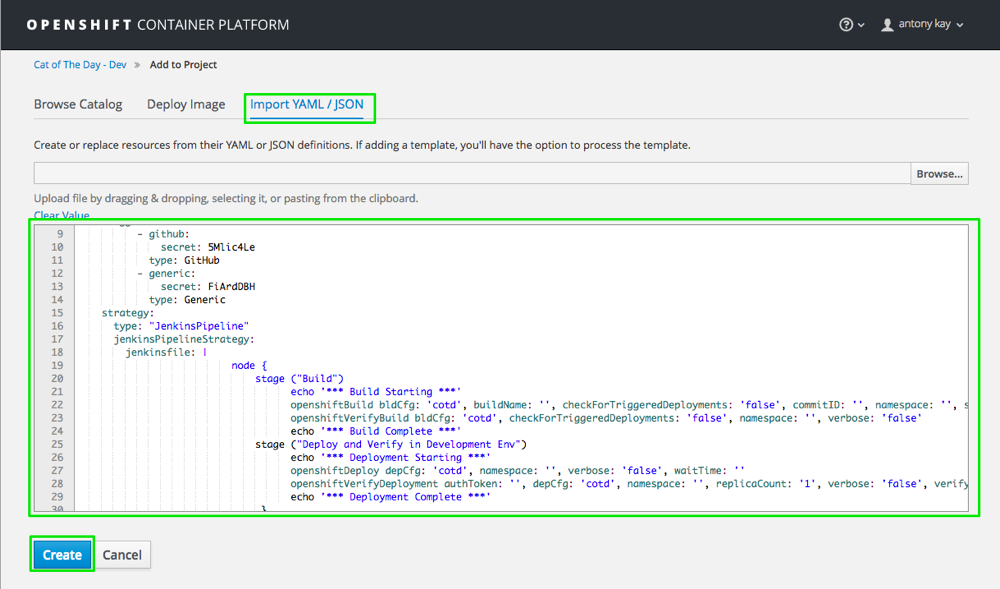
oc get routes -n pipeline-${GUID}-dev

Method 1: Use the command line from any host with the OpenShift client:

* 1. Create a file with the contents of the Build Config pipeline sample above.
  2. Use the oc create -f FILENAME.yaml command to create the Build Config pipeline.

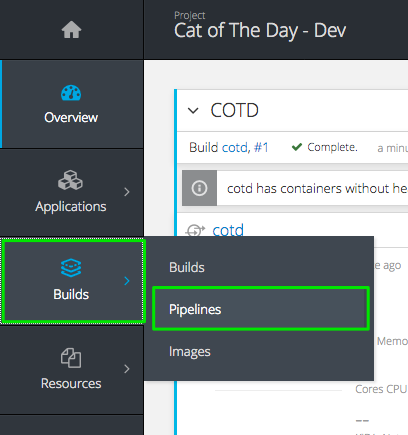
Method 2: Use the OpenShift web console at [https://master.na.openshift.opentlc.com](https://master.na.openshift.opentlc.com/) :

* 1. Log in to the OPENTLC shared OpenShift web console.
  2. Select your dev project and click Add to Project.
  3. Select the Import YAML/JSON tab, paste the Build Config pipeline text in the text box and click Create:

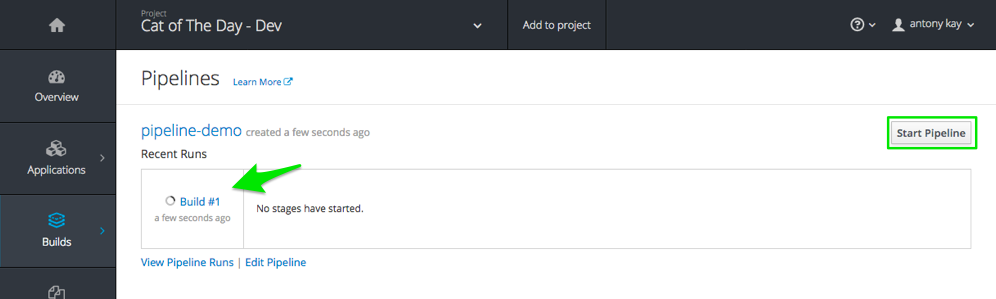


### 1.6. Test the New Pipeline

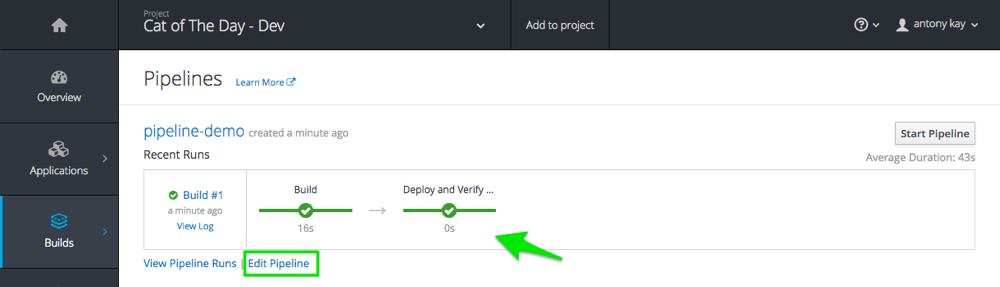
1. Using the OpenShift web console, make sure that the pipeline has been created:
   1. Select Builds → Pipelines:



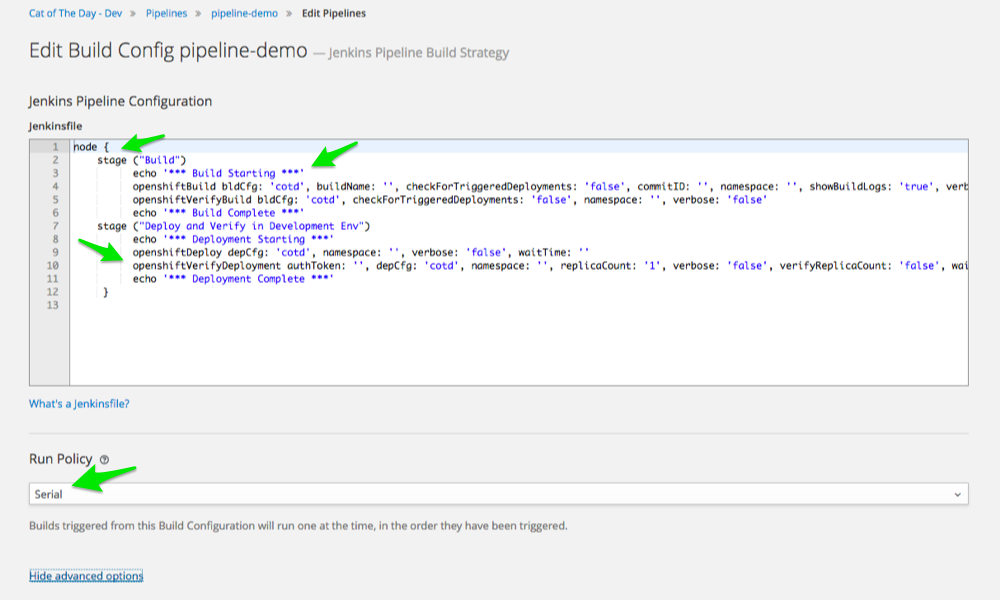
* 1. Click Start Pipeline:



1. Wait for the pipeline to complete.
   1. Make a note of the time it takes to build.
2. Click Edit Pipeline:

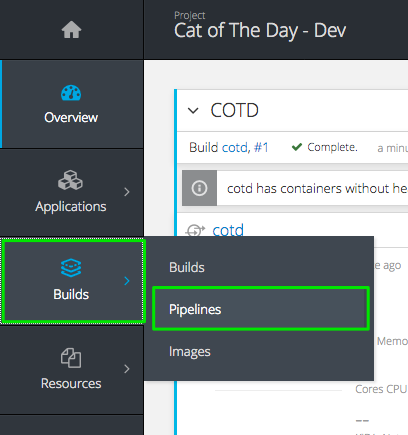


1. Review the pipeline and note the following:
   1. This basic pipeline has only two stages.
   2. Each stage has multiple actions that you can define.
   3. In the Build stage, the pipeline triggers a build and verifies that it was completed successfully.
   4. In the Deploy… stage, the pipeline triggers a new deployment and verifies that the containers deployed successfully.
   5. The serial run policy is set so that only one build runs at a time:

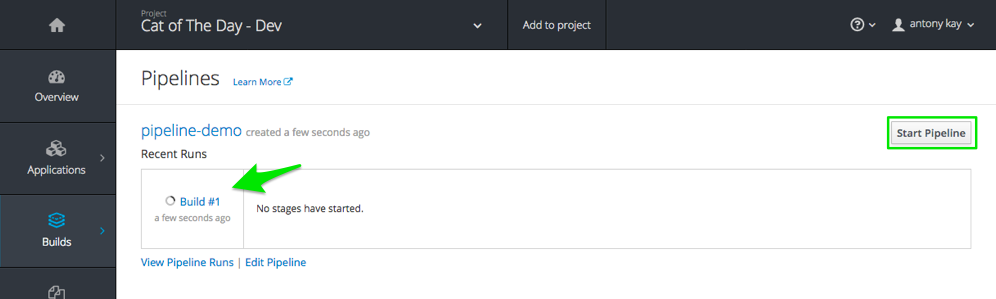


## 2. Demonstrate OpenShift Pipelines Integration

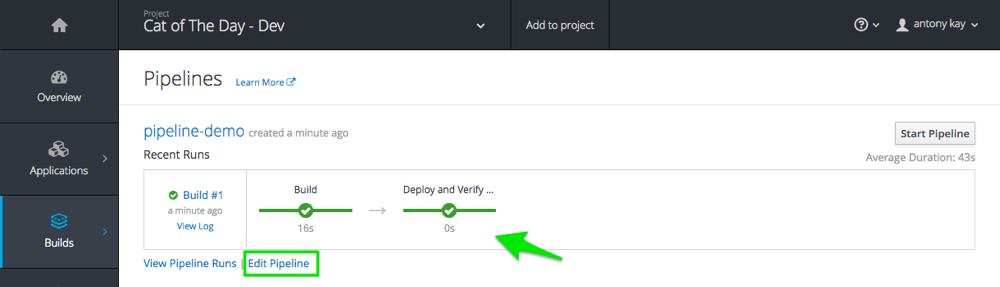
1. Using the OpenShift web console, start with a tour of your deployed environment:
   * Explain and Show that there are three projects in your environment representing three stages in the application life cycle, and that each project has a similar deployment for the same application:
     + The development project contains your Jenkins CI container and other development stage containers.
     + The test project is used to test containers and integration to other applications and data sources in a confined environment.
     + The production project has containers that run on your production nodes and are the "live" version of your application. You can scale deployment on demand.
2. Action: Select Builds → Pipelines:



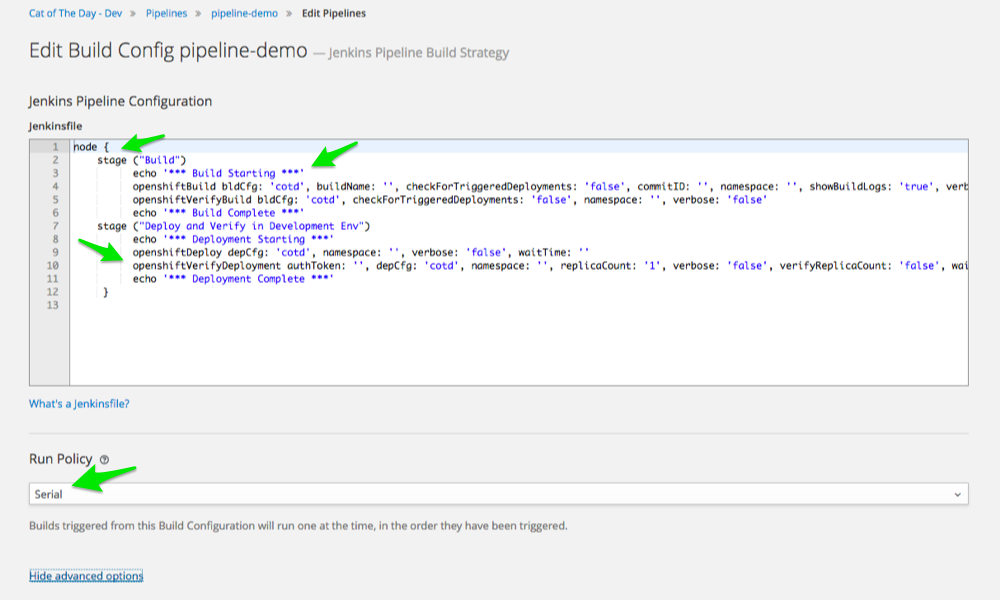
1. Action: Click Start Pipeline.
   * Explain that pipelines are made up of stages, and that you can easily manage and monitor pipelines in Jenkins using OpenShift:



1. Action: Select Edit Pipeline:



* + Explain the contents of the Pipelines page and point out the following:
    - This basic pipeline has just two stages.
    - Each stage has multiple actions that you can define.
    - In the Build stage, the pipeline triggers a build and verifies that it was completed successfully.
    - In the Deploy… stage, the pipeline triggers a new deployment and verifies that the containers deployed successfully.
    - The serial run policy is set so that only one build runs at a time:



1. Action: Edit the pipeline to add a few more steps by copying and pasting the contents of the following CICD Build Config pipeline:

|  |  |
| --- | --- |
|  | Make sure to change the value of GUID to your unique identifier. |

Example 1. CICD Build Config Pipeline

node {

withEnv(['GUID=mydemo']) {

stage ("Build") {

echo '\*\*\* Build Starting \*\*\*'

openshiftBuild bldCfg: 'cotd', buildName: '', checkForTriggeredDeployments: 'false', commitID: '', namespace: '', showBuildLogs: 'false', verbose: 'false', waitTime: ''

openshiftVerifyBuild apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', bldCfg: 'cotd', checkForTriggeredDeployments: 'false', namespace: '', verbose: 'false'

echo '\*\*\* Build Complete \*\*\*'

}

stage ("Deploy and Verify in Development Env") {

echo '\*\*\* Deployment Starting \*\*\*'

openshiftDeploy apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', depCfg: 'cotd', namespace: '', verbose: 'false', waitTime: ''

openshiftVerifyDeployment apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', depCfg: 'cotd', namespace: '', replicaCount: '1', verbose: 'false', verifyReplicaCount: 'false', waitTime: ''

echo '\*\*\* Deployment Complete \*\*\*'

echo '\*\*\* Service Verification Starting \*\*\*'

openshiftVerifyService apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', namespace: 'pipeline-${GUID}-dev', svcName: 'cotd', verbose: 'false'

echo '\*\*\* Service Verification Complete \*\*\*'

openshiftTag(srcStream: 'cotd', srcTag: 'latest', destStream: 'cotd', destTag: 'testready')

}

stage ('Deploy and Test in Testing Env') {

echo "\*\*\* Deploy testready build in pipeline-${GUID}-test project \*\*\*"

openshiftDeploy apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', depCfg: 'cotd', namespace: 'pipeline-${GUID}-test', verbose: 'false', waitTime: ''

openshiftVerifyDeployment apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', depCfg: 'cotd', namespace: 'pipeline-${GUID}-test', replicaCount: '1', verbose: 'false', verifyReplicaCount: 'false'

sleep 10

sh 'curl http://cotd-pipeline-${GUID}-test.apps.na1.openshift.opentlc.com/data/ | grep cats -q'

}

stage ('Promote and Verify in Production Env') {

echo '\*\*\* Waiting for Input \*\*\*'

input 'Should we deploy to Production?'

openshiftTag(srcStream: 'cotd', srcTag: 'testready', destStream: 'cotd', destTag: 'prodready')

echo '\*\*\* Deploying to Production \*\*\*'

openshiftDeploy apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', depCfg: 'cotd', namespace: 'pipeline-${GUID}-prod', verbose: 'false', waitTime: ''

openshiftVerifyDeployment apiURL: 'https://openshift.default.svc.cluster.local', authToken: '', depCfg: 'cotd', namespace: 'pipeline-${GUID}-prod', replicaCount: '1', verbose: 'false', verifyReplicaCount: 'false'

sleep 10

sh 'curl http://cotd-pipeline-${GUID}-prod.apps.na1.openshift.opentlc.com/data/ | grep cats -q'

}

}

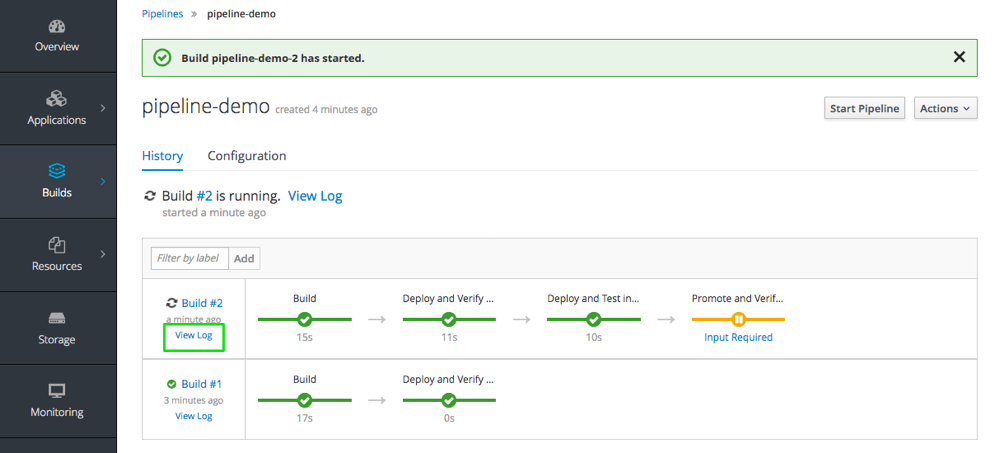
}

|  |  |
| --- | --- |
|  | To save time, you can save the changes, start a new build, and return to the Edit page to review the pipeline code. |

* + Explain the following aspects of the pipeline:
    - This CICD pipeline has four stages.
    - Each stage has multiple actions that you can define.
    - The serial run policy is set so that only one build runs at a time.
  + Explain the different stages in the pipeline:
    - Build: OpenShift builds and verifies a successful build.
    - Deploy and Verify in Development Env: OpenShift deploys the latest application in the development project, verifies that the container and service were deployed correctly, and tags the image as "testready".
    - Deploy and Test in Testing Env: OpenShift deploys the image tagged "testready" in the testing project, runs different integration tests—in this case, just a cURL command—and tags the image as "prodready" if all tests pass.
    - Promote and Verify in Production Env: OpenShift waits for human approval, then deploys and verifies the container and service in the production project.

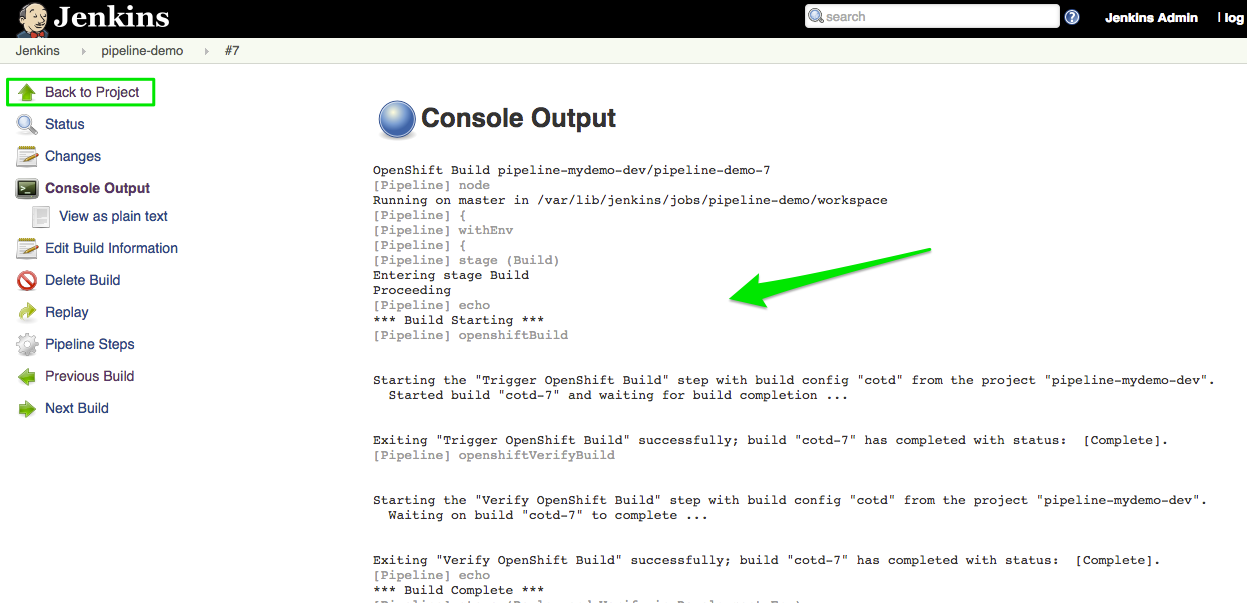
1. Action: Click Save and start a new pipeline build.
2. Action: While you wait for the pipeline to complete its run, click View Log on one of the previous pipeline builds (either a successful or a failed pipeline):

|  |  |
| --- | --- |
|  | You need to authenticate to Jenkins as admin using the default password password. |

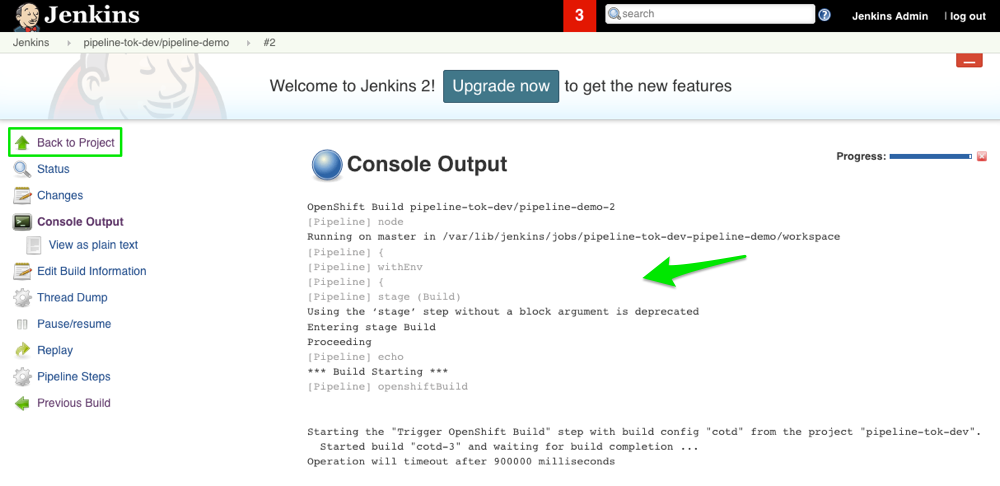


* + Explain that OpenShift directs you to the Jenkins console log, where you can manage and monitor builds from the Jenkins interface.

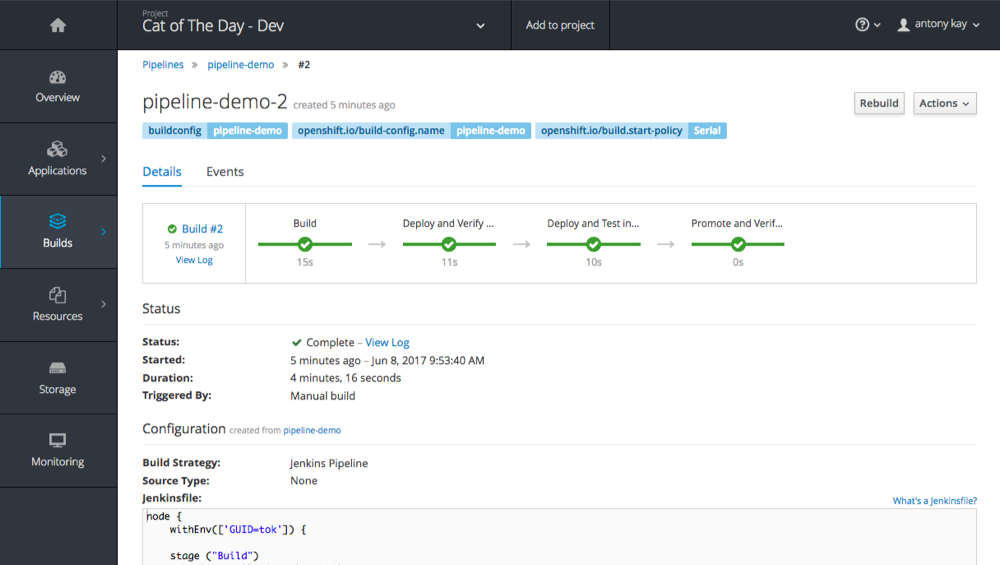
1. Action: Click Back to Project when you are done:



* + Explain that the Jenkins web interface also allows you to monitor, configure, and manage your pipelines.
    - Point out that the Build Now button can be used to start a new pipeline.
    - Note that graphical progress representation is available for the different pipelines and their stages.
    - [OPTIONAL] Show the Pipeline Syntax page and demonstrate how easy it is to create a new pipeline.



1. Action: Go back to the OpenShift web console, and if the Build is waiting for input, click Input Required and select Proceed.
2. Action: Go back to the OpenShift web console and ask for questions:



## 3. Additional Demonstrations

You can do the following optional demonstrations:

* Edit the pipeline and remove the manual input step.
* Show how to scale a deployment up and down.
* Show metrics and logging for a container.
* Show the quota restrictions on your project.