

Lab (Option B: JavaScript): CI/CD with OpenShift Pipelines



Estimated time needed: 45 minutes

Welcome to the hands-on lab for **CI/CD with OpenShift Pipelines**. In this lab, you will create a CI/CD workflow using the OpenShift Pipelines.

Learning objectives

After completing this lab, you will be able to:

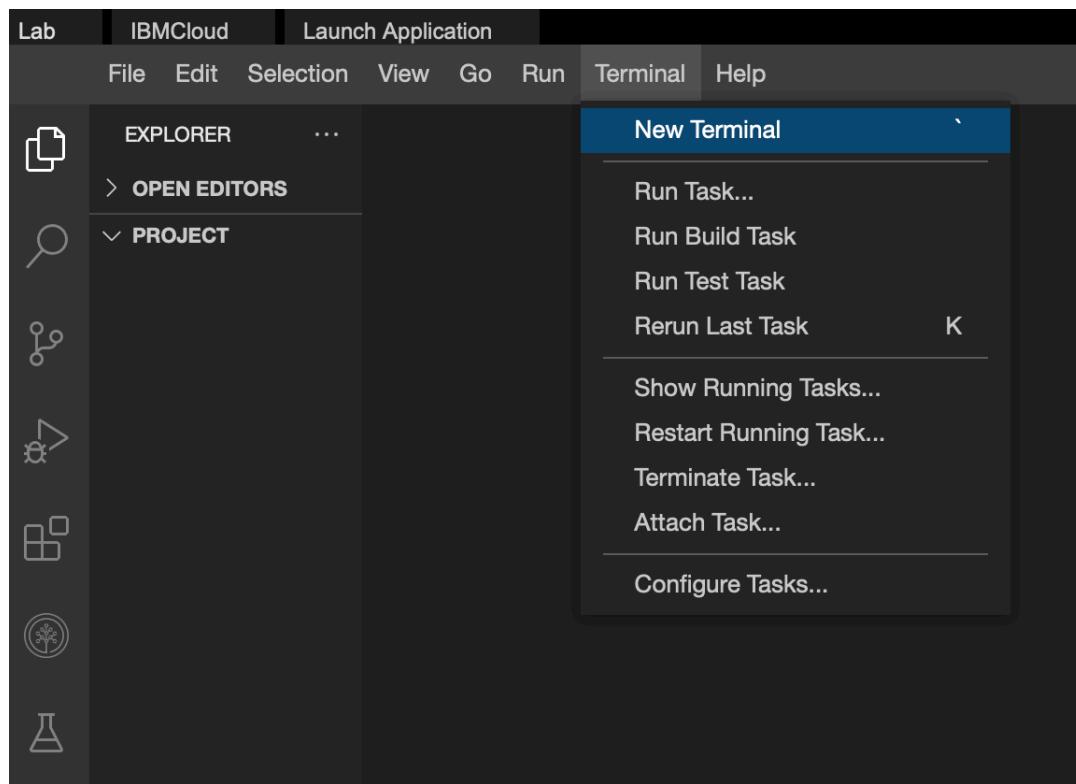
- Create a CI/CD workflow using the OpenShift Pipelines
- Add parameters to tasks created using OpenShift Pipelines
- Add a workspace and persistent volume claim in the OpenShift UI
- Add tasks that clone the GitHub repository, lint the source code, run unit tests, and finally deploy the application to the OpenShift cluster

Set up the lab environment

You have a little preparation to do before you can start the lab.

Open a terminal

Open a terminal window by using the menu in the editor: Terminal > New Terminal.



In the terminal, if you are not already in the `/home/project` folder, change to your project folder now.

```
cd /home/project
```

You can use the following command to ensure you are connected to an OpenShift cluster:

```
oc config current-context
```

You should see something like:

```
harshsingh15-context
```

You are now ready to continue installing the **Prerequisites**.

Optional

If working in the terminal becomes difficult because the command prompt is very long, you can shorten the prompt using the following command:

```
export PS1="\[\e[01;32m\]\u\[\e[00m\]: \[\e[033[01;34m\]\w\[\e[033[00m\]]\$ "
```

Prerequisites

This lab requires the installation of the tasks introduced in the previous labs. To be sure, apply the previous tasks to your cluster before proceeding. Reissue these commands:

Establish the tasks

First create an empty file called `tasks.yaml` in the root folder:

```
touch tasks.yaml
```

Open the `tasks.yaml` file and add the following yaml content.

[Open tasks.yaml in IDE](#)

```
---  
apiVersion: tekton.dev/v1beta1  
kind: Task  
metadata:  
  name: cleanup  
spec:  
  description: This task will clean up a workspace by deleting all the files.  
  workspaces:  
    - name: source  
  steps:  
    - name: remove  
      image: alpine:3  
      env:  
        - name: WORKSPACE_SOURCE_PATH  
          value: ${workspaces.source.path}  
      workingDir: ${workspaces.source.path}  
      securityContext:  
        runAsNonRoot: false  
        runAsUser: 0  
      script: |  
        #!/usr/bin/env sh  
        set -eu  
        echo "Removing all files from ${WORKSPACE_SOURCE_PATH} ..."  
        # Delete any existing contents of the directory if it exists.  
        #  
        # We don't just "rm -rf ${WORKSPACE_SOURCE_PATH}" because ${WORKSPACE_SOURCE_PATH} might be "/"  
        # or the root of a mounted volume.  
        if [ -d "${WORKSPACE_SOURCE_PATH}" ] ; then  
          # Delete non-hidden files and directories  
          rm -rf "${WORKSPACE_SOURCE_PATH:?}/*"  
          # Delete files and directories starting with . but excluding ..  
          rm -rf "${WORKSPACE_SOURCE_PATH}"/[!.]*  
          # Delete files and directories starting with .. plus any other character  
          rm -rf "${WORKSPACE_SOURCE_PATH}"/..?*  
        fi  
---  
apiVersion: tekton.dev/v1beta1  
kind: Task  
metadata:  
  name: eslint  
spec:  
  workspaces:  
    - name: source  
  steps:  
    - name: install-dependencies  
      image: node:18  
      workingDir: ${workspaces.source.path}  
      script: |  
        npm install  
    - name: run-eslint  
      image: node:18  
      workingDir: ${workspaces.source.path}  
      script: |  
        npx eslint .  
---  
apiVersion: tekton.dev/v1beta1  
kind: Task  
metadata:  
  name: jest  
spec:  
  workspaces:  
    - name: source  
  params:  
    - name: args  
      description: Arguments to pass to jest  
      type: string  
      default: "--passWithNoTests"  
  steps:  
    - name: jest-tests  
      image: node:18-alpine  
      workingDir: ${workspaces.source.path}  
      script: |  
        #!/bin/sh  
        set -e  
        # Install dependencies  
        if [ ! -f "package-lock.json" ] ; then  
          npm ci  
        elif [ ! -f "yarn.lock" ] ; then  
          yarn install --frozen-lockfile  
        else  
          npm install  
        fi  
        # Run tests with Jest  
        npm test ${params.args}
```

Make sure you save the file. Next, apply the tasks to your OpenShift Cluster:

```
kubectl apply -f tasks.yaml
```

Check that you have all of the previous tasks installed:

```
oc get tasks
```

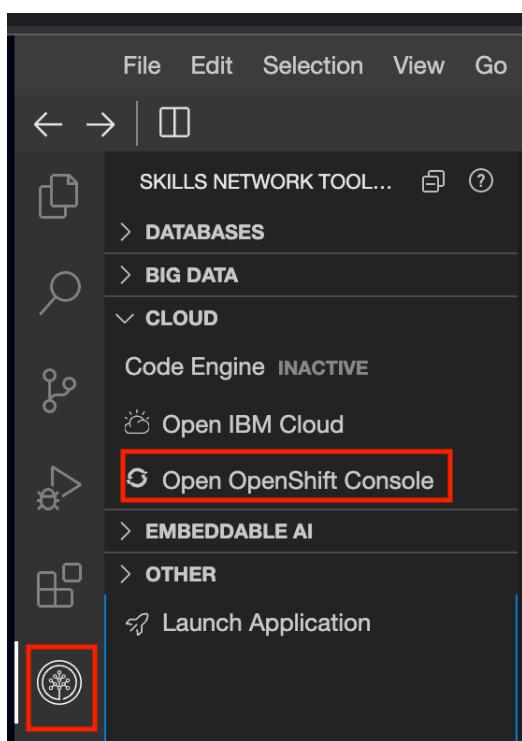
You should see the output similar to this:

NAME	AGE
cleanup	16s
eslint	16s
jest	16s

Step 1: Create PersistentVolumeClaim

You also need a PersistentVolumeClaim (PVC) to use as a workspace. You can use the OpenShift Administrator perspective to create the PVC.

Open the OpenShift console using the **Open OpenShift Console** under the **Skills Network Toolbox** menu.



The lab should open the **Developer** perspective for the OpenShift console in a new tab.

← → G harshsingh15-console.openshift-sandbox.labs.cognitiveclass.ai/k8s/cluster/projects/sn-labs-h

☰ Skills Network OpenShift Lab

Developer

+Add

Topology

Search

Builds

Pipelines

Helm

Project

ConfigMaps

Secrets

Projects > Project details

PR sn-labs-harshsingh15 Active

Overview Details YAML Workloads RoleBindings

Details

Name sn-labs-harshsingh15

Requester No requester

Labels kubernetes.io/metadata.name=sn-labs-harshsingh15
learnersandbox=true name=sn-labs-harshsingh15

Description No description

View all

Inventory

1 Deployment
0 DeploymentConfigs
0 StatefulSets
2 Pods
0 PersistentVolumeClaims
1 Service
0 Routes
5 ConfigMaps
11 Secrets
VolumeSnapshots Not available

Status

Active

Utilization

Resource

CPU

Memory

Filesystem

Network transfer

Pod count

ResourceQuotas

> RQ harshsingh15-resourcequota

AppliedClusterResourceQuotas

No AppliedClusterResourceQuotas

Open the **Administrator** perspective using the drop-down on the left side of the screen.



Developer

Administrator

Developer

Search

Builds

Pipelines

Helm

Project

ConfigMaps

Secrets

Once the page switches to the Administrator view, click **Storage** and **PersistentVolumeClaims**.

The screenshot shows the Skills Network OpenShift Lab interface. On the left, a sidebar menu is open under the 'Storage' category. The 'PersistentVolumeClaims' item is highlighted with a red box. The main content area displays a 'Projects' list with one item: 'sn-labs-harhsingh15'. The interface has a dark theme with light-colored text and icons.

- Administrator
- Home
- Operators
- Workloads
- Networking
- Storage
 - PersistentVolumeClaims
 - StorageClasses
 - VolumeSnapshots
 - VolumeSnapshotClasses
- Builds
- Pipelines
- User Management
- Administration

Projects

Filter Name S

Name ↑

PR sn-labs-harhsingh15

Note: If you encounter an error when opening OpenShift and accessing the Persistent Claim value, please close the OpenShift window and then reopen it.

Click `Create PersistentVolumeClaim` to create a new PVC:

The screenshot shows the OpenShift web interface. The left sidebar is dark-themed and includes a user dropdown for 'Administrator'. The main navigation bar at the top shows the project 'sn-labs-harshsingh15'. The main content area is titled 'PersistentVolumeClaims' and lists several sub-options: StorageClasses, VolumeSnapshots, VolumeSnapshotClasses, Builds, Pipelines, User Management, and Administration. The 'PersistentVolumeClaims' option is currently selected, indicated by a blue border around its corresponding row.

Next, fill out the form as follows:

- StorageClass: skills-network-learner
- PersistentVolumeClaim name: oc-lab-pvc
- Size: 1GiB

The screenshot shows the 'Create PersistentVolumeClaim' dialog box. At the top, it says 'Project: sn-labs-harshsingh15'. Below that is the title 'Create PersistentVolumeClaim' and a 'Edit YAML' link. The form fields are as follows:

- StorageClass**: A dropdown menu with 'SC skills-network-learner' selected. This field is highlighted with a red box.
- PersistentVolumeClaim name ***: A text input field containing 'oc-lab-pvc'. This field is also highlighted with a red box.
- Access mode ***: Radio buttons for 'Single user (RWO)', 'Shared access (RWX)', and 'Read only (RO)'. The first option is selected.
- Permissions to the mounted drive**: A section for specifying storage capacity.
- Size ***: A slider with values - (minus), 1 (selected), + (plus), and 'GiB' with a dropdown arrow. The value '1' is highlighted with a red box.
- Desired storage capacity**: A note below the slider.
- Use label selectors to request storage**: A checkbox that is unchecked.
- PersistentVolume resources**: A note indicating that resources matching label selectors will be considered for binding.
- Volume mode ***: Radio buttons for 'Filesystem' (selected) and 'Block'.
- Create**: A blue button at the bottom left. This button is highlighted with a red box.
- Cancel**: A grey button at the bottom right.

Finally, click **Create** to create the PVC. Once the PVC is created, you should see the details. Notice the **Status** is **Pending**. It takes a few minutes for the PVC to complete. You don't have to wait for this to finish, as it will most likely be in place by the time you need it in the pipeline.

The screenshot shows the Skills Network OpenShift Lab interface. On the left, there is a sidebar with navigation links: Home, Operators, Workloads, Networking, Storage (with PersistentVolumeClaims selected), StorageClasses, VolumeSnapshots, VolumeSnapshotClasses, Builds, Pipelines, User Management, and Administration. The main content area shows the following details for a PersistentVolumeClaim named "oc-lab-pvc":

- Project:** sn-labs-harshsingh15
- PVC:** oc-lab-pvc (Status: Pending)
- Details** (selected), **YAML**, **Events**, **VolumeSnapshots**
- PersistentVolumeClaim details**
- Name:** oc-lab-pvc
- Namespace:** NS sn-labs-harshsingh15
- Labels:** No labels
- Annotations:** 0 annotations
- Label selector:** No selector
- Created at:** 24 Jun 2025, 01:12
- Owner:** No owner
- Conditions:** (empty)

Note: If you encounter an error when opening OpenShift and accessing the Persistent Claim value, please close the OpenShift window and then reopen it.

Click **Create PersistentVolumeClaim** to create a new PVC:



Administrator

Home

Operators

Workloads

Networking

Storage

PersistentVolumeClaims

StorageClasses

VolumeSnapshots

VolumeSnapshotClasses

Project: sn-labs-captainfedo1

PersistentVolumeClaims

No PersistentVolume

Next, fill out the form as follows:

- StorageClass: skills-network-learner
- PersistentVolumeClaim name: oc-lab-pvc
- Size: 16G

Create PersistentVolumeClaim

[Edit YAML](#)**StorageClass** skills-network-learner

StorageClass for the new claim

PersistentVolumeClaim name *

oc-lab-pvc

A unique name for the storage claim within the project

Access mode *

-
- Single user (RWO)
-
- Shared access (RWX)
-
- Read only (ROX)

Permissions to the mounted drive

Size *

-	1	+
---	---	---

 GiB ▾

Desired storage capacity

 Use label selectors to request storage

PersistentVolume resources that match all label selectors will be considered for binding.

Volume mode *

-
- Filesystem
-
- Block

[Create](#)[Cancel](#)

Finally, click Create to create the PVC. Once the PVC is created, you should see the details. Notice the Status is Pending. It takes a few minutes for the PVC to complete. You don't have to wait for this to finish as it will most likely be in place by the time you need it in the pipeline.



- Administrator**
- Home >
- Operators >
- Workloads >
- Networking >
- Storage >
 - PersistentVolumeClaims
 - StorageClasses
 - VolumeSnapshots
 - VolumeSnapshotClasses
- Builds >
- Pipelines >
- User Management >
- Administration >

Project: sn-labs-captainfedor1 ▾

PersistentVolumeClaims > PersistentVolumeClaim details

PVC **oc-lab-pvc** 🕒 Pending

Details YAML Events VolumeSnapshots

PersistentVolumeClaim details

Name **oc-lab-pvc**

Namespace **NS** **sn-labs-captainfedor1**

Labels Edit

No labels

Annotations **0 annotations** Edit

Label selector
No selector

Created at **Aug 27, 2023, 11:25 AM**

Owner
No owner

Note:- In case you face permission security error in the OpenShift Console while creating a PVC, you can create a pvc through terminal using the steps below:-

Option 2 for creating a PVC through terminal:-

You start by creating a **PersistentVolumeClaim** (PVC) to use as the workspace:

A workspace is a disk volume that can be shared across tasks. The way to bind to volumes in Kubernetes is with a **PersistentVolumeClaim**.

Create a **pvc.yaml** file with these contents:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pipelinerun-pvc
spec:
  storageClassName: skills-network-learner
  resources:
    requests:
      storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
```

```
kubectl apply -f pvc.yaml
```

You should see the following output:

```
persistentvolumeclaim/pipelinrun-pvc created
```

You can now use this persistent volume `pipelinrun-pvc` as a workspace in your Tekton tasks by selecting it from the dropdown in the OpenShift Console.

Step 2: Create a new Pipeline

Now that you have a PVC in place, the next step is to start working on the pipeline. First, go back to the **Developer** perspective.

The screenshot shows the OpenShift developer perspective interface. On the left, there's a navigation sidebar with sections like Project, PersistentVolumeClaims, Storage, Pipelines, and Administration. The 'PersistentVolumeClaims' section is currently selected and highlighted in blue. In the main content area, the title is 'PersistentVolumeClaim details' for a resource named 'oc-lab-pvc'. The details shown include:

- Name:** oc-lab-pvc
- Namespace:** NS sn-labs-harshsingh15
- Labels:** No labels
- Annotations:** 0 annotations
- Label selector:** No selector
- Created at:** 24 Jun 2025, 01:12
- Owner:** No owner

Below these details is a section titled 'Conditions'.

Next, click **Pipelines** on the left panel and create a new pipeline.



Developer

Project: sn-labs-harshsingh15 ▾

+Add

Topology

Search

Builds

Pipelines

Helm

Project

ConfigMaps

Secrets

Pipelines

[Pipelines](#)[PipelineRuns](#)[Repositories](#)

You are presented with the pipeline builder. Ensure that you have **Pipeline Builder** selected in **Configure Via** and enter **ci-cd-pipeline** as the name of your pipeline.



Developer

+Add

Topology

Search

Builds

Pipelines

Helm

Project

ConfigMaps

Secrets

Project: sn-labs-harshsingh15

Pipeline builder

Configure via: Pipeline builder YAML view

Name *

ci-cd-pipeline

Tasks *

Add task

Add finally task

Parameters

No parameters are associated with this Pipeline.

[+ Add parameter](#)

Workspaces

No workspaces are associated with this pipeline.

[+ Add workspace](#)[Create](#)[Cancel](#)

Before you create your first task, let's add a workspace to your pipeline. Scroll to the bottom of the page and add a new workspace with the name `output`. This workspace will be used to clone the code.



Parameters

No parameters are associated with this Pipeline.

[+ Add parameter](#)

Workspaces

Name *

output

Optional workspace

[+ Add workspace](#)

Great! We can now start adding tasks to your pipeline.

Step 3: Add the cleanup task

You were asked to apply a `tasks.yaml` file that contained the `jest` and the `nose` tasks. You can confirm the tasks are installed by using the following command:

```
oc get tasks
```

You should see the output similar to this:

NAME	AGE
cleanup	16s
eslint	16s
jest	16s

If you don't see both of these tasks, go back to the `Prerequisites` step and make sure you apply the `tasks.yaml` file.

You will create the first task in this step. Click **Add Task** in the builder UI to open the `Add task ...` dialog.

Pipeline builder

Configure via: Pipeline builder YAML view

Name *

ci-cd-pipeline



Add task...

Tasks *

Add task

Add finally task

Type cleanup to see the task you installed earlier from the yaml file. Click Add to use the task in the builder.

Pipeline builder

Configure via: Pipeline builder YAML view

Name *

ci-cd-pipeline

Tasks *

[Add task](#)

[!\[\]\(f2985ce41098c80ba580115aeca2d55e_img.jpg\) clean](#)

 **boskos-release**
Community Cloud

 **cleanup**
Red Hat Red Hat

 **orka-full**
Community Cloud

 **orka-teardown**
Community Cloud

boskos-release[Install and add](#)

0.1 (latest)

i This task is not installed

Adding this task may take a few moments.

Release a project acquired using Boskos will release the specified resource from the server-url. It also assumes the resource to acquire and so terminates the hear that Task to keep the resource obtained dirty so that the boskos Janitor will created). [Read more ↗](#)

[Categories](#) [Cloud](#)

Parameters

No parameters are associated with this Pipeline.

This should install your first task. You will notice the red exclamation mark on the task. This means the task has not yet been completely configured. Click the task to open the task flyout. Change the workspace to **output**.

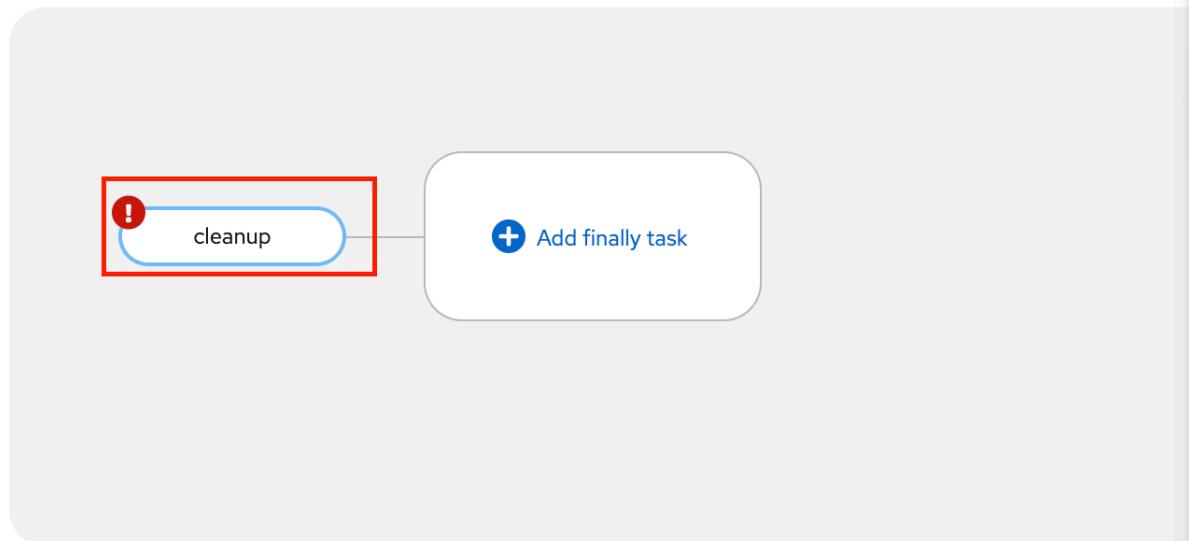
Pipeline builder

T cleanConfigure via: Pipeline builder YAML view**Name ***

ci-cd-pipeline

Display name

cleanup

Tasks ***Workspac****source *** Select work output**When exp**

No when expres

Add when e

Parameters

You should see the exclamation mark go away and **Create** enabled. Click **Create** to finish creating the task in the pipeline.

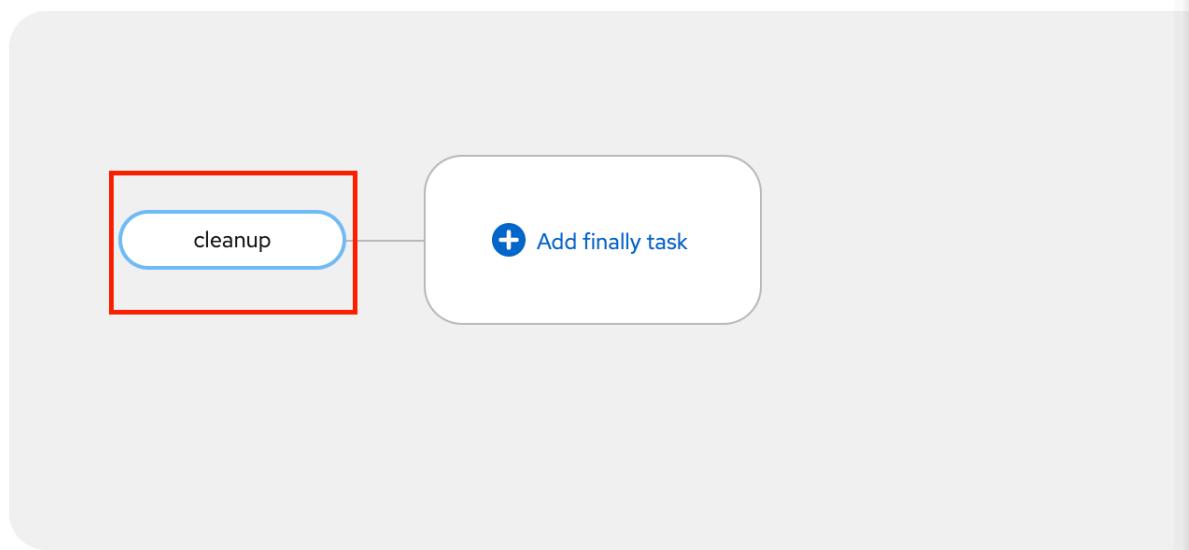
Pipeline builder

T cleanConfigure via: Pipeline builder YAML view**Name ***

ci-cd-pipeline

Display name

cleanup

Tasks ***Workspac****source ***

output

When exp

No when expres

+ Add when e

Parameters

No parameters are associated with this Pipeline.

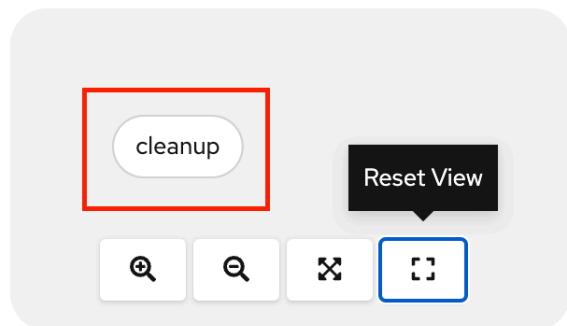
+ Add parameter**Create****Cancel**

You should now see your pipeline with the one task you just added.

PL ci-cd-pipeline

Details YAML PipelineRuns Parameters Resources Metrics

Pipeline details



Name

ci-cd-pipeline

Tasks

T cleanup

Namespace

NS sn-labs-harshsingh15

Workspaces

output

Labels

Edit

No labels

Annotations

0 annotations

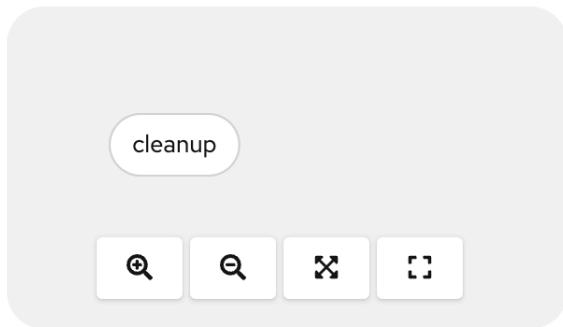
Step 4: Run the pipeline

Now that you have a pipeline with the one cleanup step, let's see how you can run this pipeline. Click **Pipelines** on the left bar, if you are not already on the pipelines page. Click **ci-cd-pipeline** pipeline. You can now use the **Actions** dropdown on the left to run the pipeline.

PL ci-cd-pipeline

[Details](#)[YAML](#)[PipelineRuns](#)[Parameters](#)[Resources](#)[Metrics](#)

Pipeline details

**Name**

ci-cd-pipeline

Tasks

cleanup

Namespace

NS sn-labs-harhsingh15

Workspaces

output

Labels[Edit](#)

No labels

OpenShift brings up the **Start Pipeline** dialog box. Ensure that you pick the following:

- output: PersistentVolumeClaim
- select a PVC: oc-lab-pvc

Click **Start** after you have filled out the form.

Start Pipeline

Workspaces

output *

PersistentVolumeClaim

PVC oc-lab-pvc

Advanced options

- [Show credential options](#)

Name	Tasks
------	-------

You should see the pipeline running on the next page. You can click the task name to see the logs for a particular task. Alternatively, you can click the **Logs** tab:

PLR ci-cd-pipeline-b039mg Succeeded[Details](#)[YAML](#)[TaskRuns](#)[Parameters](#)[Logs](#)[Events](#)

PipelineRun details

**Name**

ci-cd-pipeline-b039mg

Status✓ Succeeded**Namespace**NS sn-labs-harshsingh15**Pipeline**PL ci-cd-pipeline**Labels**[Edit](#)

tekton.dev/pipeline=ci-cd-pipeline

Triggered by:

system:serviceaccount:sr

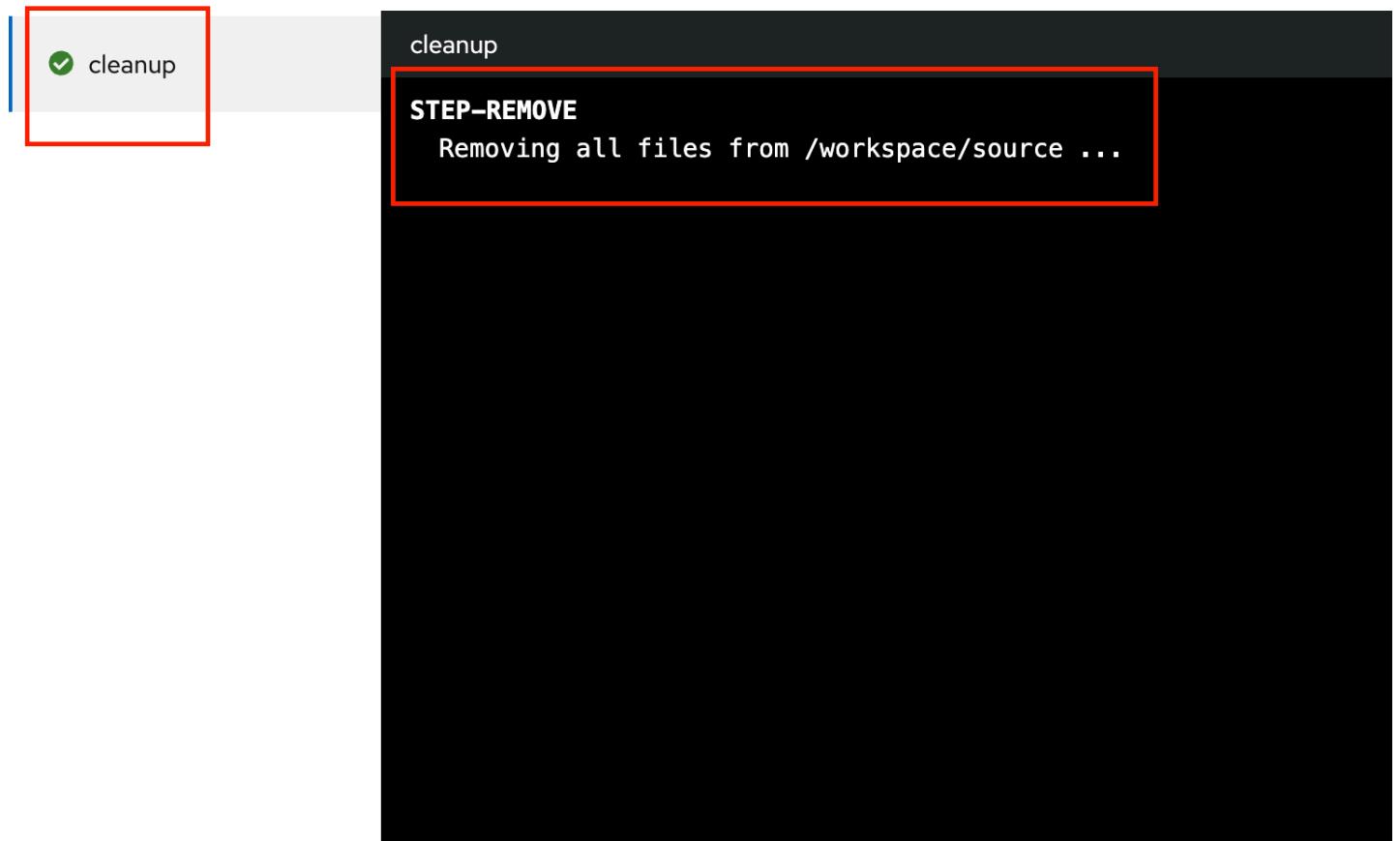
Workspace ResourcesPVC oc-lab-pvc (output)**Annotations**

2 annotations

You can see the detailed logs and also have an option to download them. You will see the task on the right turn green, if it completes successfully.

PLR ci-cd-pipeline-b039mg Succeeded

Details YAML TaskRuns Parameters **Logs** Events



Congratulations! You created a pipeline from scratch and added the cleanup tasks on it. You then ran the pipeline and viewed the logs. This first task was explained in detail as an example. The lab is now asking you to finish the rest of the tasks in this pipeline on your own. Good luck!

Step 5: Add the Git clone task

You are asked to use the `git-clone` in-built task to clone the GitHub code into your pipeline.

Your task

1. Open the pipeline in edit mode. Select Pipeline from the left menu, select the pipeline name, and go to Actions -> Edit Pipeline. See the hint for a screenshot.
2. Add a new task after the cleanup task in the pipeline from the previous step. Hover over the step to display the + buttons. Use the + button on the right of the task to add a task instead of using the `Add finally` task link. See the hint for a screenshot.
3. Look for the RedHat `git-clone` task and add it to the placeholder task.
4. Click the red exclamation on the task or the task card to open the configure task flyout. Configure the `git-clone` task as follows:
 - o `url: https://github.com/ibm-developer-skills-network/ttwst-jhxyb-ci-cd-pipeline_js`
 - o `workspace.output: output`
5. Save the pipeline.
6. Run the pipeline.
7. Check the logs to see if there are issues with the pipeline.

Hint

- Click here for a hint.

Check your solution

- Click here for the solution.

Step 6: Add the ESLint task

You are asked to use the `eslint` task to lint the source code. As part of this task, you will configure the task with specific arguments.

Your task

1. Open the pipeline in edit mode.
2. Add a new task after the `git-clone` task in the pipeline from the previous step.
3. Look for the `eslint` task from the RedHat.
4. Install and add it to the placeholder task.
5. Click the red exclamation on the task or the task card to open the configure task flyout. Configure the `eslint` task as follows:
 - o `workspace.source: output`
6. Save the pipeline.
7. Run the pipeline.
8. Check the logs to see if there are issues with the pipeline.

Hint

► Click here for a hint.

Check your solution

► Click here for the solution.

Step 7: Add the Jest task

The next step is to add the `jest` task for unit testing the source code of the application.

Your task

1. Open the pipeline in edit mode.
2. Add a new task after the `eslint` task in the pipeline from the previous step.
3. Look for the `jest` task.
4. Install and add it to the placeholder task.
5. Click the red exclamation on the task or the task card to open the configure task flyout. Configure the `jest` task as follows:
 - `workspace.source: output`
6. Save the pipeline.
7. Run the pipeline.
8. Check the logs to see if there are issues with the pipeline.

Check your solution

► Click here for the solution.

Step 8: Add the buildah task

The next step is to add a task to create an image from the GitHub source code. You will use the `buildah` in-built task to perform this action.

Your task

1. Open the pipeline in edit mode.
2. Add a new task after the `jest` task in the pipeline from the previous step.
3. Look for the `buildah` task from RedHat.
4. Install and add it to the placeholder task.
5. You will need the namespace of your lab environment for one of the arguments. You can obtain this by using the command line terminal and using the `echo $SN_ICR_NAMESPACE` command in the lab terminal.

```
echo $SN_ICR_NAMESPACE
```

6. Click the red exclamation on the task or the task card to open the configure task flyout. Configure the `buildah` task as follows:
 - `image: $(params.build-image)`
 - `workspace.source: output`

7. Click the main page to close the flyout. Add the following parameter and the default value to the pipeline:
 - `parameter.name: build-image`
 - `parameter.default: image-registry.openshift-image-registry.svc:5000/SN_ICR_NAMESPACE/tekton-lab:latest`.
 - Replace `SN_ICR_NAMESPACE` with the value above.

8. Save the pipeline.

9. Run the pipeline.

10. Check the logs to see if there are issues with the pipeline.

Hint

► Click here for a hint.

Check your solution

► Click here for the solution.

Step 9: Deploy application

Next, you will create a task to deploy the image you created to the lab OpenShift cluster. You will use the `OpenShift client` task to execute the `oc deploy` command with the image you built in the previous step.

Your task

1. Open the pipeline in edit mode.
2. Add a new task after the `buildah` task in the pipeline from the previous step.
3. Look for the `openshift-client` task from RedHat.
4. Install and add it to the placeholder task.
5. Click the red exclamation on the task or the task card to open the configure task flyout. Configure the task with the following
 - `display name: deploy`
 - `SCRIPT: oc create deployment $(params.app-name) --image=$(params.build-image) --dry-run=client -o yaml | oc apply -f -`
6. Click the main page to close the flyout. Add the following parameter and the default value to the pipeline:
 - `parameter.name: app-name`
 - `parameter.default: cicd-app`.
7. Save the pipeline.
8. Run the pipeline.
9. Check the logs to see if there are issues with the pipeline.

Hint

► Click here for a hint.

Check your solution

► Click here for the solution.

Step 10: Validate application

You have done all the hard work! Let's confirm if the application was deployed.

Your tasks

1. Click Topology on the left panel in the Developer perspective. You should see two applications on the canvas.

The screenshot shows the Skills Network OpenShift Lab interface. On the left, a sidebar titled 'Developer' contains links for Topology, Search, Builds, Pipelines, Helm, Project, ConfigMaps, and Secrets. The 'Topology' link is highlighted with a red box. The main area displays two application icons: 'opensh...onsole' and 'cicd-app'. The 'cicd-app' icon is enclosed in a red box. At the top, there are dropdown menus for 'Project: sn-labs-harshsingh15' and 'Application: All applications', along with 'Display options', 'Filter by resource', 'Name', 'Find by name...', and search bar.

2. Click the one called cicd-app to open the flyout. Click logs.

The screenshot shows the Skills Network OpenShift Lab interface with the 'cicd-app' application details flyout open. The flyout includes the application icon, name ('cicd-app'), and a 'Logs' button. The background shows the same developer perspective interface as the previous screenshot.

3. You should see a message `SERVICERUNNING` in the logs indicating the application was deployed successfully and is running.

The screenshot shows the OpenShift UI for a project named "sn-labs-harshsingh15". In the "Pods" section, a pod named "cicd-app-5dfff976cc-lxpz7" is selected, which is currently "Running". The "Logs" tab is active, displaying the following log output:

```
4 lines
1 INFO: ****S E R V I C E R U N N I N G ****
2 INFO: ** S E R V I C E R U N N I N G **
3 INFO: ****S E R V I C E R U N N I N G ****
4 INFO: Server running on http://0.0.0.0:8000
```

Conclusion

Congratulations! You have just created a CI/CD workflow using OpenShift Pipelines without writing a single line of code!

In this lab, you learned how to use the OpenShift UX and the Pipelines feature. You also learned how to install the task locally using the Tekton CLI and how to modify your pipeline in the UX to reference the task and configure its parameters. You then learned how to create default parameters for your pipeline. Finally, you now know how to create a PersistentVolumeClaim using the UX.

Next steps

Congratulations on successfully completing this lab! Your dedication and effort have paid off, and you're now equipped with the skills and knowledge to tackle the exciting final project of this course. This project will be a culmination of all that you've learned, allowing you to put your newfound expertise into practice.

If you are interested in continuing to learn about Kubernetes and containers, you can get your own [free Kubernetes cluster](#) and your own free [IBM Container Registry](#).

Author(s)

Skills Network

© IBM Corporation. All rights reserved.