OCI DevOps Project: Work with Code Repositories in OCI DevOps Project

Lab 04-1 Practices

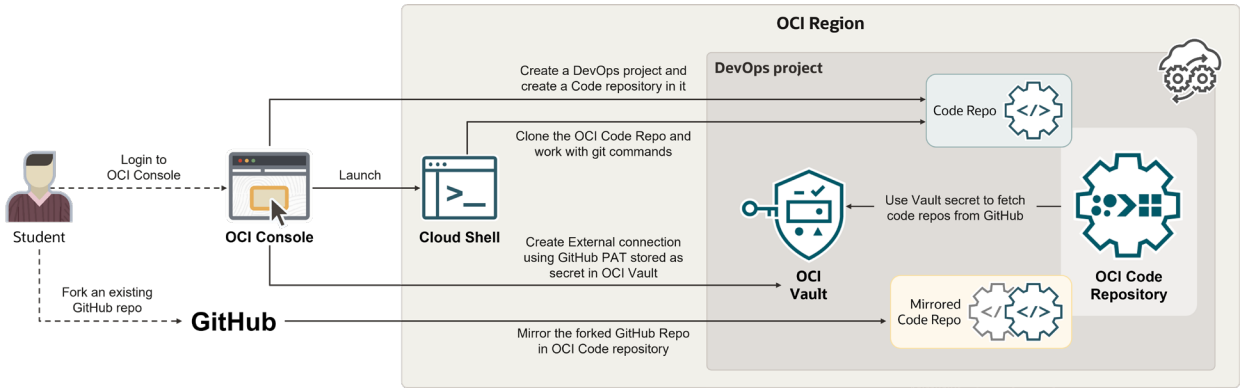
Estimated Time: 45 minutes

Get Started

Overview

There are many ways you can work with Git in the DevOps service. You can use GitHub, GitLab, or Bitbucket or create an OCI Code repository inside your project and upload artifacts.

In this lab, you'll create a sample repository and integrate your GitHub repository with OCI DevOps service. You'll also learn to test and validate your integration.



In this lab, you'll:

a. Create a Personal Access Token in GitHub.

b. Create a Key and Vaults secret in OCI.

c. Create a DevOps project.

d. Create an External Connection.

e. Mirror your GitHub repository.

f. Create an OCI Code Repository in your DevOps project.

g. Clone OCI Code Repository in your Cloud shell session.

h. Perform basic Git operations on the Code Repository.

For more information on Code repositories in OCI DevOps project, see the [OCI Code Repositories Documentation.]

Prerequisites

- You need to have a GitHub account.

Assumptions

- This lab assumes you're working in the Ashburn region. The resource naming convention (`iad`) used in this lab is according to Ashburn.

  If you're working in a different region, change the resource names accordingly. For example, for Phoenix, use `phx`.
- You will replace the *<userID>* placeholder with your Firstname.

Identity and Access Management Policies

Add the following policies

Compartment-level policies

Policies to manage Devops-family

```
Allow group <group_name> to manage devops-project in compartment
<compartment_name>

Allow group <group_name> to manage devops-repository in
compartment <compartment_name>

Allow group <group_name> to manage devops-connection in
compartment <compartment_name>
```

Policy to manage log-group

```
Allow group <group_name> to manage log-groups in compartment
<compartment_name>
```

Policy to create notification topic

```
Allow group <group_name> to manage ons-topics in compartment
<compartment_name>
```

Policy to manage repositories

```
Allow group <group_name> to manage repos in compartment
<compartment_name>
```

Dynamic group

Create dynamic groups for your code repository and connection.

DevOps Repository:

```
ALL {resource.type = 'devopsrepository', resource.compartment.id
= '<compartment OCID>’}
```

DevOps Connection:

```
ALL {resource.type = 'devopsconnection', resource.compartment.id
= '<compartment OCID>'}
```

Dynamic group - Compartment level Policies

```
Allow dynamic-group <Dynamic Group DevOps Connection name> to
use vaults in compartment <compartment_name>

Allow dynamic-group <Dynamic Group DevOps Connection name> to
read secret-family in compartment <compartment_name>

Allow dynamic-group <Dynamic Group DevOps Repository name> to
manage devops-family in compartment <compartment_name>

Allow dynamic-group <Dynamic Group DevOps Repository name> to
read secret-family in compartment <compartment_name>
```

Policy to create keys and secrets within compartment

Compartment level Policy:

```
Allow group <group_name> to manage vaults in compartment
<compartment_name>

Allow group <group_name> to manage keys in compartment
<compartment_name>

Allow group <group_name> to manage secret-family in compartment
<compartment_name>
```

Create a Personal Access Token in GitHub

You'll learn to fork a GitHub repository and create a Personal access token in your GitHub account.

Tasks

1. Sign in to your GitHub account and go to the https://github.com/ou-developers/docker-helloworld-demo repository.

2. In the top-right corner, click **Fork** and then click **Create fork** at the bottom of Create a new fork page.

   Note: By default, forks use the same name as their upstream repository.

3. In your GitHub account, click your profile icon on the top-right corner, and then go to **Settings**.

4. Navigate to **Developer settings** and find **Personal access tokens** > **Token (classic)** on the left menu and then click **Generate new token > Generate new token (classic)** for general use.

5. On the New personal access token (classic) page.

   a. Provide a name as `OCI-DevOps-CLS-LAB` in **Note.**

   b. Set the token **Expiration** to **30 days**.

   c. In the **Select scopes** section, select **repo** (Full control of private repositories) as your scope.

6. Click **Generate token** and make a note of it in a notepad. You'll need this token later when you create secrets. Here's an example how a token would look like:

   ```
   ghp_YnDABCDEPQRxzGZXXXXduoAZgrPemTj1xxXxx
   ```

Create an OCI vault

These steps explain how to create a virtual Vault, which is a lower-cost option than a private vault.

Tasks

1. Sign in to your Oracle Cloud Infrastructure (OCI) account.
2. Open the navigation menu. Click **Identity & Security** and then select **Vault**.
3. Select your Compartment, if not already selected.
4. Click **Create Vault**.
5. Select your Compartment, if not already selected.
6. For Name, enter **OracleDevopsVault**.
7. For the lower-cost option, leave unchecked the option to make it a private vault.
8. Click **Create Vault**.

Create Keys and Vault Secrets

You'll use the vault you created in previous step to create keys and secrets required to connect to an external repository.

Tasks

1. Sign in to your Oracle Cloud Infrastructure (OCI) account.
2. Open the navigation menu. Click **Identity & Security** and then select **Vault**.
3. Select **your** compartment from List Scope on the left menu.
4. From the list of available vaults, click **OracleDevopsVault.**
5. On the vault Details page, Click **Create Key** to create a Master Encryption key.
6. Enter the following values for your key:
   - **Create in Compartment**: Select your *<Compartment Name>*.
   - **Protection Mode**: `HSM`
   - **Name**: `iad-dop-lab04-1-vk-01`
   - Leave everything else to default values and click **Create Key**. It will take about a minute to create the master encryption key. The keys will go through the **Creating** state to the **Active** state.
7. On the Vault details page, **select** your *<Compartment Name>* from List scope on the left menu. You'll see the key "`iad-dop-lab04-1-vk-01`" that you created which is in **Enabled** state.
8. Now, in the **Resources** section on the left menu of the Vault details page, click **Secrets**.
9. Click **Create secret** and enter the following values for your secret:
   - **Compartment**: Select your *<Compartment Name>*.
   - **Name**: `iad-dop-lab04-1-vs-01-`*<userID>*

     For example, `iad-dop-lab04-1-vs-01-mahendra`.
   - **Description**: `Secret to pull GitHub repositories.`
   - **Encryption Key**: `iad-dop-lab04-1-vk-01`
   - **Secret Type Template**: `Plain-Text`

- **Secret Contents**: Add the personal access token you created in your GitHub account and copied into a notepad in the previous task.

  For example, `ghp_YnDABCDEPQRxzGZXXXXduoAZgrPemTj1xxXxx`
- **Click** the **Create Secret** button at the bottom to create the secret. It will take few minutes to create the Vault Secret. The secret will go through the **Creating** state to the **Enabled** state.

Create a DevOps Project

You'll create a topic and DevOps project.

Tasks

1. In the Console, open the navigation menu and click **Developer Services**. Under **Application Integration**, click **Notifications**.
2. Select your *<Compartment Name>* from List scope on the left menu. The page gets updated to display only the resources in that compartment.
3. Click **Topics** under the notification in the **left menu.** You need this topic when you create your DevOps project. This topic will help you to send messages to its subscriptions.
4. Click **Create Topic** at the top of the topic list.
5. In the **Create Topic** page, configure your topic and click **Create**.
   - **Name**: `iad-dop-lab04-1-nt-01-<userID>`. It must be unique across the tenancy; validation is case-sensitive.
   - **Description**: `This topic is for my DevOps lab.`
6. Open the navigation menu and click **Developer Services**. Under **DevOps**, click **Projects**.
7. On the DevOps Projects page, **select** your *<Compartment Name>* from List scope on the left menu.
8. Click **Create devops project**.
   - **Name**: `IAD-DOP-LAB04-1-DP-01-<userID>`
   - **Description**: `This project is for working with OCI DevOps CI/CD.`
   - To set up project notifications, click **Select Topic**.
     - In the Select topic window. Select the option **"Select topic by name"**
     - In the **compartment** field, select your *<Compartment Name>*
     - In the **Topic** field, select the topic that you created earlier `iad-dop-lab04-1-nt-01-<userID>`.

       for example, **`iad-dop-lab04-1-nt-01-mahendra`**. Project notifications keep you informed of important events and the latest project status.
     - Click **Select Topic** at the bottom.
   - Click **Create devops project**.

9. You can use the OCI logging service to record the output it generates when the pipeline runs. This will mean that the build logs are available for use in other tooling. On the page of your newly created project, click **Enable Log** which takes you to the log management page.

   In the **Logs** table, toggle to **enable** the log. This will pop-up to Enable Log window. Leave all the options as default and click **Enable Log** at the bottom. The logs will go through the **Creating** state to the **Active** state. You have successfully created a DevOps project.

Create an External Connection

You'll create a connection to external repositories, such as GitHub.

Tasks

1. Open the navigation menu and click **Developer Services**. Under **DevOps**, click **Projects**.
2. Select the project **IAD-DOP-LAB04-1-DP-01-*<userID>*** and go to **External Connections** on the left menu.
3. **Click** on **Create external connection**. Create an external connection by entering these values.
   - **Name**: `IAD-DOP-LAB04-1-EC-01`
   - **Description**: `Connecting to GitHub.`
   - **Select a type of external connection**: `GitHub`
   - In the Vault Secret section, Under **Vault in `<compartment_name>`** click **Change Compartment** and select the root compartment.
   - Select the **OracleDevopsVault.**Vault from the drop-down list.
   - Under the **Secret in `<Compartment Name>`** field. Select the secret value **iad-dop-lab04-1-vs-01-*<userID>*** within your compartment that contains your Personal access token (PAT) to connect to GitHub.
4. Click **Create**. The connection to the selected external repository is successfully created and active.

   You can now mirror a code repository from GitHub.

Mirror Your GitHub Repository

You'll learn to mirror repositories to and from external sources.

Tasks

1. Navigate to your DevOps project **IAD-DOP-LAB04-1-DP-01-*<userID>*** using the breadcrumb.

2. Click **Code Repositories** on the left menu of your project page.

3. Click **Mirror Repository** to mirror code repository from GitHub. Fill the details as given below:
   - **Connection**: Select `IAD-DOP-LAB04-1-EC-01` from the drop-down list. This is the external connection you created earlier.
   - **Repository**: Select the `docker-helloworld-demo` repository from the drop-down list which you had forked earlier.
   - **Mirroring Schedule**: Select **Custom** from the drop-down list and set the minutes field to 1.
   - **Name**: `IAD-DOP-LAB04-1-MR-01`
   - **Description**: `This is mirroring GitHub repository.`
   - Click **Mirror repository** at the bottom.

   After a while, the mirrored repository will be available in OCI Code Repository.

4. Check if your files are getting updated from your Git Repository.

   a. Sign in to your GitHub account and navigate to the forked repository **docker-helloworld-demo**.

   b. Click **Add File** and select **Create a New File**. This opens a new file.

   c. Give a name to your file, for example, `Mirror_test.txt`.

   Add a line in the file: `This is a test file to check if mirroring is happening in the OCI Code Repository.`

   d. Scroll down the page and click **Commit New File**.

   e. Switch to the OCI Console and go to your Mirrored Code Repository (IAD-DOP-LAB04-1-MR-01). You'll see a message "**Mirroring is in Progress**" at the top of the page.

f. Click **Files** in the left menu. After one minute, scan through the files and check if `Mirror_test.txt` is present in that branch.

5. Clean up your mirrored repo.

   a. Click **Code Repositories** on the left menu of your project page and locate your mirrored repository **IAD-DOP-LAB04-1-MR-01**.

   b. Click the three dots on the right to open the Actions menu. Select **Delete**.

   c. Type the repository name in the provided field to confirm the Delete action and then click **Delete**.

Create an OCI Code Repository in Your DevOps Project

You'll learn to create a code repository inside your DevOps project, which is very similar to your Git repository.

Tasks

1. Navigate to your DevOps project **IAD-DOP-LAB04-1-DP-01-<*userID*>.**
2. Click **Code Repositories** on the left menu of your project page.
3. Click **Create Repository**. Enter the following details:
   - **Repository name**: `IAD-DOP-LAB04-1-CR-01`
   - **Description**: `This code repository will be cloned with Git.`
   - **Default branch**: `main`
4. Click **Create Repository**. An empty code repository is created with the main branch.

You can perform the following actions on the repository: access your files, access all the commits pertaining to the code repository you just created, compare file changes, branch actions such as GitHub, view Git tags, and monitor the status of all the operations.

Clone OCI Code Repository in Your Cloud Shell Session

You'll clone the code repository to create a local copy on your cloud shell session, add or remove files, commit changes, and work on different branches by using Git operations. You can use two methods to clone: HTTPS and SSH keys. In this lab, you'll use HTTPS.

Tasks

1. Navigate to your DevOps project **IAD-DOP-LAB04-1-DP-01-<*userID*>.**
2. Click **Code Repositories** on the left menu of your project page.
3. Click **IAD-DOP-LAB04-1-CR-01** and click **Clone** in the Code Repository details page.
4. In the **Clone** window, to the right of the **Clone with HTTPS** field, click **Copy** to get the path to access the repository using Git. Save this information in a notepad.
5. Open **Cloud Shell**. In the Cloud Shell, navigate to the home directory and copy-paste the URL to clone the public repository.

    a. Go to home directory.

    ```
    $ cd ~
    ```

    b. Clone by copy-pasting the URL.

    ```
    $ git clone <paste the HTTPS URL copied in the Clone page.>
    ```

    Sample code:

    ```
    $ git clone https://devops.scmservice.us-ashburn-
    1.oci.oraclecloud.com/namespaces/oracletenancy/projects/IAD-DOP-
    LAB04-1-DP-01-<userID>/repositories/IAD-DOP-LAB04-1-CR-01
    ```

    c. You must provide your username: `<tenancy-namespace>/<username>`. For example, `oracletenancy/ mahendra@acme.com`.

    d. Your password is your auth token. When you enter or paste the password, you'll not see masked characters. Press Enter on your keyboard to continue.

    Note: You need an Auth Token to clone the repository using HTTPS. Use the auth token created in the earlier lab (IAD-DOP-LAB02-1-AT-1), that is saved in your notepad. If you don't have it, then create a new one by referring to the lab *Microservices and Container Orchestration: Create and work with OCIR repository* (Lab02-1).

6. Switch to your recently cloned directory and you'll see that there are no files.

```
$ cd ~/IAD-DOP-LAB04-1-CR-01
$ ls
```

7. You can now add the files from your existing `docker-helloworld-demo` directory to the **IAD-DOP-LAB04-1-CR-01** directory you just cloned.

```
$ cd ~/docker-helloworld-demo
$ cp * ~/IAD-DOP-LAB04-1-CR-01
```

8. Navigate to the cloned directory (IAD-DOP-LAB04-1-CR-01) in Cloud Shell. You should see all the files copied.

```
$ cd ~/IAD-DOP-LAB04-1-CR-01
$ ls
```

9. Now check the current configuration of Git in your IAD-DOP-LAB04-1-CR-01 directory with the following command:

```
$ git remote -v
```

   Check if the configuration for the remote repository is pointing to your OCI Code Repository. For example,

```
origin  https://devops.scmservice.us-ashburn-
1.oci.oraclecloud.com/namespaces/oracletenancy/projects/IAD-DOP-
LAB04-1-DP-01-<userID>/repositories/IAD-DOP-LAB04-1-CR-01
(fetch)
origin  https://devops.scmservice.us-ashburn-
1.oci.oraclecloud.com/namespaces/oracletenancy/projects/IAD-DOP-
LAB04-1-DP-01-<userID>/repositories/IAD-DOP-LAB04-1-CR-01 (push)
```

10. Every time you make changes to your files and save it, it will not automatically update the OCI Code Repository (IAD-DOP-LAB04-1-CR-01) within the DevOps Project (IAD-DOP-LAB04-1-DP-01-*<userID>*). All the changes you made in the file are updated only in your local repository. To update the changes to the main branch in OCI Code Repository within the DevOps Project run the following commands:

```
$ git add .
$ git config --global user.email "enter you email"
$ git config --global user.name "Your Name"
$ git commit -m "first push into OCI Code Repository"
$ git push -u -f origin main
```

- When it prompts for your username: `<tenancy-namespace>/<username>`. Replace the `<tenancy-namespace>` and `<username>` values from the info given in the Profile menu.

  For example, `oracletenancy/mahendra@acme.com`.

  Your password is the auth token, this is token created in the earlier lab (IAD-DOP-LAB02-1-AT-01), that you saved in your notepad earlier.

11. In the OCI Console, go to your DevOps project and then to the IAD-DOP-LAB04-1-CR-01 code repository you created. Click **Files** in the left menu and notice all the files are available in the code repository.

    The initial push of all your code for a sample Web Application has taken place into your OCI Code Repository. As you do further practices, you will make changes to the files in the local repository in the Cloud Shell and push it into your OCI Code Repository.

Perform Basic Git Operations on the Code Repository

Learn to run some basic Git operations.

Tasks

1. In the Cloud Shell, go to the `IAD-DOP-LAB04-1-CR-01` directory.
   ```
   $ cd ~/IAD-DOP-LAB04-1-CR-01
   ```
2. Create a new branch in the local repository.
   ```
   $ git branch new_branch
   ```
3. Move to the newly created branch.
   ```
   $ git checkout new_branch
   ```
4. Create a sample file in the new branch.
   ```
   $ echo "OCI_GIT_TEST" >> test1.txt
   ```
5. Use the `ls` command to verify the new file is now present in the directory.
   ```
   $ ls
   ```
   The `test1.txt` file must be present in the directory.
6. Now add the file to the git repository for commit.
   ```
   $ git add test1.txt
   ```
   Adds the file `test1.txt` in the local repository and stages them for commit.
7. Before you commit, check what files are staged.
   ```
   $ git status
   ```
   Lists all new or modified files to be committed.
8. Commit the changes you made to your Git Repository.
   ```
   $ git commit -m "second commit- added file test1.txt in
   new_branch"
   ```
9. Push the newly created branch to OCI Code Repository
   ```
   $ git push -u origin new_branch
   ```

- When it prompts for your username: `<tenancy-namespace>/<username>`. Replace the `<tenancy-namespace>` and `<username>` values from the info given in the Profile menu.

  For example, `oracletenancy/mahendra@acme.com`.

  Your password is the auth token, this is token created in the earlier lab (IAD-DOP-LAB02-1-AT-01), that you saved in your notepad earlier.

10. In the Console, navigate to the code repository **IAD-DOP-LAB04-1-CR-01** within your Devops project.

11. Select **Files** in the left menu and **click** the drop-down list to select a branch. You should see **new_branch**. Select the newly created branch and scan through the files and check if `test1.txt` is present in that branch.

    Important Note: Do not delete any artifacts and resources created in this lab because they will be required in the upcoming labs.

Congratulations! in this lab, you've learned to create a project, mirror a repository, and clone the code repository to create a local copy.