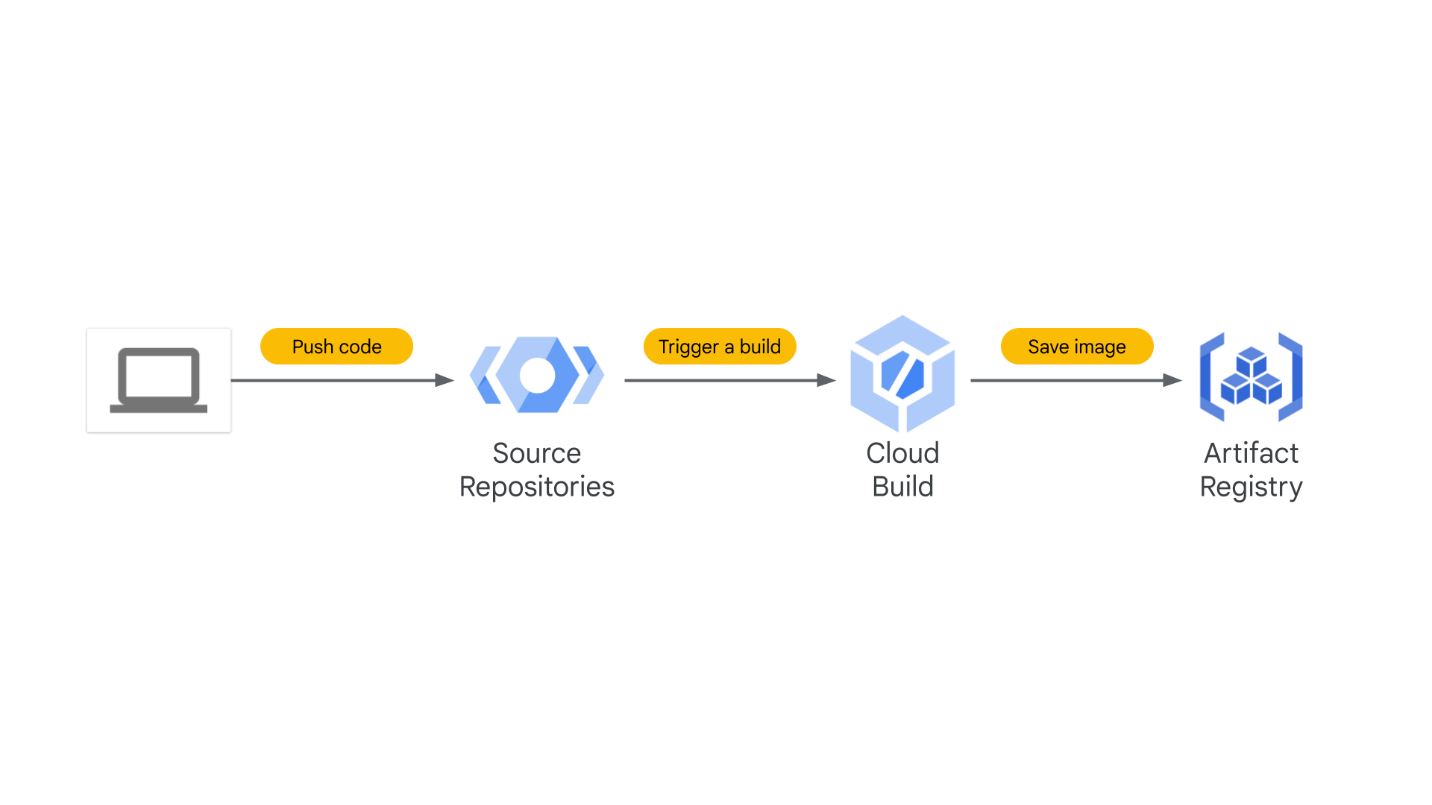
**Building a DevOps Pipeline**

experimentLabschedule2 hoursuniversal\_currency\_alt5 Creditsshow\_chartIntroductory

infoThis lab may incorporate AI tools to support your learning.

**Overview**

In this lab, you will build a continuous integration pipeline using Cloud Source Repositories, Cloud Build, build triggers, and Artifact Registry.



**Objectives**

In this lab, you will learn how to perform the following tasks:

* Create a Git repository
* Create a simple Python application
* Test Your web application in Cloud Shell
* Define a Docker build
* Manage Docker images with Cloud Build and Artifact Registry
* Automate builds with triggers
* Test your build changes

**Set up your lab environment**

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 1:15:00), and make sure you can finish within that time.  
   There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.  
   If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

**Note:** Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

**Task 1. Create a Git repository**

First, you will create a Git repository using the Cloud Source Repositories service in Google Cloud. This Git repository will be used to store your source code. Eventually, you will create a build trigger that starts a continuous integration pipeline when code is pushed to it.

1. On the Google Cloud console title bar, type **Source Repositories** in the **Search** field, then click **Source Repositories** in the **Products & Pages** section.
2. Click **Add repository**.
3. Select **Create new repository** and click **Continue**.
4. Name the repository **devops-repo**.
5. Select your current project ID from the list.
6. Click **Create**.
7. Click **Cloud Console**, and in the new tab click **Activate Cloud Shell** (Cloud Shell icon).
8. If prompted, click **Continue**.
9. Enter the following command in Cloud Shell to create a folder called gcp-course:

mkdir gcp-course

Copied!

content\_copy

1. Change to the folder you just created:

cd gcp-course

Copied!

content\_copy

1. Now clone the empty repository you just created. If prompted, click **Authorize**:

gcloud source repos clone devops-repo

Copied!

content\_copy

**Note:**You may see a warning that you have cloned an empty repository. That is expected at this point.

1. The previous command created an empty folder called devops-repo. Change to that folder:

cd devops-repo

Copied!

content\_copy

Click *Check my progress* to verify the objective.

Create a git repository.

Check my progress

**Task 2. Create a simple Python application**

You need some source code to manage. So, you will create a simple Python Flask web application. The application will be only slightly better than "hello world", but it will be good enough to test the pipeline you will build.

1. In Cloud Shell, click **Open Editor** (Editor icon) to open the code editor. If prompted click **Open in a new window**.
2. Select the **gcp-course > devops-repo** folder in the explorer tree on the left.
3. Click on **devops-repo**.
4. Click **New File**.
5. Name the file main.py and press **Enter**.
6. Paste the following code into the file you just created:

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route("/")

def main():

model = {"title": "Hello DevOps Fans."}

return render\_template('index.html', model=model)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=8080, debug=True, threaded=True)

Copied!

content\_copy

1. To save your changes. Press **CTRL + S**.
2. Click on the devops-repo folder.
3. Click **New Folder**.
4. Name the folder templates and press **Enter**.
5. Right-click on the templates folder and create a new file called layout.html.
6. Add the following code and save the file as you did before:

<!doctype html>

<html lang="en">

<head>

<title>{{model.title}}</title>

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">

</head>

<body>

<div class="container">

{% block content %}{% endblock %}

<footer></footer>

</div>

</body>

</html>

Copied!

content\_copy

1. Also in the templates folder, add another new file called index.html.
2. Add the following code and save the file as you did before:

{% extends "layout.html" %}

{% block content %}

<div class="jumbotron">

<div class="container">

<h1>{{model.title}}</h1>

</div>

</div>

{% endblock %}

Copied!

content\_copy

1. In Python, application prerequisites are managed using pip. Now you will add a file that lists the requirements for this application.
2. In the **devops-repo** folder (*not the templates folder*), create a **New File** and add the following to that file and save it as requirements.txt:

Flask>=2.0.3

Copied!

content\_copy

1. You have some files now, so save them to the repository. First, you need to add all the files you created to your local Git repo. Click **Open Terminal** and in Cloud Shell, enter the following code:

cd ~/gcp-course/devops-repo

git add --all

Copied!

content\_copy

1. To commit changes to the repository, you have to identify yourself. Enter the following commands, but with your information (*you can just use your Gmail address or any other email address*):

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

Copied!

content\_copy

1. Now, commit the changes locally:

git commit -a -m "Initial Commit"

Copied!

content\_copy

1. You committed the changes locally, but have not updated the Git repository you created in Cloud Source Repositories. Enter the following command to push your changes to the cloud:

git push origin master

Copied!

content\_copy

1. Refresh the **Source Repositories** web page. You should see the files you just created.

**Task 3. Define a Docker build**

The first step to using Docker is to create a file called **Dockerfile**. This file defines how a Docker container is constructed. You will do that now.

1. Click **Open Editor**, and expand the **gcp-course/devops-repo** folder. With the **devops-repo** folder selected, click **New File** and name the new file **Dockerfile**.

The file *Dockerfile* is used to define how the container is built.

1. At the top of the file, enter the following:

FROM python:3.9

Copied!

content\_copy

This is the base image. You could choose many base images. In this case, you are using one with Python already installed on it.

1. Enter the following:

WORKDIR /app

COPY . .

Copied!

content\_copy

These lines copy the source code from the current folder into the /app folder in the container image.

1. Enter the following:

RUN pip install gunicorn

RUN pip install -r requirements.txt

Copied!

content\_copy

This uses pip to install the requirements of the Python application into the container. Gunicorn is a Python web server that will be used to run the web app.

1. Enter the following:

ENV PORT=80

CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 main:app

Copied!

content\_copy

The environment variable sets the port that the application will run on (in this case, 80). The last line runs the web app using the gunicorn web server.

1. Verify that the completed file looks as follows and save it:

FROM python:3.9

WORKDIR /app

COPY . .

RUN pip install gunicorn

RUN pip install -r requirements.txt

ENV PORT=80

CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 main:app

Copied!

content\_copy

**Task 4. Manage Docker images with Cloud Build and Artifact Registry**

The Docker image has to be built and then stored somewhere. You will use **Cloud Build** and **Artifact Registry**.

1. Click **Open Terminal** to return to Cloud Shell. Make sure you are in the right folder:

cd ~/gcp-course/devops-repo

Copied!

content\_copy

1. The Cloud Shell environment variable DEVSHELL\_PROJECT\_ID automatically has your current project ID stored. The project ID is required to store images in Artifact Registry. Enter the following command to view your project ID:

echo $DEVSHELL\_PROJECT\_ID

Copied!

content\_copy

1. Enter the following command to create an Artifact Registry repository named devops-repo:

gcloud artifacts repositories create devops-repo \

--repository-format=docker \

--location=us-east1

Copied!

content\_copy

1. To configure Docker to authenticate to the Artifact Registry Docker repository, enter the following command:

gcloud auth configure-docker us-east1-docker.pkg.dev

Copied!

content\_copy

1. To use Cloud Build to create the image and store it in Artifact Registry, type the following command:

gcloud builds submit --tag us-east1-docker.pkg.dev/$DEVSHELL\_PROJECT\_ID/devops-repo/devops-image:v0.1 .

Copied!

content\_copy

*Notice the environment variable in the command*. The image will be stored in Artifact Registry.

1. On the Google Cloud console title bar, type **Artifact Registry** in the **Search** field, then click **Artifact Registry** in the **Products & Pages** section.
2. Click **devops-repo**.
3. Click **devops-image**. Your image should be listed.
4. On the Google Cloud console title bar, type **Cloud Build** in the **Search** field, then click **Cloud Build** in the **Products & Pages** section. Your build should be listed in the history.

You will now try running this image from a Compute Engine virtual machine.

1. On the **Navigation menu**, click **Compute Engine > VM Instance**.
2. Click **Create Instance** to create a VM.
3. On the **Create an instance** page, specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value** |
| **Container** | Click DEPLOY CONTAINER |
| **Container image** | 'us-east1-docker.pkg.dev/qwiklabs-gcp-03-8b27d6e61947/devops-repo/devops-image:v0.1` and click SELECT |
| **Firewall** | Allow HTTP traffic |

1. Click **Create**.
2. Once the VM starts, click the VM's external IP address. A browser tab opens and the page displays Hello DevOps Fans.

**Note:**You might have to wait a minute or so after the VM is created for the Docker container to start.

1. You will now save your changes to your Git repository. In Cloud Shell, enter the following to make sure you are in the right folder and add your new Dockerfile to Git:

cd ~/gcp-course/devops-repo

git add --all

Copied!

content\_copy

1. Commit your changes locally:

git commit -am "Added Docker Support"

Copied!

content\_copy

1. Push your changes to Cloud Source Repositories:

git push origin master

Copied!

content\_copy

1. Return to Cloud Source Repositories and verify that your changes were added to source control.

Click *Check my progress* to verify the objective.

Assessment Completed!

Manage Docker images with Cloud Build and Artifact Registry.

Check my progress

*Assessment Completed!*

**Task 5. Automate builds with triggers**

1. On the **Navigation menu**, click **Cloud Build**. The **Build history** page should open, and one or more builds should be in your history.
2. Click the **Triggers** link on the left.
3. Click **Create trigger** and specify the following:

|  |  |
| --- | --- |
| **Property** | **Value** |
| **Name** | devops-trigger |
| **Region** | us-east1 |
| **Repository** | devops-repo(Cloud Source Repositories) |
| **Branch** | .\*(any branch) |
| **Configuration Type** | Cloud Build configuration file (yaml or json) |
| **Location** | Inline |

1. Click **Open Editor** and replace the code with the code metioned below and click **Done**.

steps:

- name: 'gcr.io/cloud-builders/docker'

args: ['build', '-t', 'us-east1-docker.pkg.dev/qwiklabs-gcp-03-8b27d6e61947/devops-repo/devops-image:$COMMIT\_SHA', '.']

images:

- 'us-east1-docker.pkg.dev/qwiklabs-gcp-03-8b27d6e61947/devops-repo/devops-image:$COMMIT\_SHA'

options:

logging: CLOUD\_LOGGING\_ONLY

Copied!

content\_copy

1. For **Service account** select the service account starting with your project-id that look similar to (qwiklabs-gcp-03-8b27d6e61947@qwiklabs-gcp-03-8b27d6e61947.iam.gserviceaccount.com) and and click **Create**.
2. To test the trigger, click **Run** and then **Run trigger**.
3. Click the **History** link and you should see a build running. Wait for the build to finish, and then click the link to it to see its details.
4. Scroll down and look at the logs. The output of the build here is what you would have seen if you were running it on your machine.
5. Return to the Artifact Registry service. You should see a new image in the **devops-repo** > **devops-image** folder.
6. Return to the **Cloud Shell Code Editor**. Find the file main.py in the gcp-course/devops-repo folder.
7. In the main() function, change the title property to "Hello Build Trigger." as shown below:

@app.route("/")

def main():

model = {"title": "Hello Build Trigger."}

return render\_template("index.html", model=model)

1. Commit the change with the following command:

cd ~/gcp-course/devops-repo

git commit -a -m "Testing Build Trigger"

Copied!

content\_copy

1. Enter the following to push your changes to Cloud Source Repositories:

git push origin master

Copied!

content\_copy

1. Return to the Cloud Console and the **Cloud Build** service. You should see another build running.

Click *Check my progress* to verify the objective.

Assessment Completed!

Automate Builds with Trigger.

Check my progress

*Assessment Completed!*

**Task 6. Test your build changes**

1. When the build completes, click on it to see its details.
2. Click **Execution Details**,
3. Click the **Image name**. This redirects you to the image page in Artifact Registry.
4. At the top of the pane, click **copy** next to the image name. You will need this for the next steps. The format will look as follows.

'us-east1-docker.pkg.dev/qwiklabs-gcp-03-8b27d6e61947/devops-demo/devops-image@sha256:8aede81a8b6ba1a90d4d808f509d05ddbb1cee60a50ebcf0cee46e1df9a54810`

**Note:**Do not use the image name located in Digest.

1. Go to the **Compute Engine** service. As you did earlier, create a new virtual machine to test this image. Click **DEPLOY CONTAINER** and paste the image you just copied.
2. Select **Allow HTTP traffic**.
3. When the machine is created, test your change by making a request to the VM's external IP address in your browser. Your new message should be displayed.

**Note:**You might have to wait a few minutes after the VM is created for the Docker container to start.

Click *Check my progress* to verify the objective.

Assessment Completed!

Test your Build Changes.

Check my progress

*Assessment Completed!*

**Congratulations!**

In this lab, you built a continuous integration pipeline using the Google Cloud tools Cloud Source Repositories, Cloud Build, build triggers, and Artifact Registry.

**End your lab**

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you’ve used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

* 1 star = Very dissatisfied
* 2 stars = Dissatisfied
* 3 stars = Neutral
* 4 stars = Satisfied
* 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Copyright 2022 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.