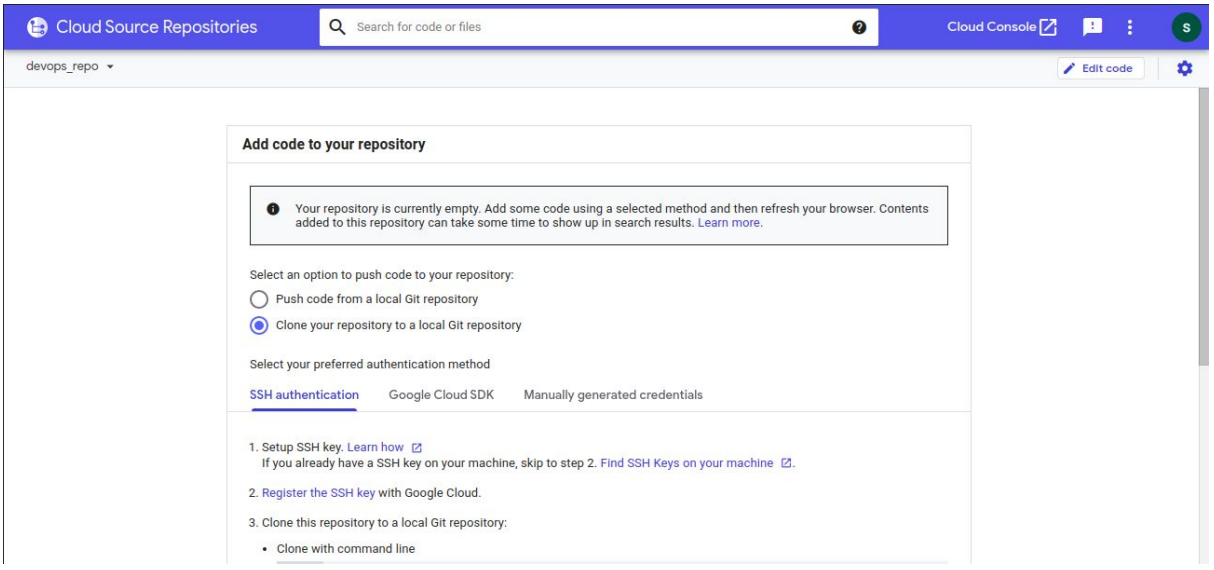
Building a Devops pipeline

Lab Objectives

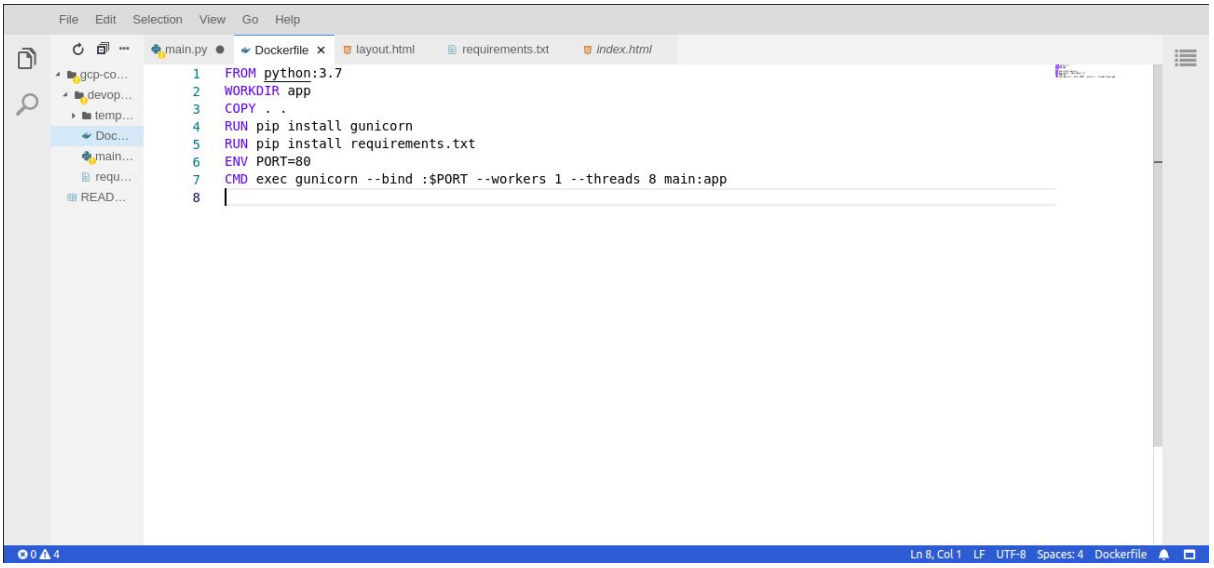
- Create a Git repository
- Create a simple python application
- Test the web application in cloud shell
- Define a Docker build
- Manage Docker images with Cloud Build and Container registry
- Automate Builds with triggers
- Test our build changes

We started the lab by creating a Git repository that we cloned in our project. This repository was created in **Cloud Sources repositories.** Like Github, Gitlab or Bitbucket, the Cloud Sources Repositories is a git repositories server designed and owned by Google. It is integrated to GCP. Our repository was named *devops-repos* like shown on the screenshot below:



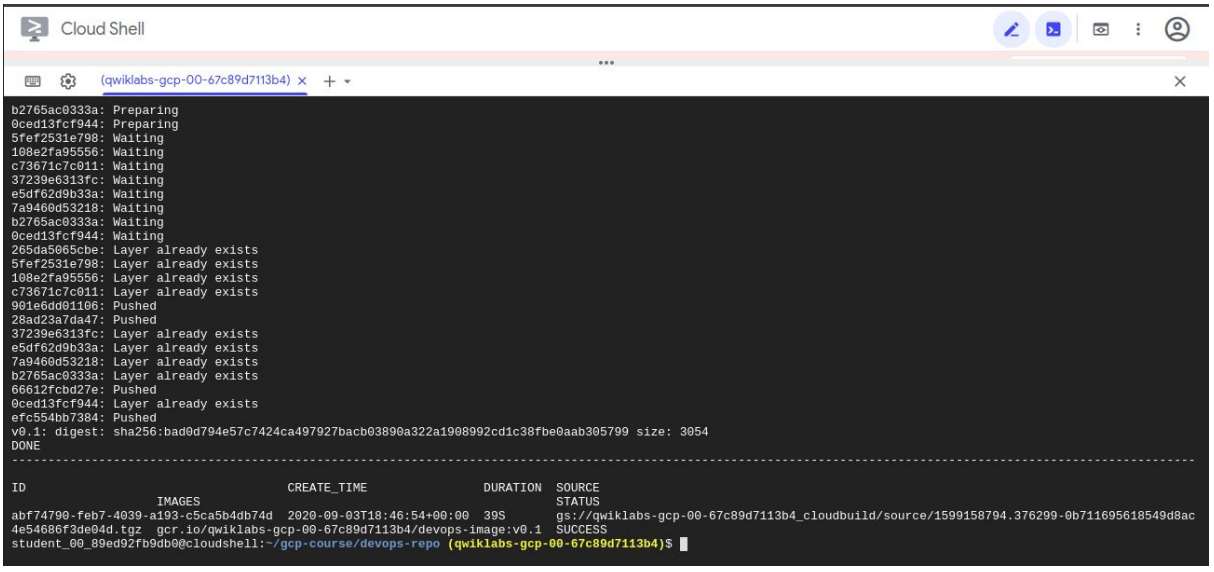
[Our repository]

We then created a simple python application. At this step, we just copied and pasted existing code in files that we saved and pushed in our repository. We then tested our web application in the cloud shell.
After the test, we made a little modification in the web application source code and pushed the code again. We then created a Dockerfile to containerize our web application.



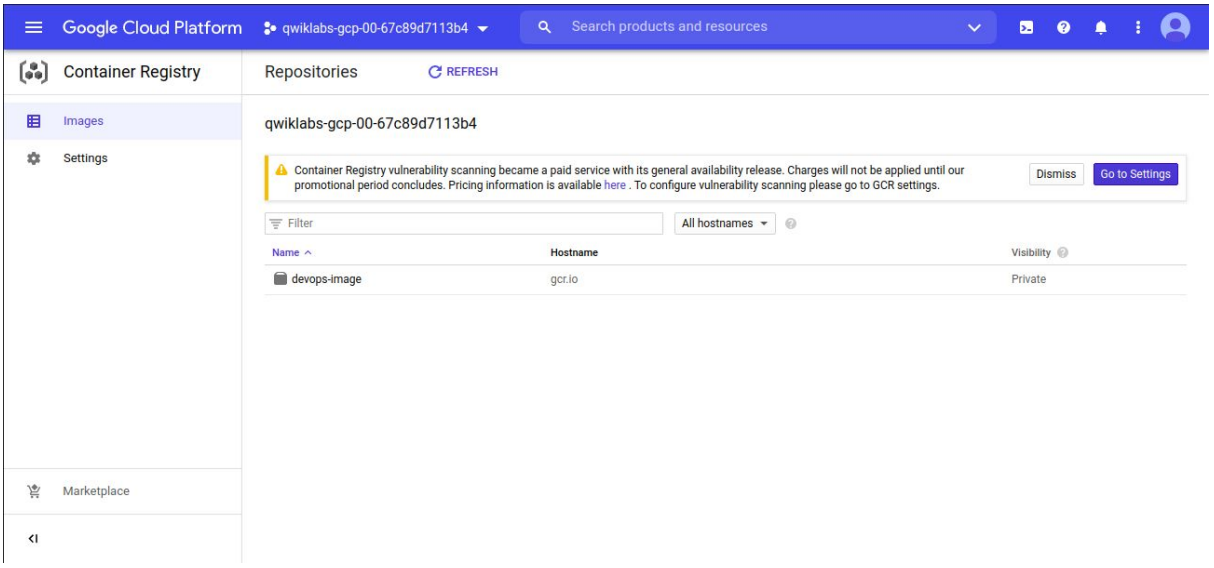
[Dockerfile content]

After this, we used Cloud Build to build our container image.



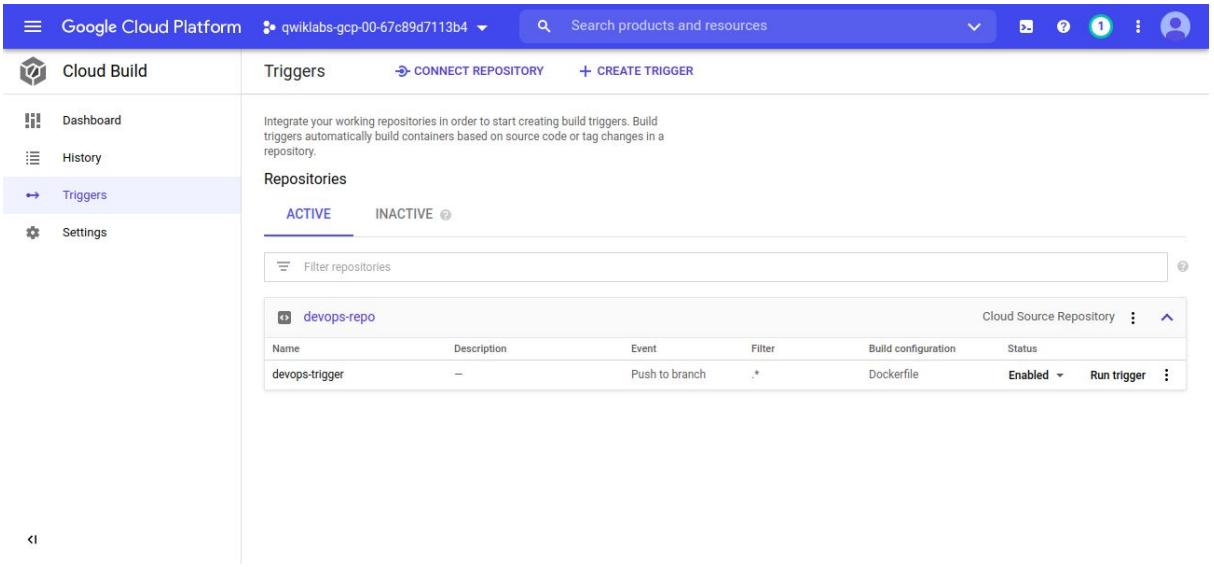
[Building image]

The image has been saved in the **Container Registry.** Container Registry is something like Docker hub. The one big difference between them is that Container Registry is only present on GCP while Docker Hub can be accessed by anyone.



[Our container Registry Content]

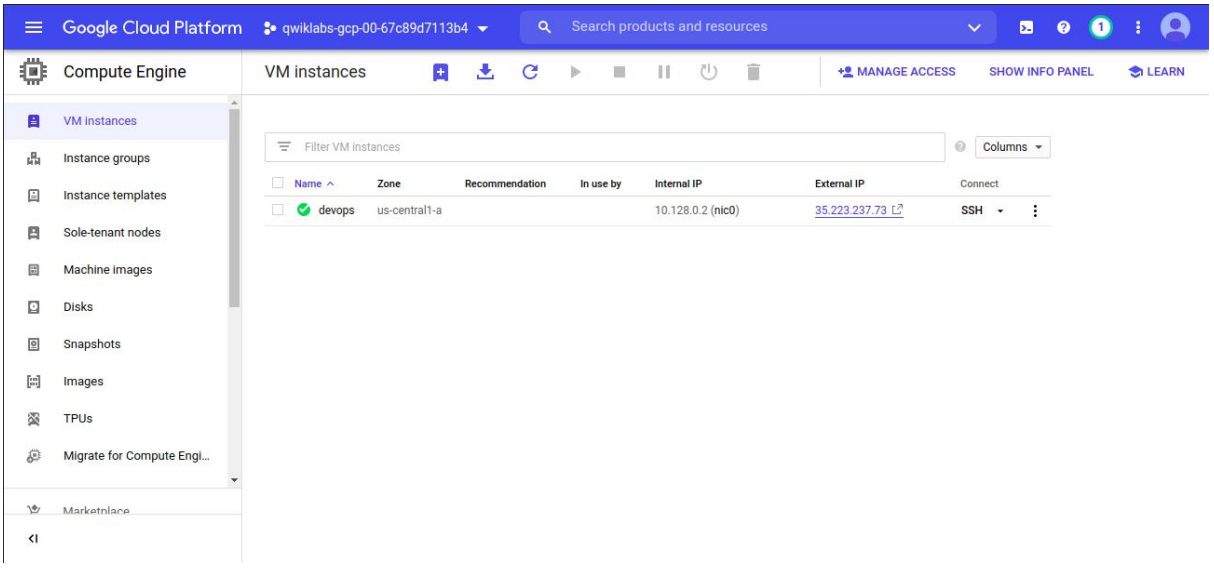
One of the features offered by the GCP is the automation of builds. The builds automation is assumed by triggers. In order to automate our builds, we created a trigger like shown below.



[The trigger we created]

With this trigger, as soon as we push a new code in our source repository, the Cloud Build automatically builds a new image for our container.

Finally, we created a Virtual machine in which we deploy our container, for testing.



[Vm instance created for test]