

Evaluation DevOps

The goal of the evaluation is to setup 3 Ubuntu virtual machines, the first one is a gitlab-instance, the second is a gitlab-runner and the third one is used to deploy the application on docker.

Table des matières

| | |
|-------------------------------|---|
| Deploy virtual machines | 1 |
| Configure Gitlab CI/CD..... | 2 |
| Configure docker server | 2 |
| Deploy the application | 3 |

Deploy virtual machines

The first step is to install 3 virtual machines with Ubuntu server without forgetting to install OpenSSH for an easy access to the machines and it will be used later to deploy the application.

On the first machine, we installed a gitlab-instance to have our gitlab in local.

On the second machine, we installed a gitlab-runner to run our pipeline jobs. On Gitlab we created a repository named « counter-app » and a runner which is assigned to our repository.

On the third machine, we installed docker to later on deploy our application.

#2 (KZYj9QHHY)

Online Project Created 1 month ago

| Details | Jobs 141 |
|---------------------|--------------------|
| Description | runner |
| Last contact | 18 minutes ago |
| Configuration | Runs untagged jobs |
| Maximum job timeout | None |
| Token expiry ? | Never expires |
| Tags | None |
| Runners ? | 1 > Show details |

Assigned Projects (3)


























| |
|------------------------------------|
| Filter projects |
| N Administrator / NodeJS Owner |
| B Administrator / bootstrap-devops |
| C Administrator / counter-app |

Configure Gitlab CI/CD

Now that we have both a gitlab-instance and a gitlab-runner, we need to add some CI/CD.

The first step is to create a `.gitlab-ci.yml` file to create our pipeline, this file can be found [here](#) on our Github repository.

This file used some CI/CD variables that we need to add to our Gitlab CI/CD in the CI/CD settings like you can see below :

| CI/CD Variables </> 5 | | | | Reveal values | Add variable |
|---|---|---|---|---|--------------|
| ↑ Key | Value | Environments | Actions | | |
| DOCKER_PASSWORD  Expanded | *****  | All (default)  |  |  | |
| DOCKER_USERNAME  Expanded | *****  | All (default)  |  |  | |
| SSH_PRIVATE_KEY  Expanded | *****  | All (default)  |  |  | |
| SSH_USER  Expanded | *****  | All (default)  |  |  | |
| VM_IPADDRESS  Expanded | *****  | All (default)  |  |  | |

Configure docker server

Now that our gitlab is configured correctly and docker is installed on our docker server. We need at first to give root rights to our machine user so our pipeline jobs will be able to execute docker commands. For this we will modify the file in the folder « etc » `sudoers` by adding at the end of the file this line for my user « axel » :

```
axel    ALL=(ALL) NOPASSWD: ALL
```

To allow our pipeline to connect to our remote machine by ssh, we need to generate SSH private & public keys. For this we will run the command « `ssh-keygen -t ed25519 -C "GitLab SSH key"` », that will generate 2 files in the folder « `.ssh` » :

- `id_ed25519` -> which contains the private key
- `id_ed25519.pub` -> which contains the public key

We will add the SSH private key as a CI/CD variable on Gitlab and add the SSH public key in the file « `authorization_keys` » on our docker server to connect remotely easily.

Deploy the application

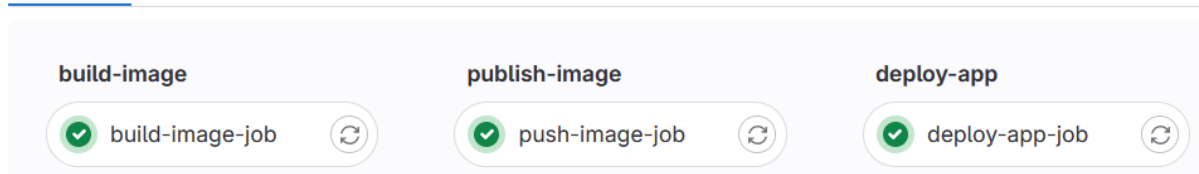
Now that everything is configured correctly, our pipeline will run without any errors like this :

✓ Passed Administrator created pipeline for commit f497a7c3 finished 1 minute ago

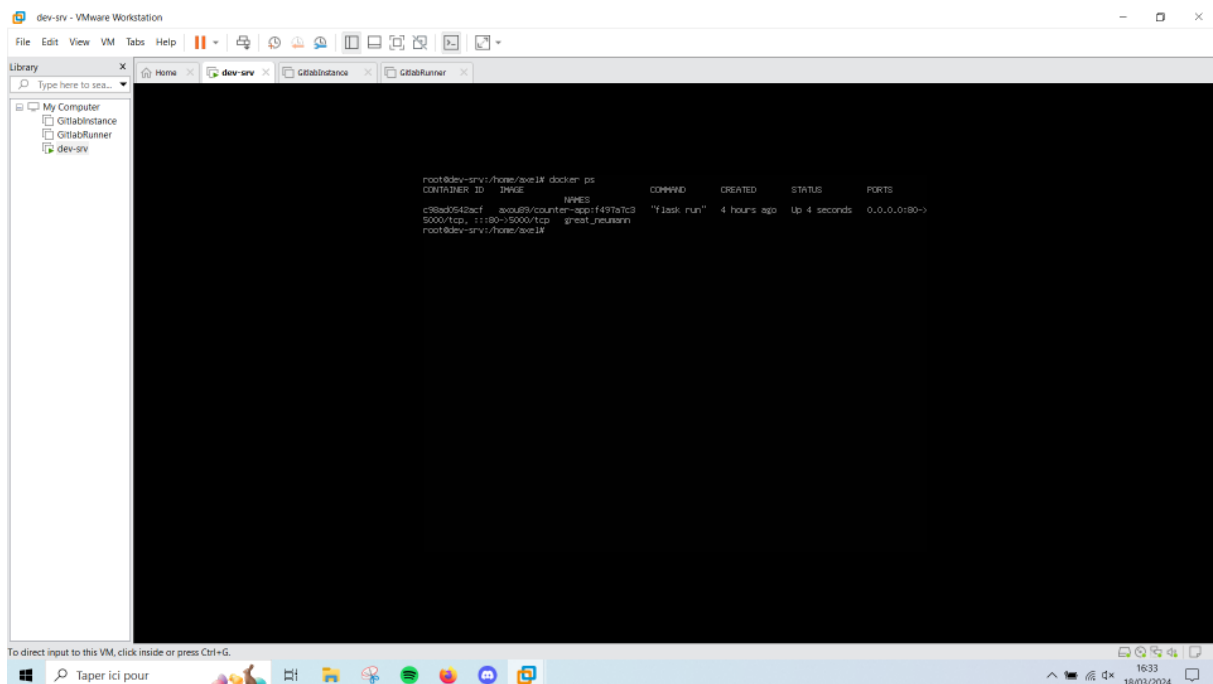
For main

latest 3 Jobs 1 minute 49 seconds, queued for 1 seconds

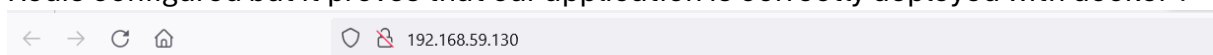
Pipeline Needs Jobs 3 Tests 0



Our pipeline will build the image, push it on docker hub then create a container on our docker server to run it :



Now if we try to reach our application we will have an error 500 because we don't have Redis configured but it proves that our application is correctly deployed with docker :



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

This evaluation was made by Thomas LEMAITRE and Axel SENECA.