

## ICT & Infra S3 Automation & Orchestration, week 12

Class:	I3-CB01
Student number:	4961854
Student name:	Heiko Morales

### Introduction

This week you will create CI/CD pipeline in GitLab.

This is **individual** assignment.

### Assignment 1. Create Continuous Integration (CI) for your own project.

Difficulty: ★★★★★

In Semester 2 you made a Flask app for your own project. For this assignment, we ask you to create Continuous Integration process, which can be verified by an automated build and automated tests (which you must create). The tests must be testing the actual application.

Beware: Creating runner in EC2 can be expensive. Better create it in Fontys InfraLab or ECS.

**Provide screenshots (evidence) for your solution. Always explain your evidence! As a prof, we expect at least:**

- Provide CI process code defined in .gitlab-ci.yml
- Provide the tests you implemented in your project
- Provide multiple screenshots of successfully executed (passed) jobs (from git.fhict.nl -> CI.CD -> Jobs -> passed)

### Assignment 2. Implement Continuous Deployment, which will complete CI/CD pipeline

Difficulty: ★★★★★☆

To make your app accessible online, you must deploy it to a web server. Complete CI/CD pipeline, so it can deploy your app, if automated build and automated tests succeed.




**Provide screenshots (evidence) for your solution. Always explain your evidence! As a prof, we expect at least:**



- Provide complete process defined in .gitlab-ci.yml
- Provide multiple screenshots of successfully executed (passed) jobs (from git.fhict.nl -> CI.CD -> Jobs -> passed)

*Solution:*

***It should be noted that being an Erasmus student and not having access to the flask application created in previous years, I have used an application that I already had from previous years at my previous university.***

*To do this task we will first create the Project tests as follows:*

 .gitlab-ci.yml  test\_api.py  test\_views.py X

src > app > tests >  test\_views.py >  test\_home

```
1 def test_home(client):
2     resp = client.get("/")
3
4     assert resp.status_code == 200
5     assert b"Python" in resp.data
6
7
8 def test_page_content(client):
9     resp = client.get("/")
10
11     assert resp.status_code == 200
12     assert b"Coleman" in resp.data
13
14
15 def test_info(client):
16     resp = client.get("/info")
17
18     assert resp.status_code == 200
19     assert b"Hostname" in resp.data
20
```

```
import json

# Test the monitor API returns JSON results we expect
def test_api_monitor(client):
    resp = client.get("/api/monitor")

    assert resp.status_code == 200
    assert resp.headers["Content-Type"] == "application/json"
    resp_payload = json.loads(resp.data)
    assert resp_payload["cpu"] >= 0
    assert resp_payload["disk"] >= 0
    assert resp_payload["disk_read"] >= 0
    assert resp_payload["disk_write"] >= 0
    assert resp_payload["mem"] >= 0
    assert resp_payload["net_recv"] >= 0
    assert resp_payload["net_sent"] >= 0
```






we will define the `.gitlab-ci.yml` as in the following image.

```
.gitlab-ci.yml X
.gitlab-ci.yml
1  variables:
2    IMAGE_NAME: nanajanashia/demo-app
3    IMAGE_TAG: python-app-1.0
4
5  stages:
6    - test
7    - build
8    - deploy
9
10 run_tests:
11   stage: test
12   image: python:3.9-slim-buster
13   before_script:
14     - apt-get update && apt-get install make
15   script:
16     - make test
17
18
19 build_image:
20   stage: build
21   image: docker:20.10.16
22   services:
23     - docker:20.10.16-dind
24   variables:
25     DOCKER_TLS_CERTDIR: "/certs"
26   before_script:
27     - docker login -u $REGISTRY_USER -p $REGISTRY_PASS
28   script:
29     - docker build -t $IMAGE_NAME:$IMAGE_TAG .
30     - docker push $IMAGE_NAME:$IMAGE_TAG
31
32
33 deploy:
34   stage: deploy
35   before_script:
36     - chmod 400 $SSH_KEY
37   script:
38     - ssh -o StrictHostKeyChecking=no -i $SSH_KEY root@161.35.223.117 "
39       docker login -u $REGISTRY_USER -p $REGISTRY_PASS &&
40       docker ps -aq | xargs docker stop | xargs docker rm &&
41       docker run -d -p 5000:5000 $IMAGE_NAME:$IMAGE_TAG"
```

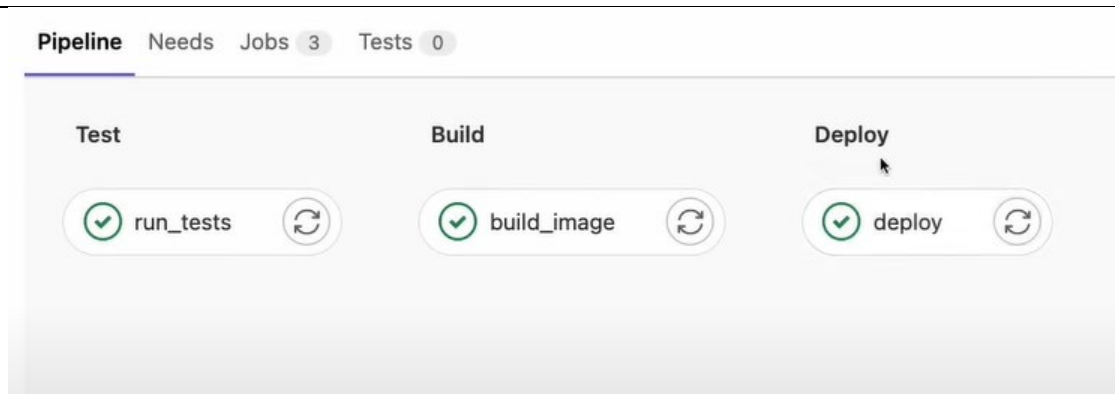
When you run the pipeline and go to the pipeline section you will find that a pipeline is being executed.

Status	Pipeline	Triggerer	Stages
 running In progress	Update .gitlab-ci.yml file #540128515  main -> 36be730a  latest		   

(different executions of the pipeline)

 passed 00:01:52 1 hour ago	Update .gitlab-ci.yml file #540018072  main -> dfba14a1 		  
 passed 00:01:00 2 hours ago	Update .gitlab-ci.yml file #540013635  main -> 3068526a 		  
 passed 00:00:45 3 hours ago	Add .gitlab-ci.yml #539951229  main -> f608a8e9 		  

After a while we find that the pipeline has finished.



And even if we look at the logs we can see that everything went perfectly.

**GitLab** Menu Search GitLab

gitlab-cicd-crash-cou...

- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
  - Pipelines
  - Editor
  - Jobs**
  - Schedules
  - Test Cases
  - Security & Compliance
  - Deployments
  - Packages & Registries
  - Infrastructure
  - Monitor
  - Analytics
  - Collapse sidebar

```

15 Fetching changes with git depth set to 20...
16 Initialized empty Git repository in /builds/nanuchi/gitlab-cicd-crash-course/.git/
17 Created fresh repository.
18 Checking out 36be730a as main...
19 Skipping Git submodules setup
21 Executing "step_script" stage of the job script
22 Using docker image sha256:27d049ce98db4e55ddfaec6cd98c7c9cfd195bc7e994493776959db33522383b for ruby:2.5 with digest rubysha256:ecc3e4f5da13d881a415c9692bb52d2b85b090f38f4ad99ae94f932b3598444b ...
23 $ chmod 400 $SSH_KEY
24 $ ssh -o StrictHostKeyChecking=no -i $SSH_KEY root@161.35.223.117 " docker login -u $REGISTRY_USER -p $REGISTRY_PASS && docker ps -aq | xargs docker stop | xargs docker rm && docker run -d -p 5000:5000 $IMAGE_NAME:$IMAGE_TAG"
25 Warning: Permanently added '161.35.223.117' (ECDSA) to the list of known hosts.
26 WARNING! Using --password via the CLI is insecure. Use --password-stdin.
27 WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
28 Configure a credential helper to remove this warning. See
29 https://docs.docker.com/engine/reference/commandline/login/#credentials-store
30 Login Succeeded
31 5f16c4b40b9e
32 a116ece5be1970979df2f2451bef75b48eeb3cd99bb89555b01108cf4bf18
34 Cleaning up project directory and file based variables
36 Job succeeded
  
```

**deploy**

Duration: 18 seconds  
 Finished: just now  
 Timeout: 1h (from project)  
 Runner: #12270837 (J2nyww-s) 4-blue.shared.runners-manager.gitlab.com/default

Commit 36be730a  
 Update .gitlab-ci.yml file

✓ Pipeline #540128515 for main  
 deploy

→ deploy

On the other hand, we can look inside the virtual machine to see if it has been executed correctly.

CONTAINER ID	IMAGE	COMMAND	CREATED
a116ece5be19	nanajanashia/demo-app:python-app-1.0	"unicorn -b 0.0.0.0..."	2 minutes ago
Up 2 minutes	0.0.0.0:5000->5000/tcp, :::5000->5000/tcp	vigilant_proskuriakova	

Finally, we will check that it is accessible from the internet.

**Python Demo** Info Monitor

**Python & Flask Demo App**

This is a simple web application written in Python and using Flask. It has been designed with cloud demos & containers in mind. Demonstrating capabilities such as auto scaling, deployment to Azure or Kubernetes, or anytime you want something quick and lightweight to run & deploy.

[GitHub Project](#)
[Docker Images](#)

[Get started with Azure & Python](#)

Microsoft ❤️ Open Source

v1.4.2 [Ben Coleman, 2018-2021]

