

18/03/2024

# YNOV - Devops



Luca Morgado & Romain Fromentin

## Contents table

Introduction .....	2
Instance configuration .....	3
Runner configuration .....	4
Distant dev server configuration .....	5
Linking distant with runner .....	6
Pipeline configuration .....	7
Deployment verification .....	9

## Introduction

During devops classes, we learnt it is important to be efficient throughout test and deployment process automatization. Pipelines are components of continuous integration, deployment and delivery.

In this project, we will deploy an application in gitlab through a virtual machine. A virtual machine will be the runner, another one the gitlab instance, and the last one will contain the application ready to be deployed. We are going to detail our process in several steps present below.

# Instance configuration

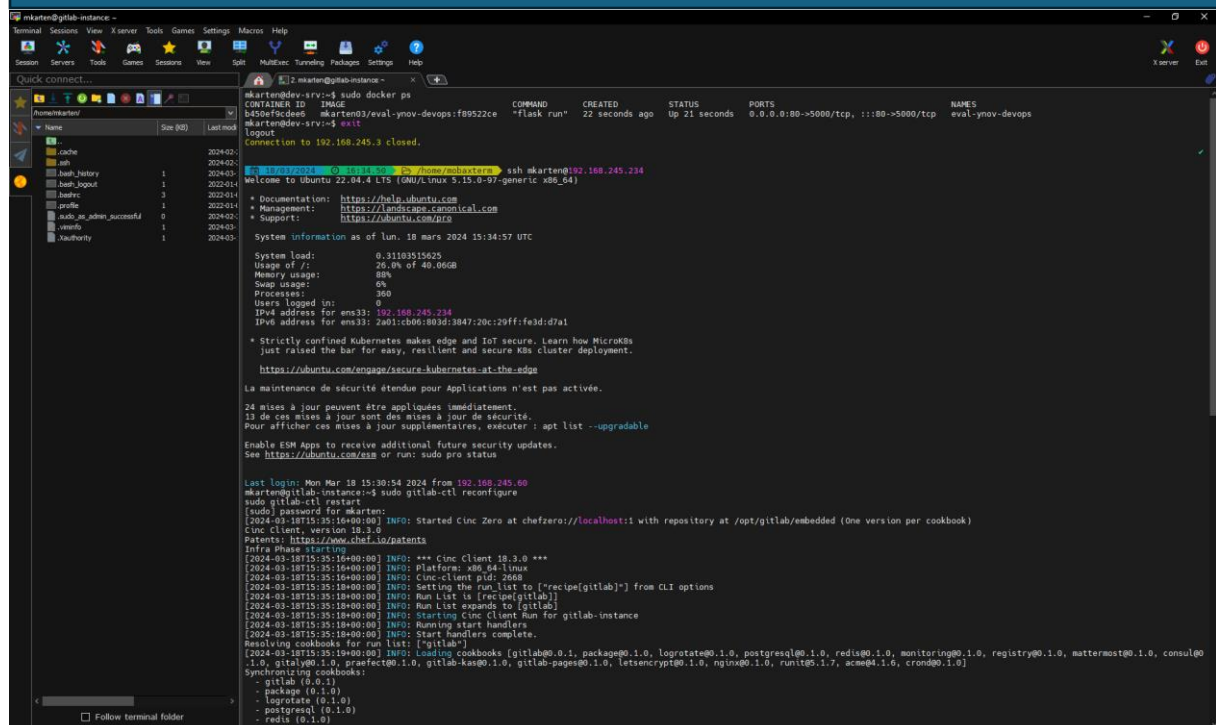
Instance ip: 192.168.245.234

Change in /etc/gitlab/gitlab.rb the parameter external\_url 'http://<ip\_adress>' to the ip of your runner instance (here our instance ip).

```
## GitLab URL
##! URL on which GitLab will be reachable.
##! For more details on configuring external_url see:
##! https://docs.gitlab.com/omnibus/settings/configuration.html#configuring-the-external-url-for-gitlab
##!
##! Note: During installation/upgrades, the value of the environment variable
##! EXTERNAL_URL will be used to populate/replace this value.
##! On AWS EC2 instances, we also attempt to fetch the public hostname/IP
##! address from AWS. For more details, see:
##! https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html
external_url 'http://192.168.245.234'
```

To apply configuration changes, we execute those commands:

```
gitlab-ctl reconfigure
gitlab-ctl restart
```



```
mkarten@gitlab-instance:~$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
b450efcde8    mkarten03/eval-ynov-devops:f0522ce  "flask run"              22 seconds ago Up 21 seconds 0.0.0.0:80->5000/tcp, :::80->5000/tcp eval-ynov-devops
mkarten@gitlab-instance:~$ exit
logout
Connection to 192.168.245.3 closed.

mkarten@gitlab-instance:~$ ssh mkarten@192.168.245.234
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-07-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of lun. 18 mars 2024 15:34:57 UTC

System load:  0.31103515625
Usage of /:   26.0% of 40.06GB
Memory usage: 88%
Swap usage:   0%
Processes:    260
Users logged in: 0
IPv4 address for ens33: 192.168.245.234
IPv6 address for ens33: 2a01:cb06:1803d:3847:20c:29ff:fe1d:d7a1

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.
  https://ubuntu.com/engage/secure-kubernetes-at-the-edge

La maintenance de sécurité étendue pour Applications n'est pas activée.

24 mises à jour peuvent être appliquées immédiatement.
13 de ces mises à jour sont des mises à jour de sécurité.
Pour afficher ces mises à jour supplémentaires, exécutez : apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Mon Mar 18 15:30:54 2024 from 192.168.245.60
mkarten@gitlab-instance:~$ sudo gitlab-ctl reconfigure
sudo gitlab-ctl restart
[sudo] password for mkarten:
[2024-03-18T15:35:16+00:00] INFO: Started Cinc Zero at chefzero://localhost:1 with repository at /opt/gitlab/embedded (One version per cookbook)
Cinc Client, version 18.3.0
Patents: https://www.chef.io/patents

Infra Phase starting
[2024-03-18T15:35:16+00:00] INFO: *** Cinc Client 18.3.0 ***
[2024-03-18T15:35:16+00:00] INFO: Platform: x86_64-linux
[2024-03-18T15:35:16+00:00] INFO: Cinc-client pid: 2668
[2024-03-18T15:35:16+00:00] INFO: Setting the run_list to ["recipe::gitlab"] from CLI options
[2024-03-18T15:35:16+00:00] INFO: Run List is ["recipe::gitlab"]
[2024-03-18T15:35:16+00:00] INFO: Run List expands to [gitlab]
[2024-03-18T15:35:16+00:00] INFO: Starting Cinc Client Run for gitlab-instance
[2024-03-18T15:35:16+00:00] INFO: Running start handlers
[2024-03-18T15:35:16+00:00] INFO: Start handlers complete.
[2024-03-18T15:35:19+00:00] INFO: Loading cookbooks [gitlab@0.0.1, logrotate@0.1.0, postgresql@0.1.0, redis@0.1.0, monitoring@0.1.0, registry@0.1.0, mattermost@0.1.0, consul@0.1.0, gitlab@0.1.0, prefect@0.1.0, gitlab-k8s@0.1.0, gitlab-pages@0.1.0, letsencrypt@0.1.0, nginx@0.1.0, runit@0.1.0, acme@0.1.0, cron@0.1.0]
Synchronizing cookbooks:
- gitlab (0.0.1)
- logrotate (0.1.0)
- postgresql (0.1.0)
- redis (0.1.0)
```

# Runner configuration

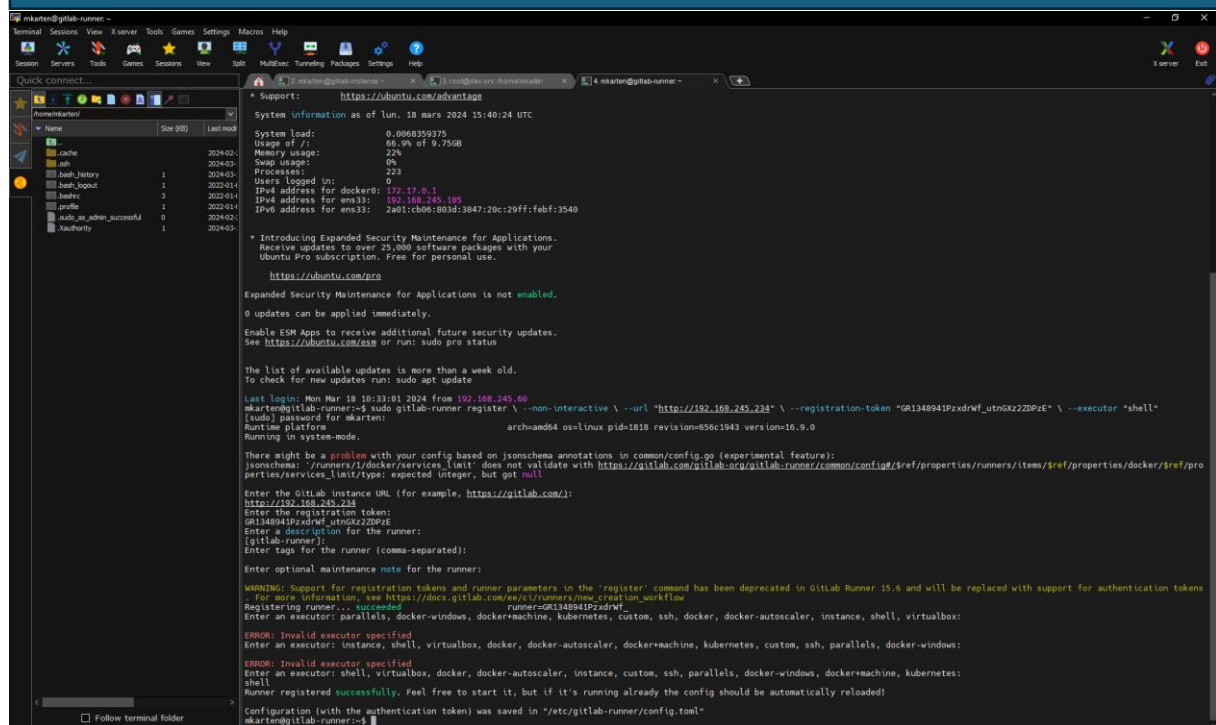
Runner ip: 192.168.245.105

Take the password from /etc/gitlab/initial\_root\_password and login as root on your gitlab instance.

Take the runner token in the CI/CD menu of your project and use it to register to your runner.

On the runner server, we register a gitlab runner (*instance ip address and runner token*):

```
sudo gitlab-runner register \
--non-interactive \
--url "http://<instance_ip_address>" \
--registration-token "<registration_token>" \
--executor "shell"
```



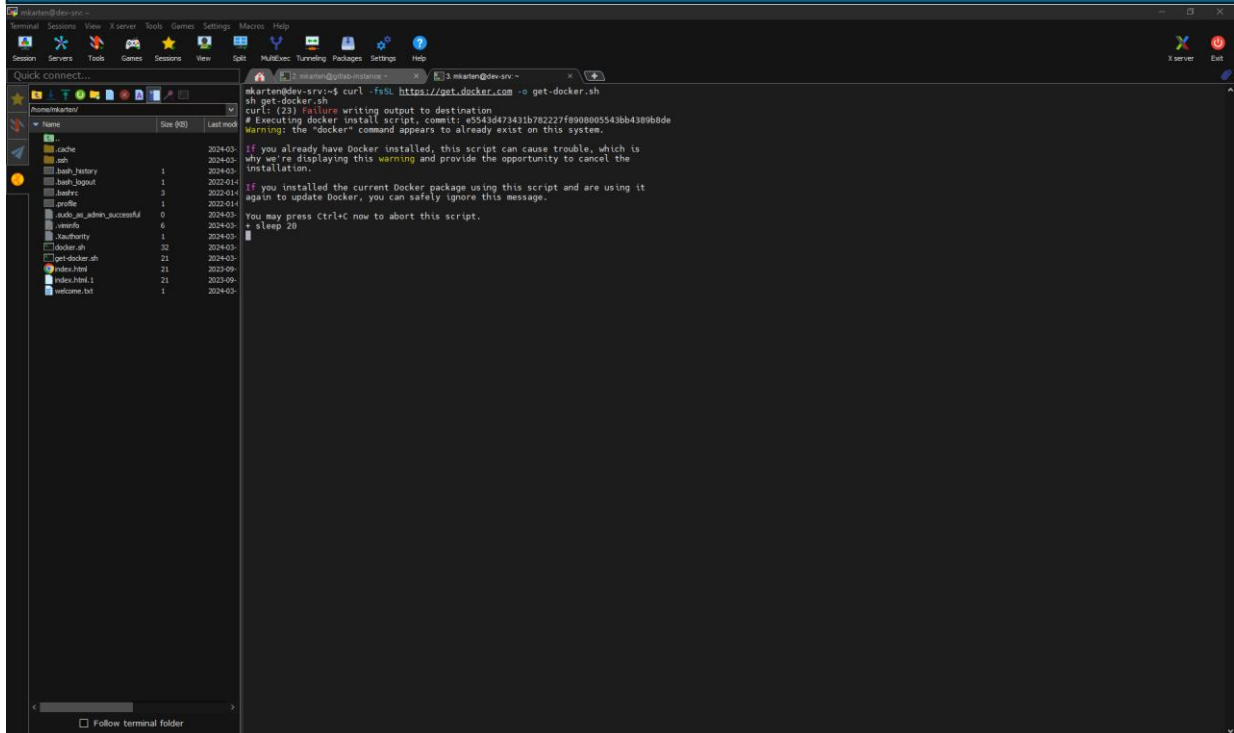
The screenshot shows a terminal window on a server named 'mkarten@gitlab-runner'. The terminal output displays system information for Ubuntu 22.04, including system load, memory usage, and network addresses. It then shows the execution of the 'gitlab-runner register' command with the following flags: --non-interactive, --url 'http://192.168.245.234', --registration-token 'GK1348941PxdwF\_utmGx2Z2DpE', and --executor 'shell'. The output confirms the registration was successful, showing the runner's name 'runner-GK1348941PxdwF' and its configuration path. A warning message indicates that support for registration tokens is deprecated in GitLab Runner 15.6 and will be replaced by authentication tokens. The terminal also shows a list of available updates and a message about the deprecated 'register' command.

# Distant dev server configuration

Distant server ip: 192.168.245.3

We need to install Docker on the distant server.

```
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh
```



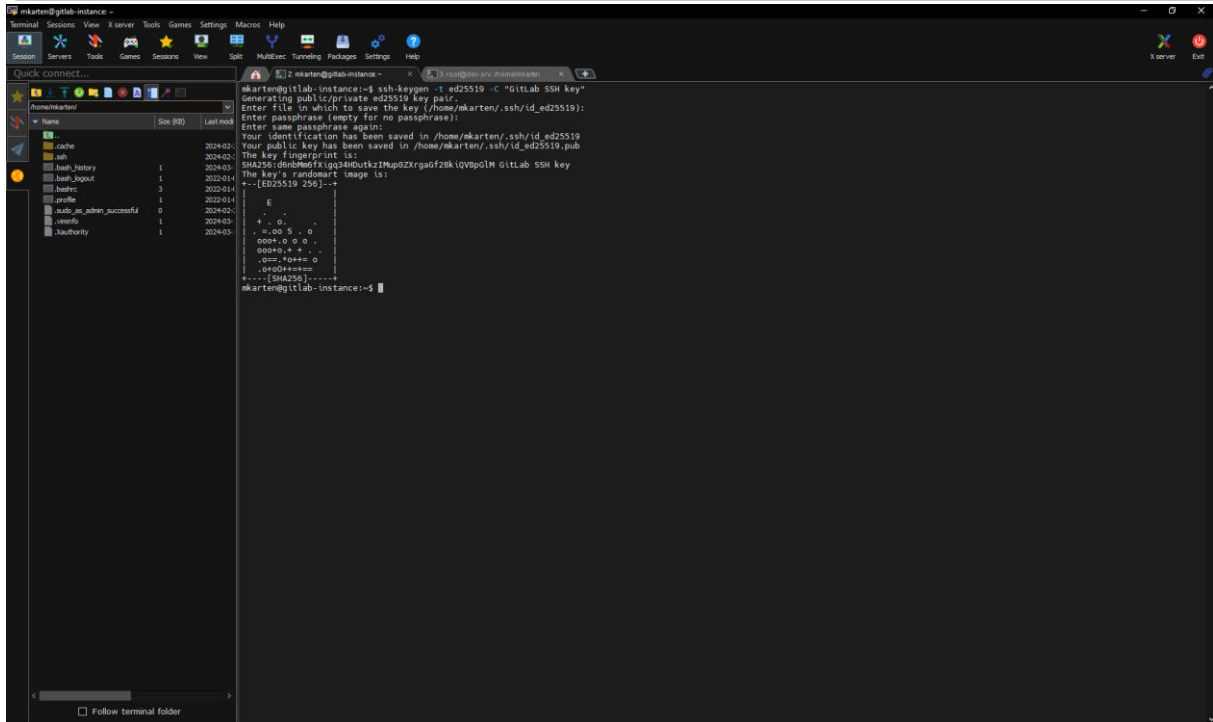
We also add the user at the end of /etc/sudoers so that we don't have to use a password when running docker commands.

```
mkarten ALL=(ALL) NOPASSWD: ALL
```

## Linking distant with runner

We generate a key pair with the following code:

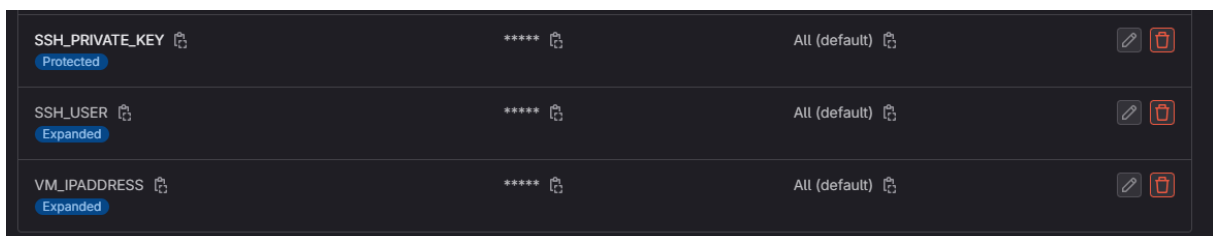
```
ssh-keygen -t ed25519 -C "Gitlab SSH key"
```



Connect to the distant server and add the public key to the authorized\_keys in:

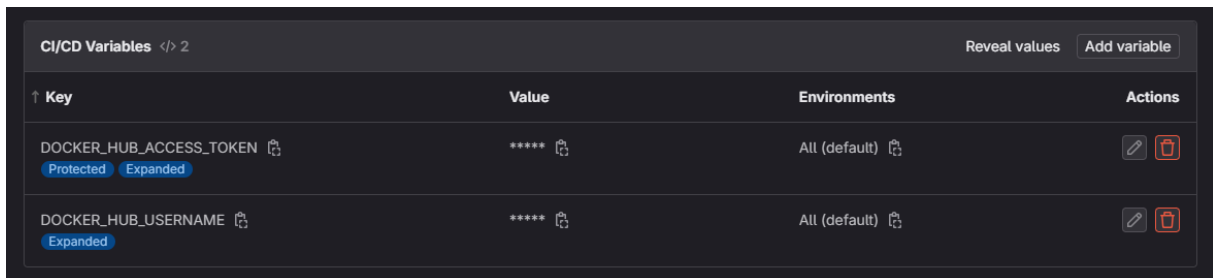
```
/home/<user>/.ssh
```











Add ssh private key, user and VM ip adress as environment variables in the CI CD menu in Gitlab parameters.



# Pipeline configuration

Add DockerHub login as environment variables in the CI CD menu in Gitlab parameters.



CI/CD Variables </> 2				Reveal values	Add variable
↑ Key	Value	Environments	Actions		
DOCKER_HUB_ACCESS_TOKEN  <span>Protected</span> <span>Expanded</span>	***** 	All (default) 	 		
DOCKER_HUB_USERNAME  <span>Expanded</span>	***** 	All (default) 	 		

Now we create the gitlab-ci.yml file for the pipeline configuration:

We first define three stages for the pipeline (build-image, publish-image and deploy-app)

```
stages:
- build-image
- publish-image
- deploy-app
```

After this, we configure each stage of the pipeline, here is the build stage:

```
build:
  stage: build-image
  script:
    - docker rmi -f $DOCKER_HUB_USERNAME/eval-ynov-devops || true
    - docker build -t eval-ynov-devops:$CI_COMMIT_SHORT_SHA .
```

We remove the existing image from the system, and build a new version named eval-ynov-devops and the short sha of the commit.

Here is the push stage:

```
push:
  stage: publish-image
  services:
  script:
    - docker login -u $DOCKER_HUB_USERNAME -p $DOCKER_HUB_ACCESS_TOKEN
    - docker tag eval-ynov-devops:$CI_COMMIT_SHORT_SHA
      $DOCKER_HUB_USERNAME/eval-ynov-devops:$CI_COMMIT_SHORT_SHA
    - docker push $DOCKER_HUB_USERNAME/eval-ynov-devops:$CI_COMMIT_SHORT_SHA
```

In this stage we connect to the docker hub account specified in the environment variable, tag the image with the short sha of the commit and push the image to docker hub.

Finally, the deploy stage:



```

deploy:
  stage: deploy-app
  before_script:
    - 'command -v ssh-agent >/dev/null || ( apk add --update openssh )'
    - eval $(ssh-agent -s)
    - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - ssh-keyscan $VM_IPADDRESS >> ~/.ssh/known_hosts
    - chmod 644 ~/.ssh/known_hosts
  script:
    - ssh $SSH_USER@$VM_IPADDRESS "sudo docker stop eval-ynov-devops || true"
    - ssh $SSH_USER@$VM_IPADDRESS "sudo docker rm eval-ynov-devops || true"
    - ssh $SSH_USER@$VM_IPADDRESS "sudo docker run -d --name eval-ynov-devops -
p 80:5000 $DOCKER_HUB_USERNAME/eval-ynov-devops:$CI_COMMIT_SHORT_SHA"

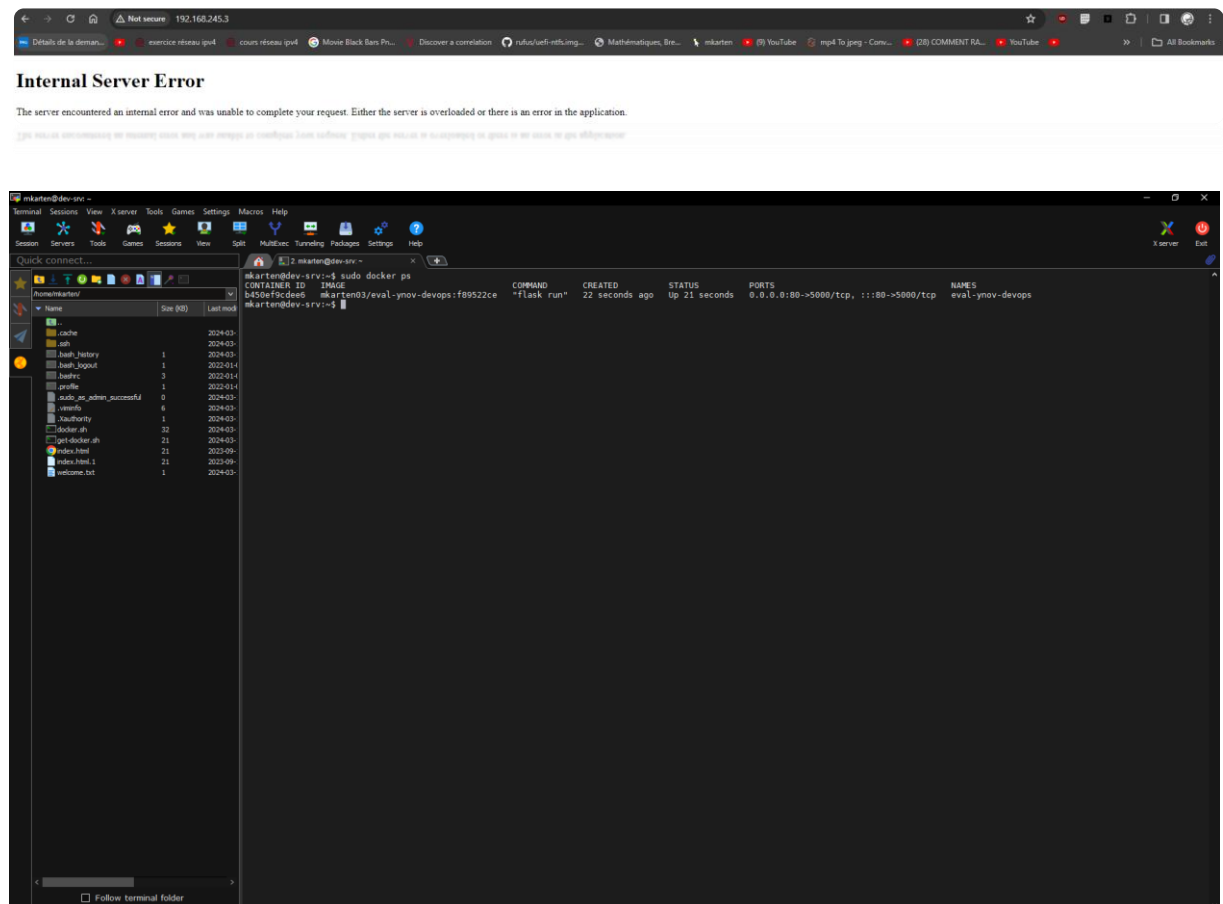
```

Before the script execution we do all the setup to enable ssh connection to the distant deployment server by adding the ssh private key, creating the .ssh directory, and adding the distant server ip to the known\_hosts file.

After this setup, we execute each command preceded by a ssh connection directive to execute the command on the distant server.

# Deployment verification

To verify the deployment, we go to our distant server ip address, and check we have this page.



After the deployment, we verify that the docker image has been pushed on dockerhub.

**mkarten03/eval-ynov-devops**

Updated 11 minutes ago

This repository does not have a description

Docker commands

To push a new tag to this repository:

`docker push mkarten03/eval-ynov-devops:tagname`

Public View

Tags

This repository contains 9 tag(s).

Tag	OS	Type	Pulled	Pushed
<a href="#">f89522ce</a>		Image	11 minutes ago	11 minutes ago
<a href="#">7cc4a244</a>		Image	2 hours ago	2 hours ago
<a href="#">5ced8129</a>		Image	an hour ago	3 hours ago
<a href="#">d072ed75</a>		Image	an hour ago	3 hours ago
<a href="#">c6eb4169</a>		Image	an hour ago	3 hours ago

[See all](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

Upgrade