18/03/2024

YNOV - Devops

Luca Morgado & Romain Fromentin

Contents table

Introduction

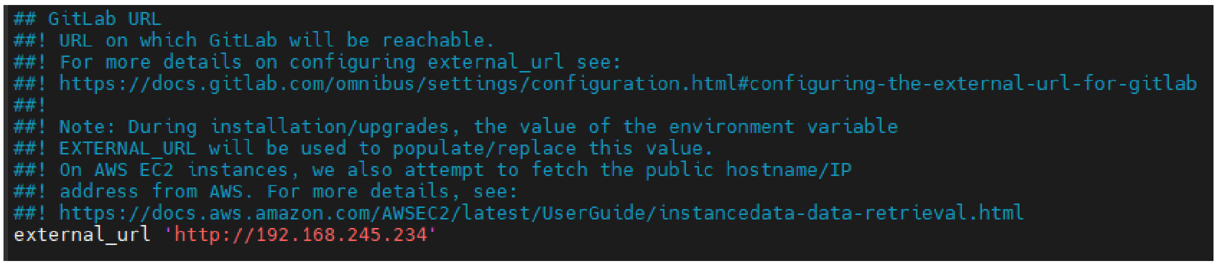
During devops classes, we learnt it is important to be efficient throughout test and deployment process automatization. Pipelines are components of continuous integration, deployment and delivery.

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
```

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_adress>" \
  --registration-token "<registration_token>" \
  --executor "shell"
```

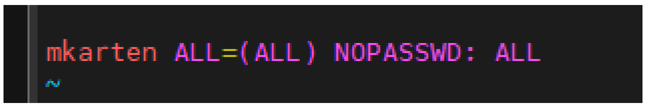
Distant dev server configuration

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

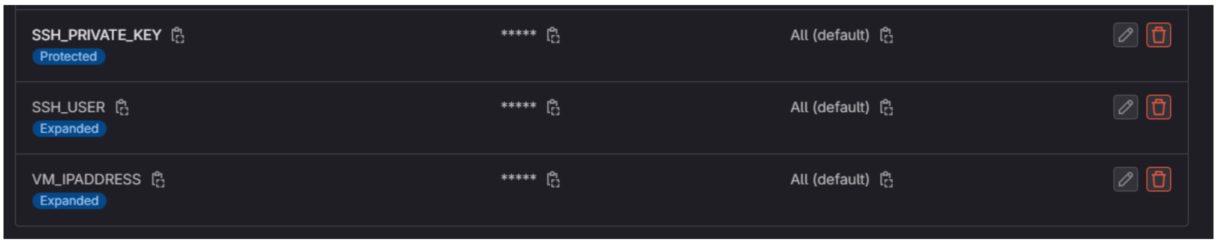
We generate a key pair with the following code:

```
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
```

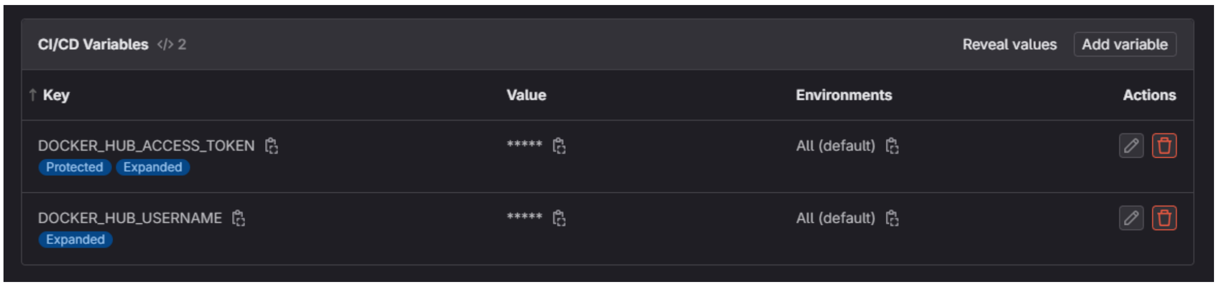
Add ssh private key, user and VM ip adress as environment variables in the CI CD menu in Gitlab parameters.



SSH_PRIVATE_KEY Protected	*****	All (default)
SSH_USER Expanded	*****	All (default)
VM_IPADDRESS Expanded	*****	All (default)

Pipeline configuration

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



↑ Key	Value	Environments	Actions
DOCKER_HUB_ACCESS_TOKEN Protected Expanded	*****	All (default)	
DOCKER_HUB_USERNAME Expanded	*****	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:

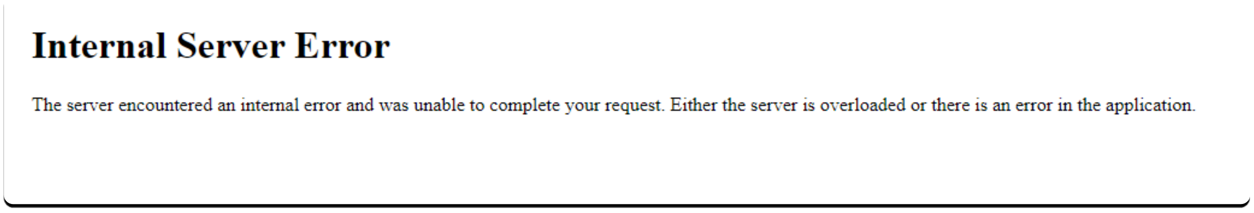
```
deploy:
  stage: deploy-app
  before_script:
    - 'command -v ssh-agent >/dev/null || ( apk add --update openssh )'
    - eval $(ssh-agent -s)
    - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
    - ssh-keyscan $VM_IPADDRESS >> ~/.ssh/known_hosts
    - 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

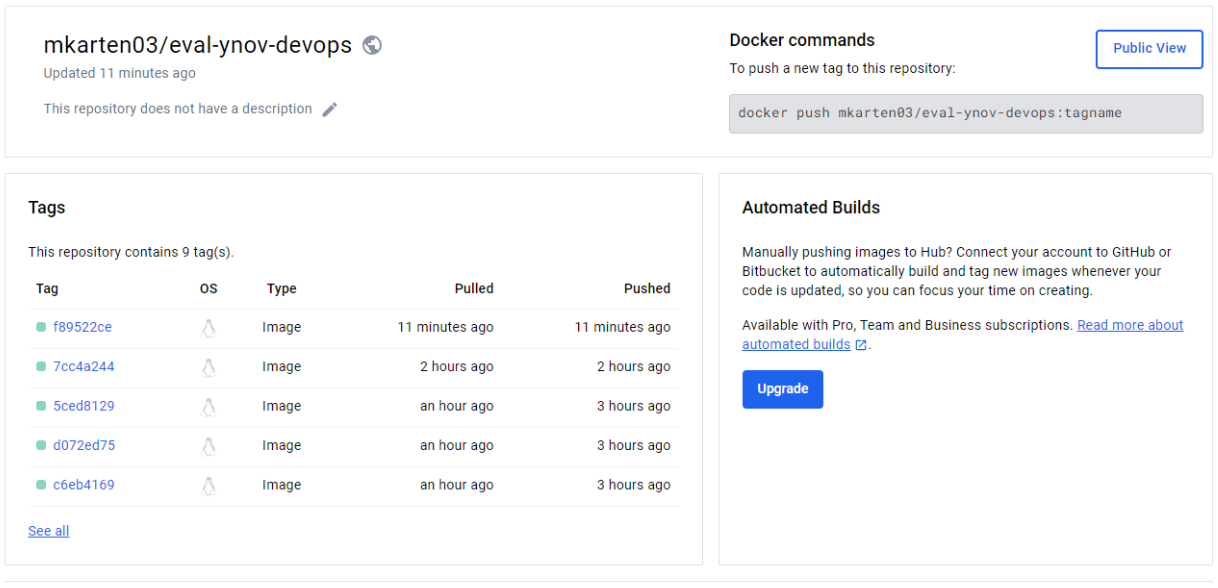
To verify the deployment, we go to our distant server ip address, and check we have this page.



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.

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
```

Tags

This repository contains 9 tag(s).

Tag	OS	Type	Pulled	Pushed
f89522ce		Image	11 minutes ago	11 minutes ago
7cc4a244		Image	2 hours ago	2 hours ago
5ced8129		Image	an hour ago	3 hours ago
d072ed75		Image	an hour ago	3 hours ago
c6eb4169		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.