Table des matières

[**Infrastructure 2**](#_Toc70188680)

[1. Pré-requis 2](#_Toc70188681)

[2. Installation de jenkins 2](#_Toc70188682)

[3. Déploiement du serveur applicatif 3](#_Toc70188683)

[1. Conception du repo git 3](#_Toc70188684)

[2. Utilisation du repo git 4](#_Toc70188685)

[4. Installation du registry 4](#_Toc70188686)

[1. Conception du repo git 4](#_Toc70188687)

[2. Utilisation du repo git 5](#_Toc70188688)

[5. Code Repository 5](#_Toc70188689)

[1. Pré-requis 5](#_Toc70188690)

[2. Installation de GitLab local 5](#_Toc70188691)

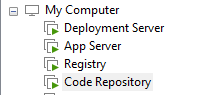
[3. Utilisation de GitLab 6](#_Toc70188692)

[**L’intégration continue 6**](#_Toc70188693)

# Infrastructure

# Pré-requis

Monter 4 VM dans VMware de la façon suivante :



Sur chacune des machines virtuelles, installer :

* Centos7.6.
* Editer les préférences réseaux via la commande *nmtui* afin d’automatiquement lancer la détection de réseau.
* Installer Docker via les instructions du site [*https://docs.docker.com/engine/install/centos/*](https://docs.docker.com/engine/install/centos/)
* Installer Git via la commande yum install git

De plus, puis-ce que nous allons souhaitez démarrer automatique nos applications au démarrage des serveurs, il faudra démarrer automatique docker engine. Le démarrage automatique des images docker se fera via les paramètres des docker-compose.yml.

* Démarrer docker engine automatiquement via la commande systemctl enable /usr/lib/systemd/system/docker.service

# Installation de jenkins

Accès à la documentation en ligne : https://www.jenkins.io/doc/book/installing/linux/

Après installation de la VM, l’installation de wget, suivi du process

sudo wget -O /etc/yum.repos.d/jenkins.repo \

https://pkg.jenkins.io/redhat-stable/jenkins.repo

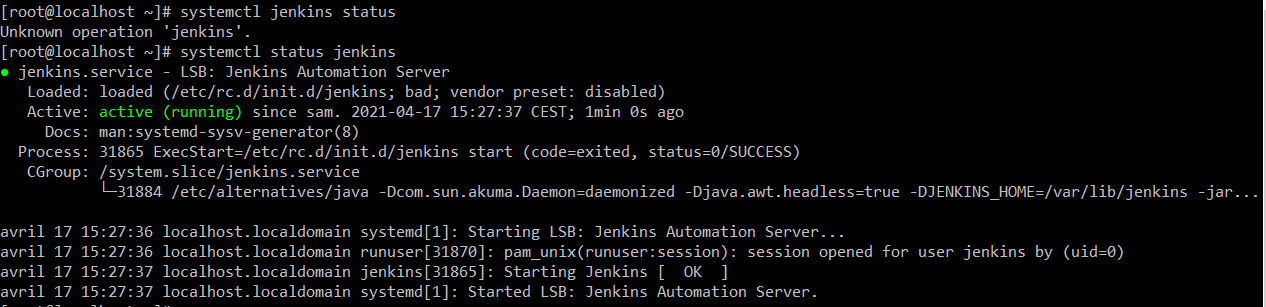
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum upgrade

sudo yum install jenkins java-1.8.0-openjdk-devel

sudo systemctl daemon-reload

Le resultat que nous avons via la commande *systemctl status jenkins*



Puis ouvrir les port dans le firewall

YOURPORT=8080

PERM="--permanent"

SERV="$PERM --service=jenkins"

firewall-cmd $PERM --new-service=jenkins

firewall-cmd $SERV --set-short="Jenkins ports"

firewall-cmd $SERV --set-description="Jenkins port exceptions"

firewall-cmd $SERV --add-port=$YOURPORT/tcp

firewall-cmd $PERM --add-service=jenkins

firewall-cmd --zone=public --add-service=http --permanent

firewall-cmd –reload

# Déploiement du serveur applicatif

## Conception du repo git

Le dossier App\_Server est organisé de la façon suivante :

D’un point de vue fichier, nous avons

**Simple\_api**

* Monte une image python :2.7-stretch comme demander par POZOS
* On installe l’ensemble des modules nécessaires au fonctionnement de flask
* On monte un volume nommé « data » dans le container
* On copie dans la racine du container le fichier student\_age.py
* On copie dans le volume monté le fichier student\_age.json
* On spécifie la commande à lancer dans le container afin de pointer sur le fichier python
* On expose le container sur le port 5000

**website**

* On monte l’image l’image php :8.0-apache comme demander par POZOS
* On copie le fichier index.php dans le dossier /var/www/html du container
* On expose le container sur le port 8080

**Docker-compose**

Lance le container php-apache

* Nomme le container : website
* Monte le container via un build du Dockerfile
* Demande le lancement automatique du container au démarrage de Docker
* Expose l’image sur le port 8080
* Créer une dépendance afin de lancer ce container avant l’autre
* Stipule les credentials via 2 variables d’environnement

Lance le container flask

* Nomme le container : flask
* Monte le container via un build du Dockerfile
* Demande le lancement automatique du container au démarrage de Docker

## Utilisation du repo git

Téléchargez le dossier de travail via la commande

git clone <https://github.com/FlorianVigier/4DVOP_POZOS.git>

Puis, vous devrez vous rendre dans le dossier télécharger puis dans le dossier App\_Server

cd 4DVOP\_POZOS/App\_Server

Puis lancer la commande

docker-compose up

L’application est lancé sur votre serveur.

# Installation du registry

## Conception du repo git

Le dépôt du registre ne compose qu’un fichier docker-compose construit de la façon suivante :

**Docker-compose**

Lance le container regristry

* Nomme le container : registry en version 2.7
* Monte l’image registry supporté par Docker
* Demande le lancement automatique du container au démarrage de Docker
* Expose l’image sur le port 5000
* On monte un volume local storage dans /var/lib/registry afin de presister les images

Lance le container front\_end

* Nomme le container : registry\_interface
* Monte l’image joxit/docker-registry-ui:static
* Demande le lancement automatique du container au démarrage de Docker
* Expose l’image sur le port 80
* Créer une dépendance afin de lancer ce container avant l’autre
* Stipule les 2 variables d’environnement afin de pointer la page sur le registre

## Utilisation du repo git

Téléchargez le dossier de travail via la commande

git clone <https://github.com/FlorianVigier/4DVOP_POZOS.git>

Puis, vous devrez vous rendre dans le dossier télécharger puis dans le dossier Registry

cd 4DVOP\_POZOS/Registry

Puis lancer la commande

docker-compose up

# Code Repository

## Pré-requis

Ne pas oublier d’éditer les préférences de la VM Centos afin de monter à 4Go de ram.

Cela permettra d’éviter les problèmes de TimeOut sur certains services.

Installation de Gitlab en local

## Installation de GitLab local

Pour installer Gitlab en local, il suffira de suivre les instruction présentent sur le site [*https://about.gitlab.com/install/#centos-8*](https://about.gitlab.com/install/#centos-8)mais vous trouverez ci-dessous un extrait des commandes.

**Installation des dépendances**

sudo dnf install -y curl policycoreutils openssh-server perl

sudo systemctl enable sshd

sudo systemctl start sshd

sudo firewall-cmd --permanent --add-service=http

sudo firewall-cmd --permanent --add-service=https

sudo systemctl reload firewalld

sudo dnf install postfix

sudo systemctl enable postfix

sudo systemctl start postfix

**Installation du package GitLab**

curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ee/script.rpm.sh | sudo bash

sudo EXTERNAL\_URL="https://gitlab.example.com" dnf install -y gitlab-ee

L’installation se terminera par le message suivant :



## Utilisation de GitLab

La page de GitLab est accessible via l’IP de la VM. La premiere fois, vous aurez à saisir un nouveau mot de passe. Le login se fera via le compte *root*.

* User : Root
* Password : a définer.

Pour creer le lien entre GitLab et le registry

Edit de /etc/gitlab/gitlab.rb

# L’intégration continue