# Plan d'action

## I ) Création d'un répertoire sur le desktop et d'un repo git

- Créer un nouveau répertoire sur le bureau pour stocker le projet :

  $ mkdir TPGitlab

  $ cd TPGitlab

- Instancier un repository git et faire un premier commit/push

  $ git init

  $ git commit -m "first commit, init gitlab project"

  $ git checkout master

  $ git add *

  $ git push origin master

## II ) Installation de Docker | Docker-compose

- Mettre à jour apt

```
dinda@DESKTOP-CHHQJKK MINGW64
$ sudo apt-get update
```

- Récupérer les packages de Docker :

```
dinda@DESKTOP-CHHQJKK MINGW64 ~
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose
-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- Vérifier l'installation de Docker-compose

```
nicolas@nicolas-VirtualBox:~/Documents/TPGITLAB/tpgitlab$ sudo docker-compose --version
docker-compose version 1.29.2, build 5becea4c
```

- Une fois que docker-compose est bien installé, créer un fichier docker-compose.yml et l'éditer de telle sorte :

```yaml
version: "3.9"
services:
  web:
    image: 'gitlab/gitlab-ce:latest'
    hostname: 'gitlab.example.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://gitlab.example.com'
    ports:
      - '80:80'
      - '22:22'
    volumes:
      - ./config:/etc/gitlab
      - ./logs:/var/log/gitlab
      - ./data:/var/opt/gitlab

  runner:
    image: 'gitlab/gitlab-runner:latest'
    restart: always
    volumes:
      - ./gitlab-runner/config:/etc/gitlab-runner
      - /var/run/docker.sock:/var/run/docker.sock:ro
```

- modification du  fichier `/etc/hosts` et ajouter la ligne suivante afin que la résolution DNS soit correctement réalisé :

  **<IP_VM>  gitlab.example.com**

- Pour exécuter docker-compose et crée une image taper la commande suivante :

```
dinda@DESKTOP-CHHQJKK MINGW64 ~
$ sudo docker-compose up -d
```

- Vérifier que l'image à bien été créer dans le container

```
nicolas@nicolas-VirtualBox:~/Documents/TPGITLAB/tpgitlab$ sudo docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS                   PORTS
                                                                                       NAMES
9917982a8a8a   gitlab/gitlab-ce:latest        "/assets/wrapper"        2 hours ago     Up 58 minutes (healthy)  0.0.0.0:22->22/tcp, :::22->22/t
cp, 0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp   tpgitlab_web_1
5a3ec8d8866d   gitlab/gitlab-runner:latest    "/usr/bin/dumb-init …"   2 hours ago     Up 15 minutes
                                                                                       tpgitlab_runner_1
b17e28d4705a   gitlab/gitlab-runner:latest    "/usr/bin/dumb-init …"   2 hours ago     Up About an hour
                                                                                       gitlab-runner
```

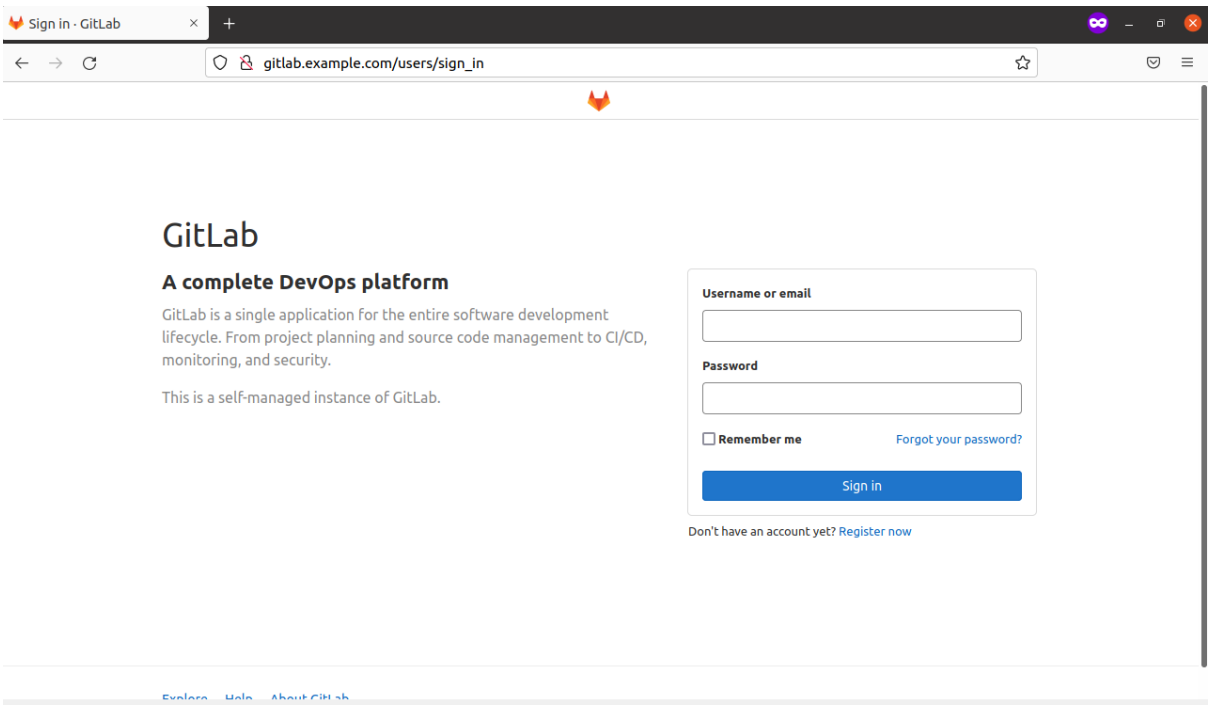
- Vérifier les différents services :

## III ) Initialisation de GitLab CI

- on se rend sur notre machine à l'adresse local suivante en tapant l'adresse dans une navigation privée:

  **http://gitlab.example.com**

- On arrive ensuite sur l'interface de GitLabCI :




- Pour se connecter ici, on va utiliser l'username : root et le password sera récupérable dans le fichier '/etc/gitlab/initial_root_password'

/!\ Attention le ficher contenant le mot de passe se supprime au bout de 24h ! /!\

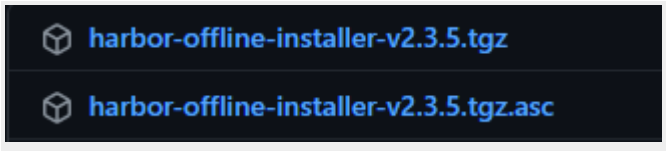
Une bonne pratique serait de stocker le password dans un autre fichier texte.

- Puis il faut ajouter un administrateur :
  1. Menu > Admin > Users
  2. New user
  3. Remplir les informations
  4. Access level : Admin
  5. Create user

- On va ensuite créer le projet en ajoutant la clé de SSH du repo git lié à son compte personnel

## IV) Installation de Harbor

- Tout d'abord, On va télécharger les packages nécessaires




- Ensuite grâce au fichier .asc on va obtenir une clé public :

```
$ gpg --keyserver hkps://keyserver.ubuntu.com --receive-keys 644FF454C0B4115C
```

- On va ensuite extraire le package

```
$ tar zxvf harbor-offline-installer-v*.tgz
```

- Ce qui va ensuite créer le dossier Harbor où dans ce dossier on aura un harbor.yml
- On va générer la clé privé de certificat CA :

```
$ openssl genrsa -out ca.key 4096
```

```
$ openssl req -x509 -new -nodes -sha512 -days 3650 \
-subj
"/C=CN/ST=Beijing/L=Beijing/O=example/OU=Personal/CN=your
domain.com" \ -key ca.key \ -out ca.crt
```

- On génère ensuite le certificat

```
$ openssl req -x509 -new -nodes -sha512 -days 3650 \
-subj /C=CN/ST=../L=../O=example/OU=../CN=yourdomain.com"
\-key ca.key \ -out ca.crt
```

(Les petits points devront être adaptés)

Génération de la clé privée

```
$ openssl genrsa -out yourdomain.com.key 4096
```

- Générer une demande de signature de certificat

```
$ openssl req -sha512 -new \ -subj
"/C=CN/ST=Beijing/L=Beijing/O=example/OU=Personal/CN=yourdomai
n.com" \ -key yourdomain.com.key \ -out yourdomain.com.csr
```

- On va ensuite créer un fichier avec une extension v3.ext qui va servir à générer un certificat pour Harbor :

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipher
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1=yourdomain.com
DNS.2=yourdomain
DNS.3=hostname
```

- On génère le certificat pour Harbor grâce à ce fichier

- Suite à toutes ces commandes des fichiers sont créés comme :


- On va ensuite copier le certificat et la clé du serveur dans le dossier hôte de Harbor

```
$ cp yourdomain.com.crt /data/cert/
```

```
$ cp yourdomain.com.key /data/cert/
```

- Par la suite on va convertir le .crt en .cert pour l'utiliser avec Docker car le "daemon" interprète le .crt en tant que certificat et le .cert en certificat client.

- Création de l'environnement pour utiliser Harbor où l'on va copier le certificat, la clé ainsi que le CA dans le certificat de Docker

```
$ cp yourdomain.com.cert /etc/docker/certs.d/yourdomain.com/
```

```
$ cp yourdomain.com.key /etc/docker/certs.d/yourdomain.com/
```

```
$ cp ca.crt /etc/docker/certs.d/yourdomain.com/
```

```
/etc/docker/certs.d/
    └── yourdomain.com:port
        ├── yourdomain.com.cert
        ├── yourdomain.com.key
        └── ca.crt
```

- Redémarrer Docker Engine

```
$ systemctl restart docker
```

- Modifié le fichier harbor.yml en mettant le chemin le certificat et la clé

```
https:
  # https port for harbor, default is 443
  port: 443
  # The path of cert and key files for nginx
  certificate: /etc/docker/certs.d/yourdomain.com/yourdomain.com.cert
  private_key: /etc/docker/certs.d/yourdomain.com/yourdomain.com.key
```

- Vérifier qu'aucune application est sur le port que vous renseignez avec la commande

```
$ nestat -lptn
```

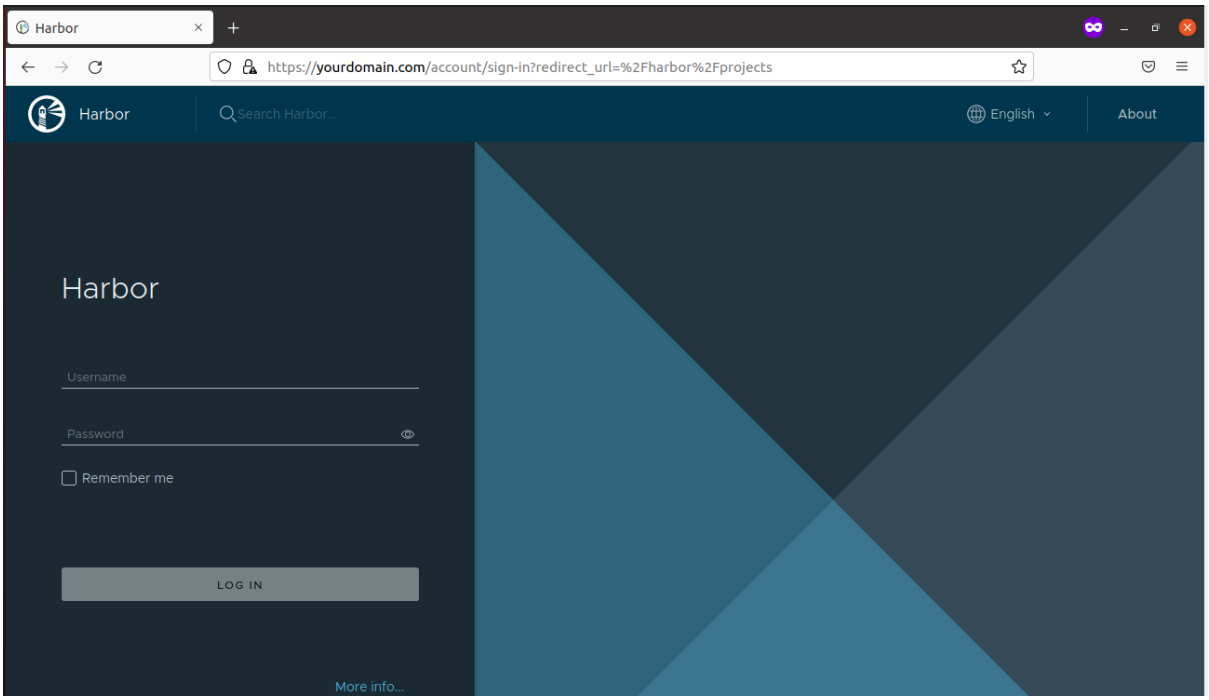
- On peut maintenant déployer l'interface Harbor en lançant le script prepare créer lors du dézippage de harbor

```
$ ./prepare
```

```
$ docker-compose down -v
```

```
$ docker-compose up -d
```

- Si maintenant nous ouvrons un nouvel onglet avec le site qu'on a enregistré, ici "yourdomain.com". Nous avons accès à l'interface de Harbor




- Pour s'y connecter l'identifiant par défaut est admin et le mot de passe est Harbor12345.

- On se connecte à Harbor avec le client Docker
  ```
  $ docker login yourdomain.com
  ```

- Pour finir on utilise le script install.sh pour installer les images Harbor qui vas aussi checker si
  - docker est installé,
  - docker-compose est installé
  - Charger les images Harbor
  - Préparer l'environnement de Harbor
  - Démarrer Harbor

# Erreur rencontrées

- Lucas sur son ordi n’a pas eu assez de RAM pour faire fonctionner GITLAB-CI

- Lors de la deuxième séance la perte du mot de passe pour se connecter sur GITLAB qui a se supprime au bout de 24h

- A l’exécution de GITLAB nous avons eu un problème de port car une application s'exécute déjà sur ce port 80

- Difficulté à générer les clés public pour le déploiement de Harbor