Introduction

- L’objectif de ce cas d’utilisation est de faire la démonstration pratique de la plupart des éléments techniques appris durant le cursus DevOps.

- Ce cas d’utilisation consiste donc logiquement à rassembler les aspects pratiques découverts dans le module 6 et de les combiner autour d’un infrastructure pour réaliser en particulier une CI/CD de notre application utilisant Jenkins.

- Ce sujet de TP est loin d’être simple :
  - N’hésitez pas à demander de l’aide au formateur.
  - Collaborez et partagez la compréhension des enjeux dans le groupe.
  - Le sujet est susceptible d’évoluer au fur et à mesure en fonction de vos retours et demandes d’information.

Rendu

- Le rendu du TP est à effectuer par groupe.

- Pour chaque groupe les éléments suivant devront être présentés lors de la présentation finale du cursus :

  - Une présentation décrivant les différents éléments de l'infrastructure et leurs objectifs ainsi que les choix réalisés lors de la réalisatio
  - On peut se servir de diapositives afin d'avoir un support oral. L'idée est de voir la gestion du temps, l'expression orale et évidemmen

    le côté technique. Et attention, à la répartition de parole dans le groupe, chacun doit occuper sa place.
  - La qualité des diapositives est notée également.
  - La présentation dure 10 mn, 5 mn de plus de questions
  - Pas de rapport écrit à part les diapositives

Objectifs

- Mettre en œuvre un système d'intégration continue et de livraison continue DevOps avec Jenkins
- Construire une image Docker capable de servir à l'application WEB/Nginx
- Automatiser la construction d'images Docker
- Mettre à jour et livrer automatiquement des images Docker
- Un dépôt sur GitHub contenant le code source de l'application et un dépôt sur Docker hub contenant l'application à déployer sur l'infrastructure.

Etudes de cas : 30 mn

Vous êtes spécialisé dans le développement d'application Web et vous avez d'industrialiser vos développements.

Vous décidez de mettre en place l'automatisation à partir d'une application Web pilote.

Décrire le workflow DevOps **de haut niveau** pour déployer l'application Web à l'aide de pratiques CI/CD.

Solution : workflow

Nouvelles fonctionnalités ou correctifs

Environnement de développement
Appi Web/Nginx
Git/Docker

Pull du code source

Push des modifications dans la forge

Forge logicielle
Repository github

Détection d'un changement de la forge et récupération du code source

Pipeline CI/CD
Jenkins

1 Vérification qualités du code (Linting)
2 Construction de l'application
3 Exécution des tests (unitaire, intégration, fonctionnelle)
4 Création de la release

Livraison de la release (Artefact)

Espace de livraison
Registre image
Web/Nginx
Dockerhub

Déploiement de la solution en production

Environnement de production

Mise en œuvre

1 Mise en place de la gestion du code source avec GIT

Installer git (voit https://git-scm.com/downloads)

- ❑ Ouvrir un terminal
- ❑ Installer git

```
sudo apt-get install git
```

Récupérer le projet contenant le code de l'application et les scripts CI/CD

- ❑ Ouvrir un terminal
- ❑ Créer un répertoire de travail :

```
cd && mkdir monTP && cd monTP
```

- ❑ Récupérer le code source et les scripts shell du formateur dans le dossier de travail :

```
git clone https://github.com/fpicot31/Jenkins-docker.git&&mv ./Jenkins-docker/* . &&rm -rf ./Jenkins-docker
```

Créer un dépôt (dossier géré par git) qui vous appartient

- ❑ Ouvrir un terminal et se placer dans le dossier de travail :

```
cd monTP
```

- ❑ Initialiser le dépôt (transforme le dossier de travail monTP en un dossier géré par git :

```
git init
```

- ❑ Suivre tous les fichiers du dossier de travail (dire à git qu'il faut inclure la version actuelle des fichiers dans git) :

```
git add .
```

- ❑ Valider vos modifications pour créer ce qu'on appelle un commit, c'est-à-dire une étape validée du code :

```
git config --global user.name "<votre nom>«
git config --global user.email "<votre email>«
git commit –m « initialisation du dépot »
```

2 Partage du code source sur la forge GitHub

Créer un repository distant

❑ Rendez-vous sur https://gitlhub.com et créer votre compte.

❑ Créer un repository avec le nom **monTP**

❑ Ajouter un Token :

(en haut à droite) `Settings`

(en bas à gauche) `Développer settings->Personal access tokens->Tokens->Generate new Token`

Partager le code source

❑ Ouvrir un terminal et se placer dans le dossier de travail :

```
cd monTP
```

❑ Pousser votre code source sur le forge :

```
git remote add origin https://<nom du token>:<valeur du token>@gitlab.com/<votre login>/monTP.git
```

```
git push –u origin master
```

3 Développement de l'application Web avec Docker

Installer Docker

❑ Option 1 : rendez-vous sur https://docs.docker.com/engine/install et suivre la procédure d'installation.

❑ Option 2 (ubuntu uniquement) : utiliser l'utilitaire snap :

```
sudo snap install docker
```

❑ Option : ajouter l'utilisateur au groupe docker pour éviter de taper la commande sudo :

```
sudo usermod –aG docker $USER
```

❑ Démarrer Docker :

```
sudo snap start docker
```

Tester l'application

❑ Ouvrir un terminal et se placer dans le dossier de travail :

```
cd monTP
```

❑ Construire l'image de l'application Web à partir du fichier Dockerfile :

```
docker build –t monServeurWeb .
```

❑ Lancer l'application Web à partir de l'image **monServeurWeb** :

```
docker run -d --name serveurweb -p 8081:80  monServeurWeb
```

❑ Ouvrir un navigateur sur le port 8081 en local :

http://localhost:8081

4 Mise en place de l’espace de livraison avec dockerhub

Créer un repository d’image docker

❑ Rendez-vous sur https://hub.docker.com et créer votre compte.

❑ Créer un repository avec le nom **monAppliWeb**

5 Création du pipeline de livraison

Mettre à jour le pipeline

❑ Ouvrir un terminal et se placer dans le dossier de travail :

```
cd monTP
```

❑ Modifier le fichier jenkinsfile en utilisant votre login dockerhub :

```
nano jenkinsfile
```

```
node {
    def app

    stage('Clonage des sources') {
        checkout scm
    }

    stage('Build') {
        app = docker.build("<votre login dockerhub>/monAppliWeb")
    }

    stage('Test') {
        docker.image('<votre login dockerhub>/monAppliWeb').withRun('--rm -p 80:80 --name devops') { c ->
        sh 'docker ps'
        sh 'docker exec devops curl localhost:80'
        sh 'echo "Tests passed"'
        }

    }

    stage('Livraision sur DockerHub') {
        docker.withRegistry('https://registry.hub.docker.com', 'docker-hub-credentials') {
            app.push("${env.BUILD_NUMBER}")
            app.push("1.0")
        }
    }
}
```

❑ Pousser le pipeline sur la forge GitLab :

```
git add . && git commit –m «creation du pipeline» && git push
```

6 Mise en place du CI/CD avec Jenkins

Démarrer Jenkins

❑ Ouvrir un terminal et se placer dans le dossier de travail :

```
cd monTP
```

❑ Lancer le script shell de démarrage :

```
./start.sh
```

❑ Vérifier le démarrage des containers myjenkins-blueocean:2.361.4-1 et docker:dind :

```
docker ps
```

```
francois@francois-VirtualBox:~/tp-devops/Jenkins-docker$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED              STATUS              PORTS
                                                                                    NAMES
3ab231460627   myjenkins-blueocean:2.361.4-1      "/usr/bin/tini -- /u…"   About a minute ago   Up About a minute   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000-
>50000/tcp, :::50000->50000/tcp                                                     jenkins-blueocean
994d880bf9c8   docker:dind                        "dockerd-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:2376->2376/tcp, :::2376->2376/tcp, 0.0.0.0:3000->
3000/tcp, :::3000->3000/tcp, 2375/tcp, 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   jenkins-docker
```

Configurer compte administrateur

❑ Depuis un terminal, entrer dans le container myjenkins-blueocean:2.361.4-1 :

```
docker exec -it myjenkins-blueocean:2.361.4-1
```

❑ Noter la clé générée par Jenkins et sortir du container :

```
cat /var/jenkins_home/secrets/initialAdminPassword
^D
```

❑ Ouvrir un navigateur sur http://localhost:8080 et entrer la clé demandée

❑ Créer le premier compte administrateur et installer les plugin de démarrage par défaut.

7 Mise en place du CI/CD avec Jenkins (suite)

Créer les credentials

- ❑ Revenir dans le navigateur sur http://localhost:8080
- ❑ Ajouter les credentials dockerhub dans la partie **Administration Jenkins->Manage Credentials**

Nom utilisateur = <votre login dockerhub> Mot de passe = <votre mot de passe> Id = docker-hub-credentials

- ❑ Ajouter les credentials github dans la partie **Administration Jenkins->Manage Credentials**

Nom utilisateur = <votre login github> Mot de passe = <votre mot de passe> Id = github-credentials Description = github credential

Créer le pipeline Jenkins

- ❑ Revenir dans le tableau de bord Jenkins
- ❑ Sélectionner **Nouveau Item** et entrer un nom
- ❑ Sélectionner **Pipeline** et positionner les propriétés :

Scrutation de l'outil de gestion de version avec un planning : * * * * *

Pipeline script from SCM

SCM : Git

Repositories URL : URL https://github.com/<Votre login github>/monTP.git

Credentials : github credential

Branches to build : */master

Script Path : jenkinsfile

8 Déclenchement du CI

Modifier l'application Web

❑ Ouvrir un terminal et se placer dans le dossier de travail :

```
cd monTP
```

❑ Modifier le fichier Dockefile et mettre « BRAVO !!! » en page d'accueil :

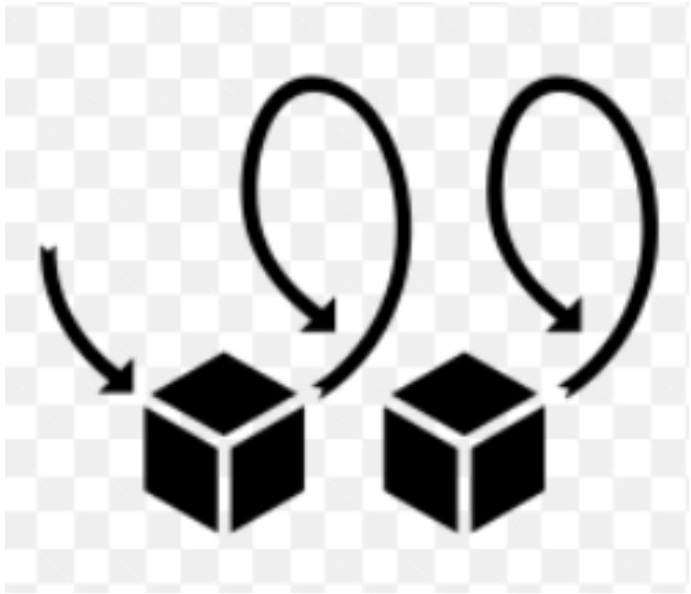
```
nano Dockerfile
```

```
FROM nginx:1.21.6-alpine
RUN sed -i 's/nginx/BRAVO !!!!!/g' /usr/share/nginx/html/index.html
EXPOSE 80
```

❑ Pousser la modification sur la forge :

```
git add . && git commit -m "modification de l'application" && git push
```

9 Vérification CI



Tester le produit livre dans Docker Hub

- ❑ Ouvrir un terminal et se placer dans le dossier de travail :

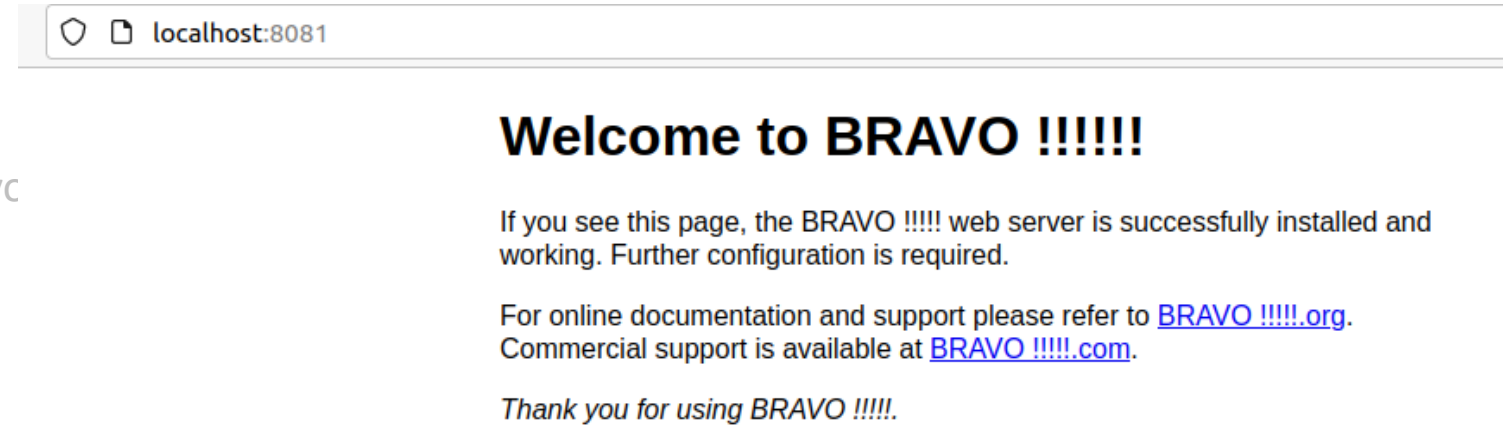
cd monTP

- ❑ Lancer l'image de l'espace dockerhub :

docker run -d --name monserver -p 8081:80 <vc
dockerhub>/monAppliWeb:1.0

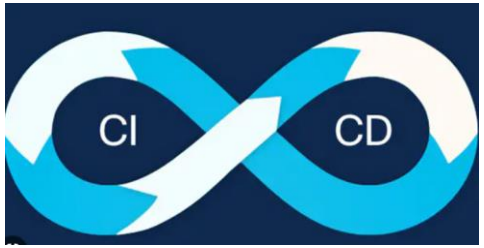
- ❑ Ouvrir le navigateur :

http://localhost:80801



10 Visualisation de l’exécution du pipeline Jenkins

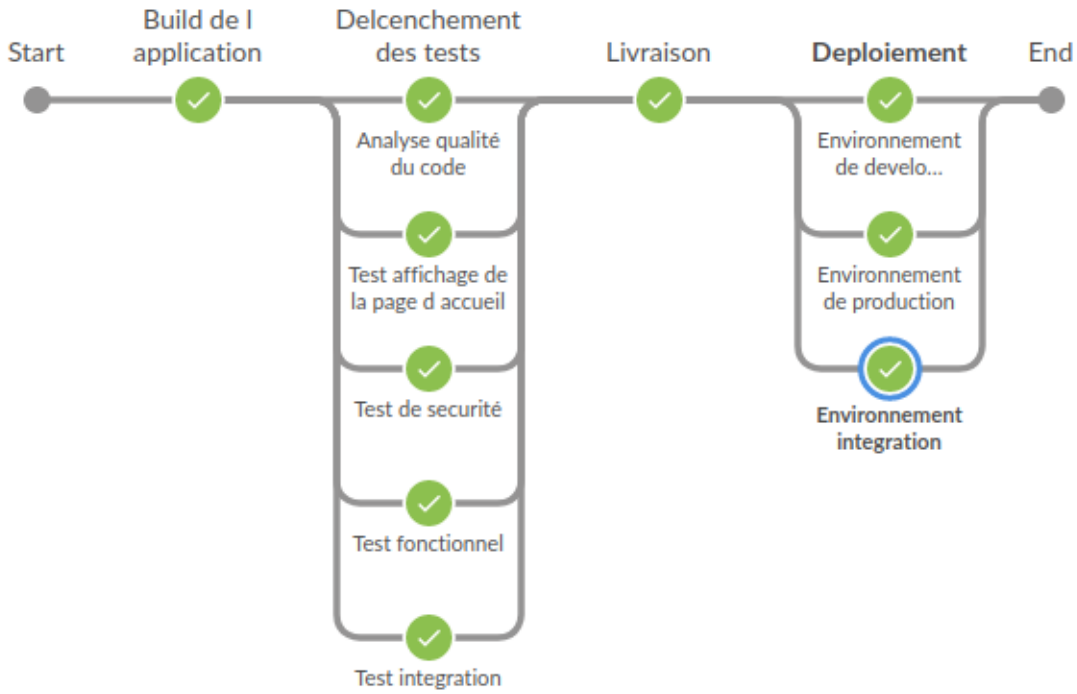
Afficher la liste des pipelines



- ❑ Revenir dans le navigateur :

http://localhost:8080

- ❑ Sélectionner **Mes Vues** à partir du **Tableau de bord**
- ❑ **Afficher le dernier lancement (build) du pipeline**
- ❑ Sélectionner le **Nom du projet** correspondant
- ❑ Sélectionner le dernier job lancé dans l’**historique des builds** (numéroté avec #<numéro>)
- ❑ Ouvrir **Open Blue Ocean**



Synthèse workflow CI/CD