Partie 1:

Comme Jonas, Vagrant ne fonctionnait déjà pas lors de notre cours, c'est pourquoi nous avons travaillé sur une instance AWS: ec2-user@ip-172-31-3-135.

Pour répondre à ce lab, voici les étapes suivies:

- Fork du projet sur mon github.
- Clone du projet sur mon instance.
- Création du Dockerfile à la racine du projet:

```
FROM httpd:latest
RUN apt-get update
RUN apt-get install -y git
EXPOSE 80
RUN git clone https://github.com/carkev/static-website-example.git /var/www/html/
```

- Ajout au dépôt git local, puis commit puis push sur le dépôt distant.
- Configuration de Jenkins (admin1 partout pour l'exercice).
- Création de freestyle job pour tester mon avancée, avec création de username/password dans jenkins global pour github et dockerhub, ainsi que le secret contenant la clé de l'API heroku. Dans ce job, je récupère le projet sur mon fork et le build, puis le test avec curl.

- Je lance le build: fonctionne (cf. build ci-dessous):

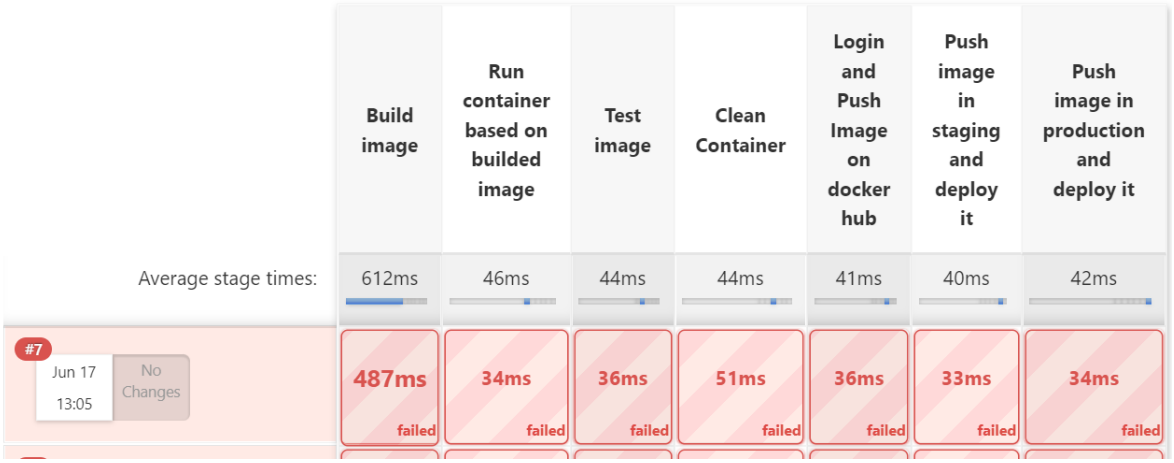
- Puis, je fais le pipeline en gardant les mêmes paramètres: échoue (cf static ci-dessus). Voici le script:

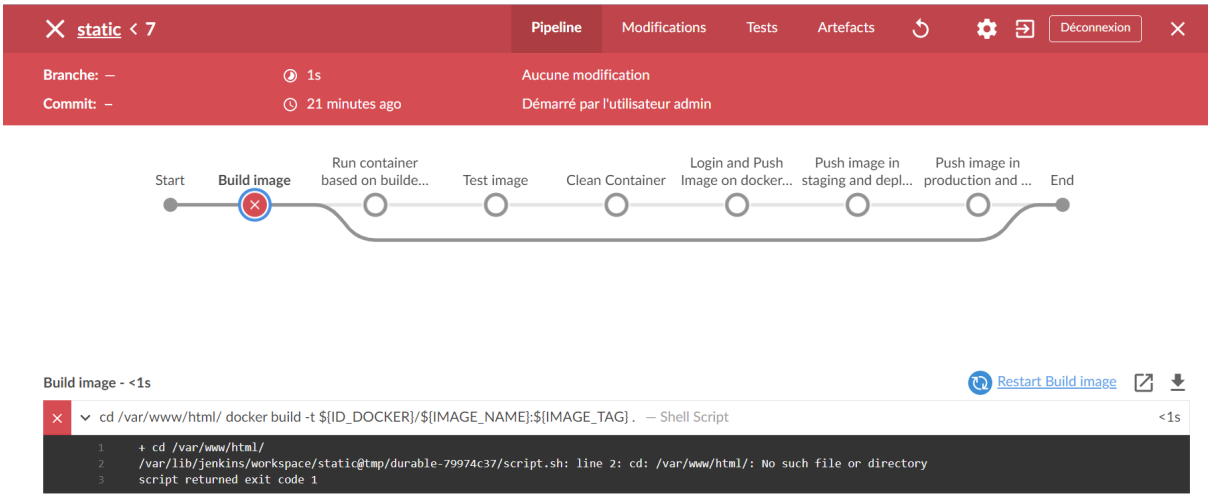
```
pipeline {
    environment {
      ID_DOCKER = "static-website-example"
      IMAGE_NAME = "httpd"
      IMAGE_TAG = "latest"
      STAGING = "${ID_DOCKER}-staging"
      PRODUCTION = "${ID_DOCKER}-production"
    }
    agent none
    stages {
       stage('Build image') {
          agent any
          steps {
            script {
              sh '''
                docker build -t ${ID_DOCKER}/${IMAGE_NAME}:${IMAGE_TAG} .
              '''
            }
          }
       }
       stage('Run container based on builded image') {
          agent any
          steps {
            script {
              sh '''
                docker run --name $IMAGE_NAME -d -p 80:5000 -e PORT=5000
--name ${IMAGE_NAME} ${ID_DOCKER}/${IMAGE_NAME}:${IMAGE_TAG}
                sleep 5
              '''
            }
          }
       }
       stage('Test image') {
          agent any
          steps {
            script {
              sh '''
                curl http://172.31.3.135 | grep -q "Hello world!"
              '''
            }
          }
       }
       stage('Clean Container') {
          agent any
          steps {
```

```
            script {
              sh '''
                docker stop $IMAGE_NAME
                docker rm $IMAGE_NAME
              '''
            }
          }
        }

        stage ('Login and Push Image on docker hub') {
          agent any
          environment {
          DOCKERHUB_PASSWORD = credentials('dockerhub.PWD')
         }
          steps {
            script {
              sh '''
                  echo $DOCKERHUB_PASSWORD | docker login -u $ID_DOCKER
--password-stdin
                  docker push ${ID_DOCKER}/$IMAGE_NAME:$IMAGE_TAG
              '''
            }
          }
         }

        stage('Push image in staging and deploy it') {
          when {
              expression { GIT_BRANCH == 'origin/master' }
            }
         agent any
         environment {
          HEROKU_API_KEY = credentials('heroku_api_key')
        }
         steps {
          script {
            sh '''
              heroku container:login
              heroku create $STAGING || echo "project already exist"
              heroku container:push -a $STAGING web
              heroku container:release -a $STAGING web
            '''
          }
        }
      }



        stage('Push image in production and deploy it') {
```

```
        when {
              expression { GIT_BRANCH == 'origin/master' }
            }
        agent any
        environment {
           HEROKU_API_KEY = credentials('heroku_api_key')
        }
        steps {
           script {
             sh '''
               heroku container:login
               heroku create $PRODUCTION || echo "project already exist"
               heroku container:push -a $PRODUCTION web
               heroku container:release -a $PRODUCTION web
             '''
           }
         }
      }
    }
}
```

Stage View





Partie 2:

- Installer Python3.x
- installer pip si non installé
- Installer pipenv pour créer un environnement virtuel dans lequel installer ansible via Python afin d'éviter les futurs incidents.
- Dans etc/ansible.cfg, pour paramètrer ansible (machines, …)
- Éventuellement, on peut répertorier les machines dans un fichier yaml.
- Dans le playbook, on va pouvoir paramétrer les installation (ici LAMP et Wordpress).
- Enfin, on pourra executer le déploiement tel que configuré dans le playbook avec ansible-playbook.