

MINI PROJET JENKINS (PROJET STATIC-WEBSITE)

Dans le cadre de notre formation DevOps, nous avons été invités à mettre en pratique les connaissances acquises lors des séances de cours et des travaux pratiques sur Jenkins. Ceci en mettant en place un pipeline d'intégration continue pour un site web statique dont le code nous a été fourni.

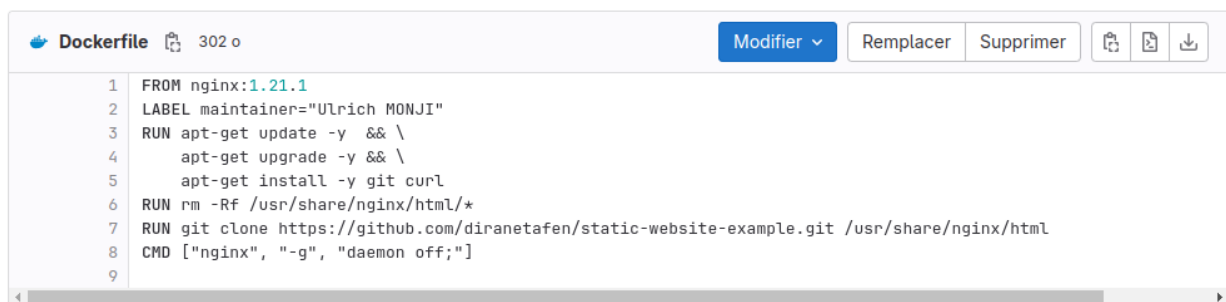
L'objectif de ce rapport est de présenter les différentes étapes de création de ce pipeline, qui vise à construire l'image de notre site, la pousser vers le registre dockerhub, puis créer un environnement de staging et de production en utilisant l'API Eazylab.

Nous déploierons cet environnement dans une machine Docker de l'environnement de notre serveur.

<https://github.com/nzapa-narcisse/mini-projet-jenkins>

I. Création du dockerfile

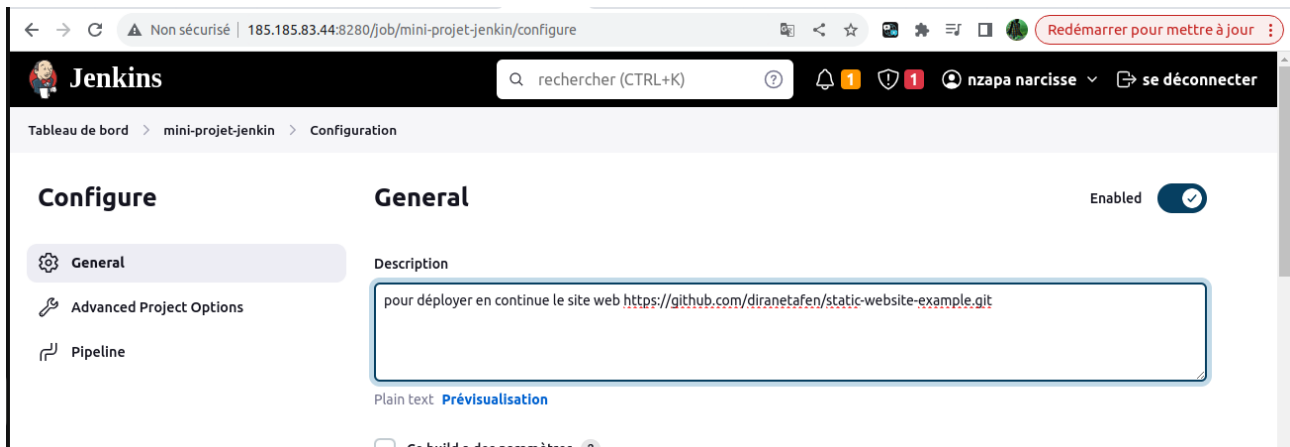
a fin de faciliter le déploiement de notre site web nous avons crée un dockerfile qui nous permettra de containeriser notre application. Nous utilisons "nginx:1.21.1" comme image de base comme le montre la capture ci-dessus.



```
1 FROM nginx:1.21.1
2 LABEL maintainer="Ulrich MONJI"
3 RUN apt-get update -y && \
4     apt-get upgrade -y && \
5     apt-get install -y git curl
6 RUN rm -rf /usr/share/nginx/html/*
7 RUN git clone https://github.com/diranetafen/static-website-example.git /usr/share/nginx/html
8 CMD ["nginx", "-g", "daemon off;"]
9
```

II. Création de notre pipeline sur jenkins

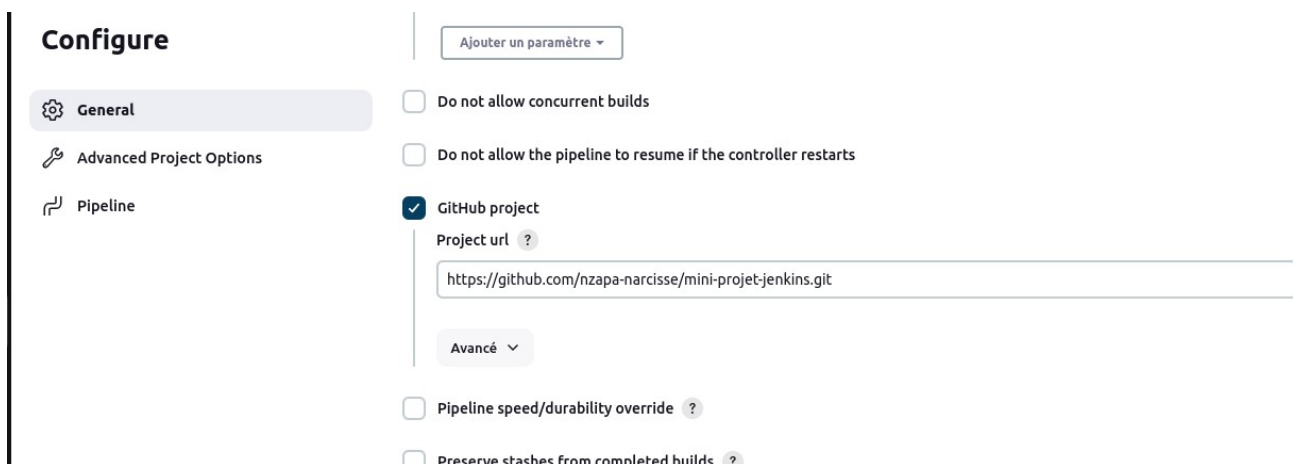
une fois connecté sur jenkins nous commençons par créer un pipeline de type pipeline donc les étapes sont les suivantes :



création paramètre `PORT_EXPOSED` avec pour valeur par défaut 82: ce paramètre définit le port d'écoute externe de notre application.



Ici nous ajoutons l'url de notre projet github afin que jenkins puisse nous générer un lien permettant de consulter notre projet sur github depuis l'interface du pipeline.



Nous choisissons l'option 'scrutation' et nous le paramétrons à chaque minute(5 étoiles). Ainsi chaque minute jenkins va envoyer les check message sur notre projet pour vérifier s'il y a eu de nouveaux commits. Si nouveaux commits le pipeline est lancé automatiquement.

Configure

- General
- Advanced Project Options
- Pipeline

☐ Github hook trigger for SCM polling

☒ Scrutation de l'outil de gestion de version

Planning

⚠️ Voulez-vous vraiment dire "chaque minute" avec l'expression "*****"? Peut-être voulez-vous dire "H*****"?
Aurait été lancé à Saturday, December 16, 2023 at 10:30:38 AM Coordinated Universal Time; prochaine exécution à Saturday, December 16, 2023 at 10:30:38 AM Coordinated Universal Time.

☐ Ignore post-commit hooks

☐ Période d'attente

☐ Déclencher les builds à distance (Par exemple, à partir de scripts)

Nous configurons en fin l'origine de notre code en fournissant url de notre projet.

Configure

- General
- Advanced Project Options
- Pipeline

Pipeline

Definition

Pipeline script from SCM

SCM

Git

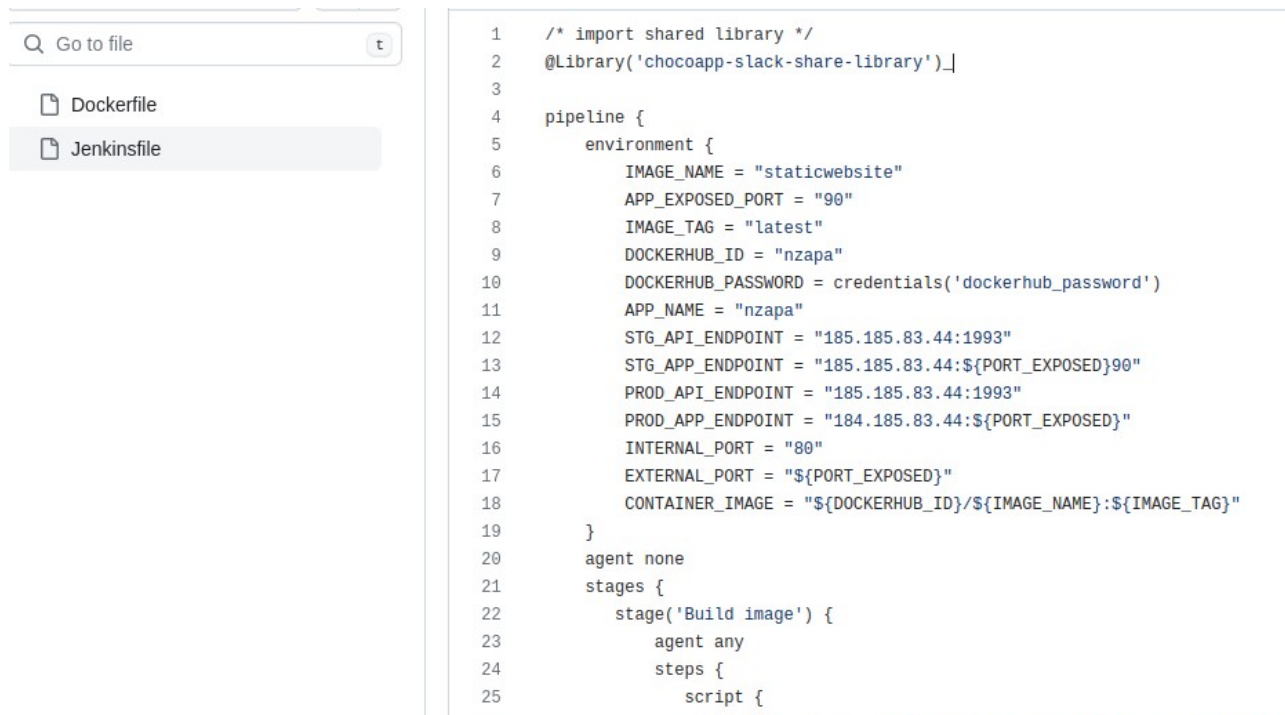
Repositories

Repository URL

https://github.com/nzapa-narcisse/mini-projet-jenkins.git

Credentials

III. Jenkinsfile



l'image ci dessus présente la session variable d'environnement ainsi que les différentes stages de notre pipeline.

Variable d'environnement :

EAZYLABS_IP: l'adresse Ip de notre serveur de déploiement API eazylab

APP_NAME: Nom de notre application

DOCKER_PASSWORD: les identifiant de notre compte dockerhub que nous avons crée sur jenkins

STG_API_ENDPOINT: url api eazylab déployé sur notre serveur 185.185.83.44

STG_APP_ENDPOINT: url de déploiement de notre application pour la qualification

PROD_APP_ENDPOINT: url de déploiement de notre application pour la production

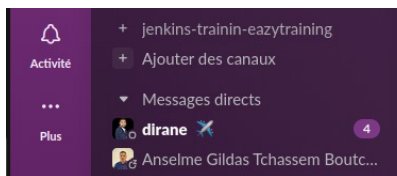
CONTAINER_IMAGE : nom de notre image

Étape de build : à ce niveau le but est de générer l'image de notre application, la déployer, puis faire un curl pour tester son bon fonctionnement.

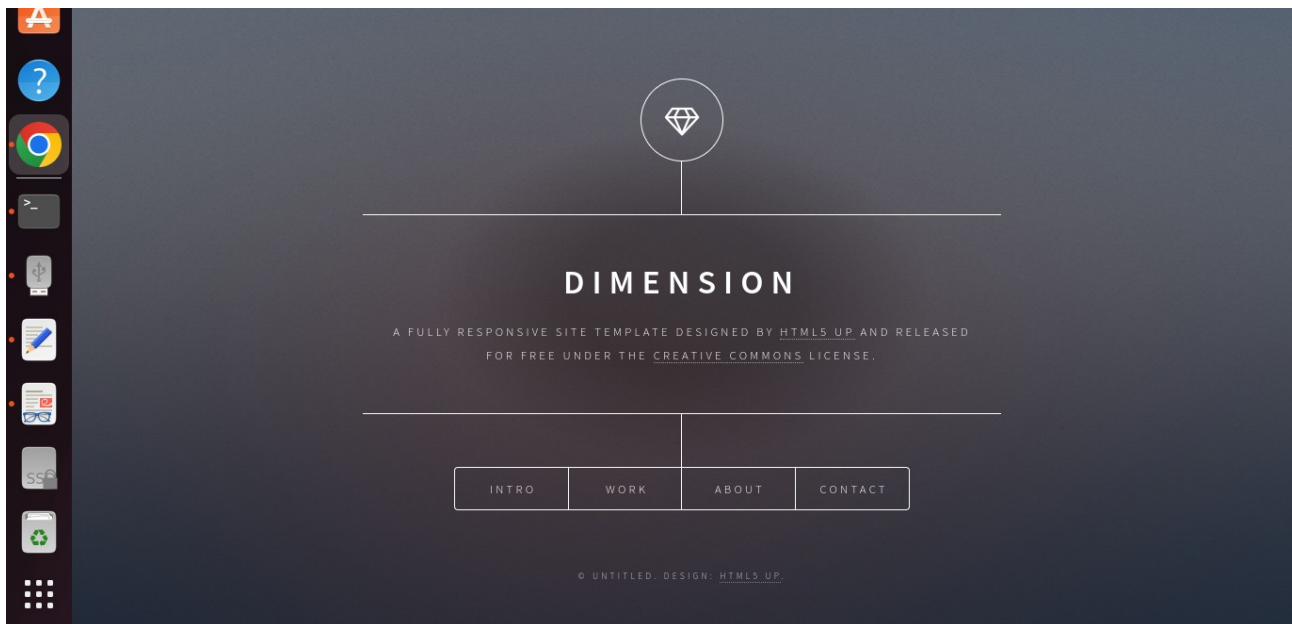
```
agent none
stages {
  stage('Build image') {
    agent any
    steps {
      script {
        sh 'docker build -t ${DOCKERHUB_ID}/${IMAGE_NAME}:${IMAGE_TAG} .'
      }
    }
  }
  stage('Run container based on builded image') {
    agent any
    steps {
      script {
        sh '''
          echo "Cleaning existing container if exist"
          docker ps -a | grep -i $IMAGE_NAME && docker rm -f $IMAGE_NAME
          docker run --name $IMAGE_NAME -d -p $APP_EXPOSED_PORT:$INTERNAL_PORT ${DOCKERHUB_ID}/${IMAGE_NAME}:${IMAGE_TAG}
          sleep 5
        '''
      }
    }
  }
}
```

```
}
stage('Test image') {
  agent any
  steps {
    script {
      sh '''
        curl -v 172.17.0.1:$APP_EXPOSED_PORT | grep -i "Dimension"
      '''
    }
  }
}
stage('Clean container') {
  agent any
  steps {
    script {
      sh '''
        docker stop $IMAGE_NAME
        docker rm $IMAGE_NAME
      '''
    }
  }
}
```

Une fois l'image buildée et testée nous pouvons la pousser sur le dockerhub public comme le montre la capture ci-dessus.



Jenkins APPLI 15 h 26
Nzapa - FAILED: Job 'mini-projet-jenkin [5]' (<http://185.185.83.44:8280/job/mini-projet-jenkin/5/>)
Nzapa - SUCCESSFUL: Job 'mini-projet-jenkin [6]' (<http://185.185.83.44:8280/job/mini-projet-jenkin/6/>) - PROD URL => <http://184.185.83.44:82> , STAGING URL => <http://185.185.83.44:8290>



En conclusion, ce projet nous a permis de mettre en pratique nos compétences en matière de DevOps, en particulier l'utilisation de Jenkins pour la création d'un pipeline d'intégration continue. Nous avons pu constater l'importance d'automatiser les différentes étapes du processus de déploiement, ce qui nous a permis d'améliorer l'efficacité et la fiabilité de nos livraisons de logiciels.

La création du pipeline nous a également permis de mieux comprendre l'architecture d'un système d'intégration continue, ainsi que les concepts clés tels que la construction d'images Docker, le déploiement d'environnements de staging et de production.

En utilisant l'API Eazylab, nous avons pu déployer notre environnement de manière rapide et reproductible, en profitant des avantages offerts par les conteneurs Docker.

Ce projet nous a donné l'occasion de renforcer nos compétences pratiques en matière de DevOps et de mieux comprendre l'importance de l'intégration continue dans le cycle de développement logiciel.

Nous sommes convaincus que les connaissances et l'expérience acquises lors de ce projet seront précieuses pour notre future carrière dans le domaine du développement logiciel et de l'ingénierie DevOps.