Dockerfile:  
FROM postgres:14.1-alpine

ENV POSTGRES\_DB=db \

   POSTGRES\_USER=usr \

   POSTGRES\_PASSWORD=pwd

Terminal:

docker build -t my-postgres-db

docker run -d -p 80:80 --name my-postgres-container my-postgres-db

docker network create app-network

docker rm my-postgres-container

docker run -d --name my-postgres-container --network app-network -p 80:80 my-postgres-db

746ade5c291f2e860faf960e3d166f510f9d844c080f734ce125b7b6d508d1b6

docker run -p "8090:8080" --net=app-network --name=adminer -d adminer

* Aller au port

We put in the dockerfile:  
FROM postgres:14.1-alpine

ENV POSTGRES\_DB=db \

   POSTGRES\_USER=usr \

   POSTGRES\_PASSWORD=pwd

COPY CreateScheme.sql /docker-entrypoint-initdb.d/

COPY InsertData.sql /docker-entrypoint-initdb.d/

docker rm my-postgres-container

docker build -t my-postgres-db .

docker run -d --name my-postgres-container --network app-network -v /data\_dir:/data my-postgres-db

public class Main {

public static void main(String[] args) {

System.out.println("Hello World!");

}

}

javac Main.java

docker build -t my-postgres-db .

docker run -d --name my-postgres-container --network app-network -v /data\_dir:/data my-postgres-db

docker run --rm my-postgres-db



# Première étape : Compilation

FROM eclipse-temurin:17-jdk-alpine AS build

# Définir le répertoire de travail

WORKDIR /usr/src/myapp

# Copier le fichier source Java dans le conteneur

COPY Main.java .

# Compiler le fichier Java

RUN javac Main.java

# Deuxième étape : Exécution

FROM eclipse-temurin:17-jre-alpine

# Copier le fichier compilé de l'étape précédente

COPY --from=build /usr/src/myapp/Main.class .

# Commande pour exécuter le programme Java

CMD ["java", "Main"]

# Étape de build

FROM maven:3.8.6-amazoncorretto-17 AS myapp-build

ENV MYAPP\_HOME /opt/myapp

WORKDIR $MYAPP\_HOME

COPY pom.xml .

COPY src ./src

RUN mvn package -DskipTests

# Étape de run

FROM amazoncorretto:17

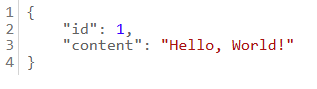
ENV MYAPP\_HOME /opt/myapp

WORKDIR $MYAPP\_HOME

COPY --from=myapp-build $MYAPP\_HOME/target/\*.jar $MYAPP\_HOME/myapp.jar

ENTRYPOINT ["java", "-jar", "myapp.jar"]

docker build -t my-app .

docker run -p 8081:8080 --name my-spring-boot-app-container my-app

spring:

  jpa:

    properties:

      hibernate:

        jdbc:

          lob:

            non\_contextual\_creation: true

    generate-ddl: false

    open-in-view: true

  datasource:

    url: jdbc:postgresql://my-postgres-container:5432/db

    username: usr

    password: pwd

    driver-class-name: org.postgresql.Driver

management:

 server:

   add-application-context-header: false

 endpoints:

   web:

     exposure:

       include: health,info,env,metrics,beans,configprops

docker run -p 8084:8080 --network app-network --name my-app2 simple-api-student

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

docker run --name my-running-app -p 8091:80 --network app-network my-apache2

<VirtualHost \*:80>

ProxyPreserveHost On

ProxyPass / http://my-spring-boot-app-container:8080/

ProxyPassReverse / http://my-spring-boot-app-container:8080/

</VirtualHost>

LoadModule proxy\_module modules/mod\_proxy.so

LoadModule proxy\_http\_module modules/mod\_proxy\_http.so

docker run --name my-running-app -p 8091:80 -d --network app-network my-running-app

Une image contenant texte, capture d’écran, Police, blanc

Description générée automatiquement

<VirtualHost \*:80>

    ProxyPreserveHost On

    ProxyPass / http://${BACKEND\_host}:${BACKEND\_port}/

    ProxyPassReverse / http://${BACKEND\_host}:${BACKEND\_port}/

</VirtualHost>

LoadModule proxy\_module modules/mod\_proxy.so

LoadModule proxy\_http\_module modules/mod\_proxy\_http.so

Docker\_compose.yml:

version: '3.7'

services:

    API:

        container\_name: API

        image: simple-api-student:latest

        environment:

            DB\_host: my-postgres-container

            DB\_port: 5432

            DB\_name: db

            DB\_user: usr

            DB\_mdp: pwd

        networks:

            - app-network

        depends\_on:

            - database

    database:

        container\_name: my-postgres-container

        image: my-postgres-db:latest

        environment:

            POSTGRES\_DB: db

            POSTGRES\_USER: usr

            POSTGRES\_PASSWORD: pwd

        volumes:

            - db-data:/var/lib/postgresql/data

        networks:

            - app-network

    server:

        container\_name: server

        image: my-running-app:latest

        environment:

            BACKEND\_host: API

            BACKEND\_port: "8080"

        ports:

            - "8091:80"

        networks:

            - app-network

        depends\_on:

            - API

networks:

    app-network:

volumes:

    db-data:

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Publication:

docker tag my-running-app guillaume225/my-running-app:1.0

docker tag my-postgres-db guillaume225/my-postgres-db:1.0

tag simple-api-student guillaume225/simple-api-student:1.0

docker login -u guillaume225 -p dckr\_pat\_5IObyxMP3LiS1ylCmpTM6ri8C6o

docker push guillaume225/my-postgres-db:1.0

docker push guillaume225/my-running-app:1.0

docker push guillaume225/simple-api-student:1.0

# TP2

2-2 Document your Github Actions configurations.

name: CI devops 2024

on:

  push:

    branches:

      - main

      - develop

  pull\_request:

    branches:

      - main

      - develop

jobs:

  test-backend:

    runs-on: ubuntu-22.04

    steps:

      # Checkout your GitHub code using actions/checkout@v2.5.0

      - uses: actions/checkout@v2.5.0

      # Set up JDK 17 using actions/setup-java@v3

      - name: Set up JDK 17

        uses: actions/setup-java@v3

        with:

          distribution: 'temurin'

          java-version: '17'

      # Change directory to where the pom.xml file is located and build/test with Maven

      - name: Build and test with Maven

        working-directory: TP1/API/simple-api-student-main

        run: mvn clean verify

name: CI devops 2024

on:

  push:

    branches:

      - main

      - develop

  pull\_request:

    branches:

      - main

      - develop

jobs:

  test-backend:

    runs-on: ubuntu-22.04

    steps:

      # Checkout your GitHub code using actions/checkout@v2.5.0

      - uses: actions/checkout@v2.5.0

      # Set up JDK 17 using actions/setup-java@v3

      - name: Set up JDK 17

        uses: actions/setup-java@v3

        with:

          distribution: 'temurin'

          java-version: '17'

      # Change directory to where the pom.xml file is located and build/test with Maven

      - name: Build and test with Maven

        working-directory: TP1/API/simple-api-student-main

        run: mvn clean verify

  build-and-push-docker-image:

    needs: test-backend

    runs-on: ubuntu-22.04

    steps:

      - name: Checkout code

        uses: actions/checkout@v2.5.0

      - name: Log in to Docker Hub

        uses: docker/login-action@v2

        with:

          username: ${{ secrets.DOCKERHUB\_USERNAME }}

          password: ${{ secrets.DOCKERHUB\_TOKEN }}

      - name: Build image and push backend

        uses: docker/build-push-action@v3

        with:

          context: ./TP1/API/simple-api-student-main

          tags: ${{ secrets.DOCKERHUB\_USERNAME }}/simple-api-student:latest

          push: ${{ github.ref == 'refs/heads/main' }}

      - name: Build image and push database

        uses: docker/build-push-action@v3

        with:

          context: ./TP1

          tags: ${{ secrets.DOCKERHUB\_USERNAME }}/my-postgres-db:latest

      - name: Build image and push httpd

        uses: docker/build-push-action@v3

        with:

          context: ./TP1/HTTP

          tags: ${{ secrets.DOCKERHUB\_USERNAME }}/my-running-app:latest

name: CI devops 2024

on:

  push:

    branches:

      - main

      - develop

  pull\_request:

    branches:

      - main

      - develop

jobs:

  test-backend:

    runs-on: ubuntu-22.04

    steps:

      # Checkout your GitHub code using actions/checkout@v2.5.0

      - uses: actions/checkout@v2.5.0

      # Set up JDK 17 using actions/setup-java@v3

      - name: Set up JDK 17

        uses: actions/setup-java@v3

        with:

          distribution: 'temurin'

          java-version: '17'

      # Change directory to where the pom.xml file is located and build/test with Maven

      - name: Build and test with Maven

        working-directory: TP1/API/simple-api-student-main

        run: mvn clean verify

  build-and-push-docker-image:

    needs: test-backend

    runs-on: ubuntu-22.04

    steps:

      - name: Checkout code

        uses: actions/checkout@v2.5.0

      - name: Login to Docker Hub

        run: docker login -u ${{ secrets.DOCKERHUB\_USERNAME }} -p ${{ secrets.DOCKERHUB\_TOKEN }}

      - name: Build image and push backend

        uses: docker/build-push-action@v3

        with:

          context: ./TP1/API/simple-api-student-main

          tags: ${{ secrets.DOCKERHUB\_USERNAME }}/simple-api-student:latest

          push: ${{ github.ref == 'refs/heads/main' }}

      - name: Build image and push database

        uses: docker/build-push-action@v3

        with:

          context: ./TP1

          tags: ${{ secrets.DOCKERHUB\_USERNAME }}/my-postgres-db:latest

          push: ${{ github.ref == 'refs/heads/main' }}

      - name: Build image and push httpd

        uses: docker/build-push-action@v3

        with:

          context: ./TP1/HTTP

          tags: ${{ secrets.DOCKERHUB\_USERNAME }}/my-running-app:latest

          push: ${{ github.ref == 'refs/heads/main' }}

**Key:**keytp2

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

TP3

Id\_rsa contains my key ssh

chmod 400 TP3/id\_rsa

ssh -i TP3/id\_rsa [centos@guillaume.lecornec.takima.cloud](mailto:centos@guillaume.lecornec.takima.cloud)

setup.yml

all:

 vars:

   ansible\_user: centos

   ansible\_ssh\_private\_key\_file: ~/.ssh/id\_rsa

 children:

   prod:

     hosts: guillaume.lecornec.takima.cloud

To manually connect to the server:  
ssh -i ~/.ssh/id\_rsa [centos@guillaume.lecornec.takima.cloud](mailto:centos@guillaume.lecornec.takima.cloud)  
  
To ping the server:  
ansible all -i inventories/setup.yml -m ping

You will request your server to get your OS distribution, thanks to the setup module:  
ansible all -i inventories/setup.yml -m setup -a "filter=ansible\_distribution\*"

Earlier you installed Apache httpd server on your machine, let’s remove it:

ansible all -i inventories/setup.yml -m yum -a "name=httpd state=absent" –become

A advances playbook: install\_docker.yml:

- hosts: all

  gather\_facts: false

  become: true

# Install Docker

  tasks:

  - name: Install device-mapper-persistent-data

    yum:

      name: device-mapper-persistent-data

      state: latest

  - name: Install lvm2

    yum:

      name: lvm2

      state: latest

  - name: add repo docker

    command:

      cmd: sudo yum-config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo

  - name: Install Docker

    yum:

      name: docker-ce

      state: present

  - name: Install python3

    yum:

      name: python3

      state: present

  - name: Install docker with Python 3

    pip:

      name: docker

      executable: pip3

    vars:

      ansible\_python\_interpreter: /usr/bin/python3

  - name: Make sure Docker is running

    service: name=docker state=started

    tags: docker

ansible-playbook -i setup.yml install\_docker.yml