TP Docker, Dockerfile, Docker-compose

Voici une application qui permet à des utilisateurs de voter entre choix, et qui affiche le résultat des votes.

L'application est décomposée en :

Partie 1

Une application web pour afficher une page pour voter.
Cette application est réalisée avec le Framework Python Flask,

Emplacement de l'application /vote

Pour exécuter l'application, on se basera sur une image Docker Python, ainsi que l'installation des extensions indiquées dans le fichier Requirements.txt.

L'application est démarrée au lancement du conteneur à l'aide de l'extension gunicorn et la commande suivante :
gunicorn <module_name>:<instance_name > -b 0.0.0.0:80 --log-file - --access-logfile - --workers 4 --keep-alive 0.

Cette partie de l'application utilisera une base de données redis non relationnelles définit dans la partie 4

Partie 2

Une application web pour afficher le résultat des votes.
Cette application est réalisée avec le Framework nodejs express,

Emplacement de l'application /result

Pour exécuter l'application, on se basera sur une image Docker node, ainsi que l'installation des extensions indiquées dans le fichier package.json, à l'aide de la commande npm install.

L'application est démarrée au lancement du conteneur à l'aide la commande suivante :
Node server.js

Cette partie de l'application utilisera une base de données de type postgres définit dans la partie 5

Ihab ABADI UTOPIOS

Partie 3

Un worker qui synchronise la base de données non relationnelles et la base de données relationnelles
Cette application est réalisée à l'aide du Framework DotNet et du c#

Emplacement de l'application /worker

Pour exécuter l'application, on se basera sur une image Docker mcr.microsoft.com/dotnet/core/sdk:3.1 pour le build de l'application, et une FROM mcr.microsoft.com/dotnet/core/runtime:3.1 pour l'exécution de l'application.

Les commandes nécessaires pour construire l'application sont :

dotnet restore

RUN dotnet publish -c Release -o /out Worker.csproj

L'application est démarrée au lancement du conteneur à l'aide la commande suivante :
Dotnet worker.dll

Cette partie de l'application se connectera à la fois à la base de données relationnelles et non relationnelles des parties 4, 5

Partie 4

Une base de données non relationnelles de type Redis à partir de l'image Docker redis:5.0-alpine3.10

Partie 5

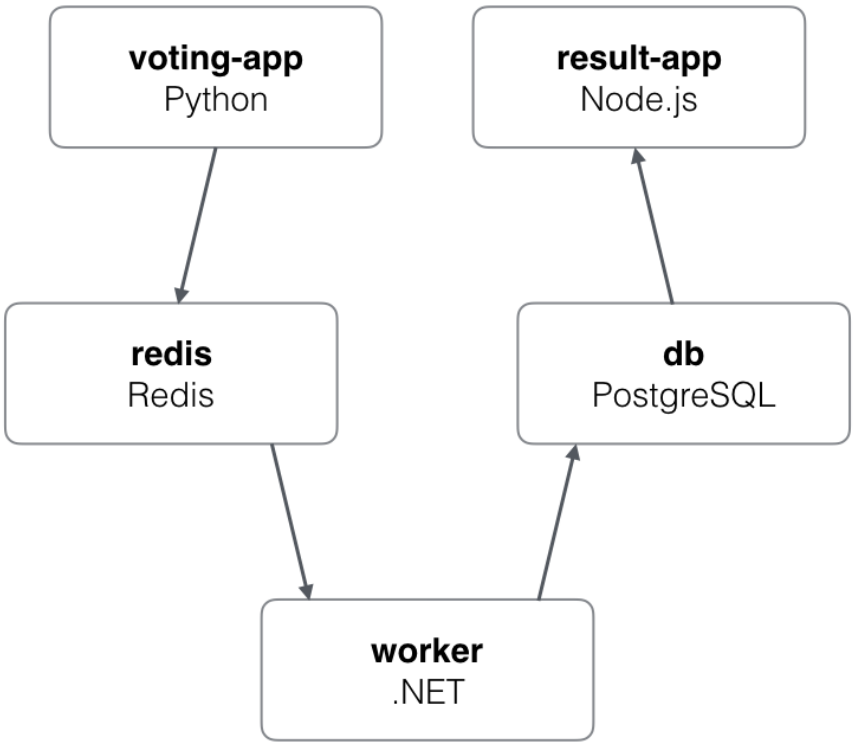
Une base de données relationnelles de types postgres:9.4

Le But du TP est de réalisé, à la fois les Dockerfile nécessaires pour la partie 1, 2 et 3.

Le Docker compose pour exécuter la totalité des services.

Les scripts pour vérifier l'état de santé de chaque service.

On séparera également les parties en deux sous réseaux du Bridge, réseau Front et Back


voting-app
Python
result-app
Node.js
redis
Redis
db
PostgreSQL
worker
.NET