Un système
ITO avec un objet connecté
et falsk_python avec Redis

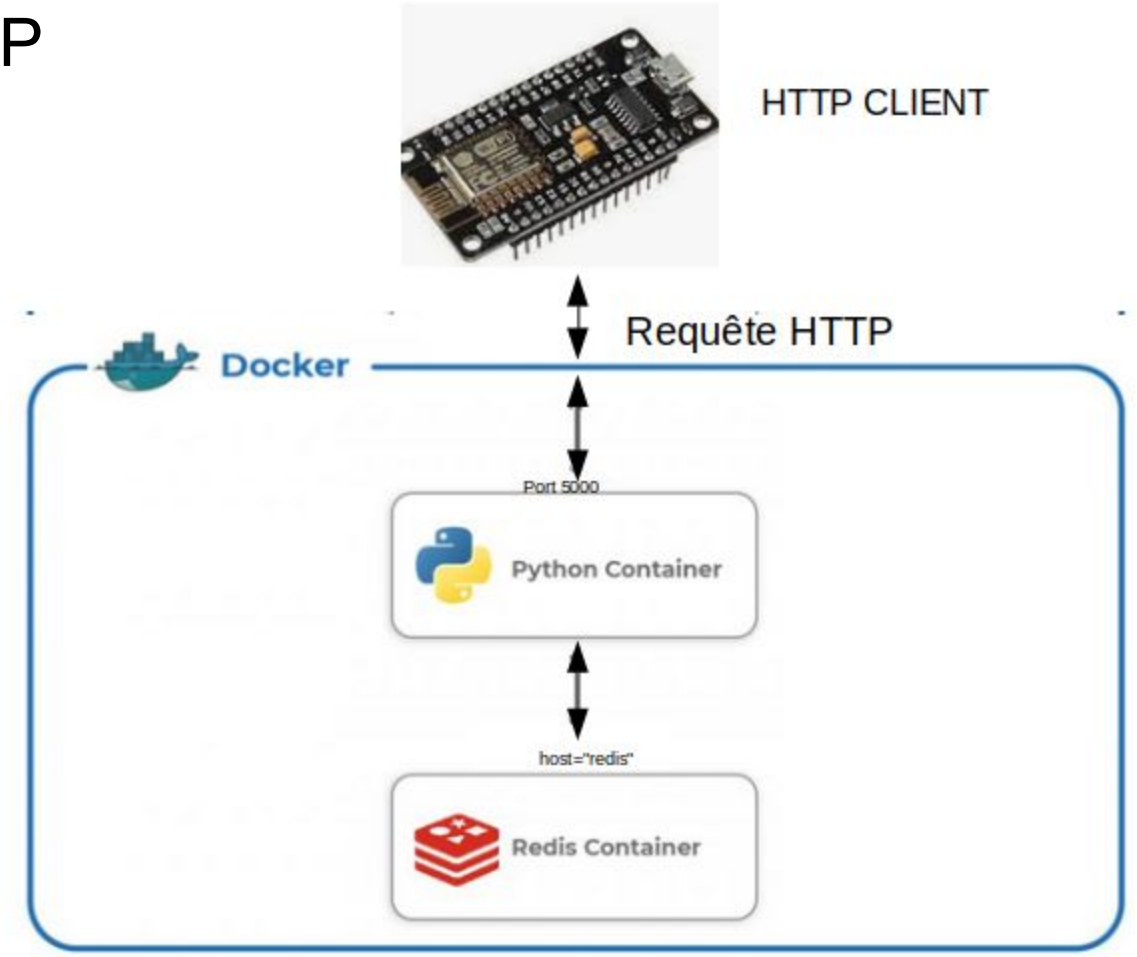
Objetif du TP

Le but de ce TP est de réaliser un système IOT avec un objet connecté , exemple ESP8266, qui envoie des données avec des requête HTTP vers une application WEB, l'application WEB est déployé en utilisant un framework nommé FLASK.

L'application WEB est relé avec un système de stockage légé nommé REDIS.

Les microservices de notre application sont déployé sur des conteneurs docker

Schéma du TP


HTTP CLIENT
Requête HTTP
Docker
Port 5000
Python Container
host="redis"
Redis Container

Framework FLASK

Flask (code source) est un framework Web Python construit avec un petit noyau autour de Werkzeug et Jinja,il est devenu l'un des frameworks d'applications Web Python les plus populaires.

Flask est un framework d'applications Web WSGI léger . Il est conçu pour rendre le démarrage rapide et facile, avec la possibilité d'évoluer vers des applications complexes.

REDIS

Redis (REmote DIctionary Server) est un serveur de dictionnaire distant redistribuer.

Redis est un système de gestion de base de données clé-valeur extensible, très hautes performances.

Il utilise un systèmes de gestion de base de données NoSQL, ce système de SGBD est compatible avec les système IOT.

ESP8266 HTTP CLIENT

LE module esp8266 est utilisé comme un objet connecté client qui va envoyer les données vers un service flask via des requête HTTP.

Pour utiliser l'esp8266 en mode client, on va travailler avec la bibliothèque "<ESP8266HTTPClient.h>".

Les fonctions utilisées sont :

1-Pour configurer le serveur tragique et l'url: http.begin(client,"http://192.168.50.226:5000/");

2-Pour spécifier le format du document envoyé dans la requête: http.addHeader("Content-Type", "application/json");

3-démarrer la connexion et envoyer l'en-tête et le corps HTTP: int httpCode = http.POST("{\"name\":\"ham\"}");

Le fichier Docker-compose:

Docker-Compose est utilisé pour définir et exécuter nos services Docker multi-conteneurs.

Le fichier YAML pour configurer les services de notre application est:

```
version: '3'
services:
  app:
        build: .
        image: flask-redis:1.0
        environment:
        - FLASK_ENV=development
        ports:
        - 5000:5000
        networks:
        - backend
        - frontend
        volumes:
        - dbdata:/database
  redis:
        image: redis:4.0.11-alpine
        networks:
        - backend
        volumes:
        - dbdata:/data
networks:
  backend:
  frontend:
volumes:
  dbdata:
```

Dockerfile:

Pour créer une image Docker de notre application web avec Flask, on utilise le Dockerfile.

Les directives de notre fichier Dockerfile sont:

FROM python:3.7.0-alpine3.8 // l'image de base de notre application

WORKDIR /usr/src/app //pour créer un WOKDIR dans notre conteneur

COPY requirements.txt ./ // copier le fichier requirements.txt dans WORKDIR

RUN pip install --no-cache-dir -r requirements.txt //lancer l'installation de Flask et Redis

COPY . . // copier les fichier "Static, Template et app.py" dans WORKDIR

ENV FLASK_APP=app.py //spécifier comment charger l'application

CMD flask run --host=0.0.0.0 vous // rendre le serveur accessible au public

Application web:

L’application web générer à partir du script python app.py conient les élément suivant:

1-redis = Redis(host="redis", db=0, socket_timeout=5, charset="utf-8", decode_responses=True)

Pour ce connecter à Redis.

2-@app.route('/', methods=['POST', 'GET'])

def index(): .

…….

Pour traiter les requêtes [POST,GET]

3-@app.route('/action')

def do():

…….

Pour afficher la page web en utilisant une page HTML qui se trouve dans un fichier templates

Travaille à faire:

1-Établir une communication ssh entre votre machine et la carte Raspberry PI.

2-Entrer dans le répertoire du projet à la racine du du fichier docker-compose.yaml.

3-Lancer les microservices avec la commande “docker-compose.yaml”.

4-Récupérer l’adresse IP du raspberry et l’utiliser pour acceder à la page .

5-Téléverser le code pour Httpclient dans wemos .

6-Ouvrir la page web “RaspberryIP:5000/action” .