Projet IoT

Le but de ce projet est de mettre en place un système de monitoring d'infrastructure virtualisé sous singularity. Vous pouvez voir un example d'un tel système de monitoring sur solarwind (https://www.solarwinds.com/server-application-monitor/use-cases/dockermonitoring) pour l'application docker qui est similaire à singularity. Pour ceci nous divisons le projet en plusieurs parties

Equipe 1 et 2

Partie 1- Docker et singularity-

L'objectif de cette partie est prendre en main docker et singularity de comprendre comment un container docker est construit. Dans un second temps on prendra en main singularity et on étudiera le mécanisme permettant à singularity de lire des descriptions docker et de les exécuter dans singularity. Nous utiliserons ces APIs pour déployer une cinquantaine de container docker divers (container wordpress, nginx, postgres-SQL, machine ubuntu, etc..) On pourra utiliser l'API de singularity (voir https://singularityhub.github.io/singularity-cli/) pour executer de façon centralisé les containers singularité.

Choses à faire

1) prendre en main docker. Utiliser les nombreux tutoriels existant sur Internet.Comprendre comment marchent les différents containers (wordpress, nginx, postgres-SQL, machine ubuntu, etc..)

2) prendre en main singularity et comprendre comment il peut lire une description python.

4) Écrire un script python permettant de lancer des machines virtuelles sous singularity

Equipe 3- Monitoring et MQTT –

L'objectif de cette partie est de mettre en place un service de broker MQTT sous Mosquitto. Nous développerons dans ce cadre deux broker Mosquitto. Un premier sera sur une raspberry pi et un second à l'intérieur d'un container docker.

Le second objectif de cette partie est de récupérer les données de performances des machines virtuelles docker et de singularity et de diffuser ces données sur un canal MQTT. Cette équipe déploiera aussi un broker MQTT.

Docker offre divers moyens de récupérer les paramètres de performances la commande stats ainsi que l'API rest permettant d'accéder aux mesures de performances. Singularity est construit au dessus de MESOS qui a déjà une API REST (http://mesos.apache.org/documentation/latest/monitoring/).

Choses à faire

1) Voir comment déployer un serveur MOSQUITTO et le rendre disponible aux autres parties du projet

2) Déployer les deux serveurs MOSQUITTO

3) Voir les commande docker stats ainsi que l'API de docker (voir https://docs.docker.com/develop/sdk/)

4) Voir l'API MESOS (http://mesos.apache.org/documentation/latest/monitoring/)

5) Ecrire un script python permettant de récupérer les paramètres de performance de docker et de singularité et de les envoyer au broker MQTT toutes le 10 secondes.

Partie 4- Base de données- Un premier utilisateur du flux de données envoyé dans MQTT est une base de données. Le but de cette partie est la mise en place de la Base de données. Dans ce cadre nous déploierons deux types de base de données. Une base de données MySQL et une base de données NoSQL REDIS. Ces deux bases seront déployés dans des composants dockers.

Choses à faire

1) prendre en main docker. Utiliser les nombreux tutoriels existant sur Internet

2) Comprendre comment marchent les différents containers (wordpress, nginx, postgres-SQL, machine ubuntu, etc..)

3) Voir comment déployer un serveur MySQL et un serveur REDIS dans deux docker séparés.

4) En coopération avec l' équipe de la partie 2 définir la structure des Bases de données à construire pour insérer les données de performance Docker

5) Ecrire un script en coopération avec l'équipe 3 qui s'abonne aux les deux serveurs MOSQUITTO et qui enregistre les mises à jour de performance docker reçu dans la base de données.

Partie 5 et 6 – Framework full stack etr visualisation des performances-

l'objectif de cette partie est de développer un framework full stack sous FLASK permettant au différentes partie du projet d'interagir et d'offrir une API unique d'interaction avec le cluster.

Le second objectif est de fournir sous sous angular un tableau de bord permettant de visualiser les performances des containers docker observé.

Choses à faire

1) Prendre en main angular. Utiliser les nombreux tutoriels existant sur Internet.

2) Demander à l'équipe 1 de vous construire un container avec python et l'installation angular.

3) Construire le tableau de bord. Essayer plusieurs choix de conception possible. Réfléchir à l'ergonomie.

4) En coopération avec l' équipe 4 récupérer les données à afficher de la base de données et afficher celle-ci avec un rafraichissement toute les dix secondes.

5) Mettre en place un mécanisme d'alarme qui vous envoi un mail ou un message quand un des paramètres de performances dépasse une limite.