

30/01/2019

Léo Guilpain – ESIR3 - IoT

TD 7

*« J’atteste que ce travail est original, qu’il indique de façon appropriée tous les emprunts, et qu’il fait référence de façon appropriée à chaque source utilisée »*

Table des matières

[Introduction 2](#_Toc536647745)

[Partie I - Émulation de clients 3](#_Toc536647746)

[Question 01 3](#_Toc536647747)

[Question 02 3](#_Toc536647748)

[Question 03 4](#_Toc536647749)

[Question 04 6](#_Toc536647750)

[Question 05 7](#_Toc536647751)

[Question 06 7](#_Toc536647752)

[Partie II - Monitoring avec Prometheus 10](#_Toc536647753)

[Question 01 10](#_Toc536647754)

[Question 02 10](#_Toc536647755)

[Question 03 11](#_Toc536647756)

[Conclusion 14](#_Toc536647757)

Introduction

Le but de ce TP est de faire en sorte que notre application passe l’échelle. Nous allons étudier la solution « scale-out »

Partie I - Émulation de clients

Question 01

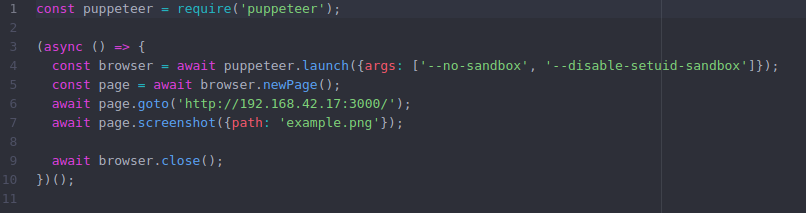


Figure 1 : client screenshot

Ici, nous prenons juste un screenshot de la page d'accueil de mon site. Dès que l’on se rend sur la page <http://192.168.42.17:3000/> une capture d’écran est prise. Ensuite on ferme le navigateur.

Question 02



Figure 2 : Navigation

J’ai créé un fichier naviguation.js qui permet de naviguer entre les pages de mon site. Dans un premier temps je définis la largeur ainsi que la longueur de ma page. Puis je me rends sur mon site.

Enfin avec la fonction “page.click” je spécifie l’ID du bouton sur lequel je veux cliquer. Ce code permet de cliquer sur un album puis de naviguer dans le carrousel. On peut voir « page.waitForSelector », cela permet d’attendre. Tant que la page n’est pas chargée et que l’item correspondant à l’ID n’est pas chargé, on ne fait rien. Les screenshot m’ont permis de savoir si les commandes étaient bien effectuées.



Figure 3 : service web logs

Question 03



Figure 4 : add Album

Cela permet d’ajouter un album. On remplit ici le formulaire en rentrant le nom de l’album et l’image à upload.

On peut aussi ajouter une image :



Figure 5 : add Image

Pour vérifier le bon fonctionnement, il suffit de regarder les logs du service Web. On obtient ceci :

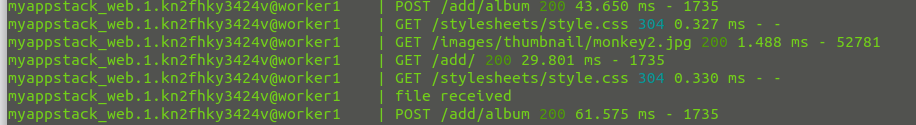


Figure 6 : web log

Le client a bien correctement ajouté un album.

Question 04



Figure 7 : remove Image

On clique sur le bouton “Remove Image” puis on remplit le formulaire avec l’id de l’image que l’on souhaite supprimer.

Ensuite on soumet le formulaire.

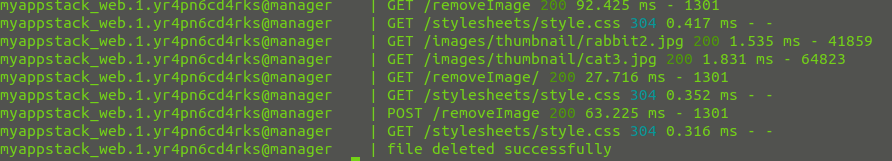


Figure 8 : service web log

On peut voir dans les logs que l’image a été correctement supprimée.

Question 05

Pour effectuer de façon aléatoire une action, il suffit de générer un nombre aléatoire puis de le traiter. On génère un nombre aléatoire entre 0 et 9. Une fois ce nombre généré, on réalise une action en fonction.

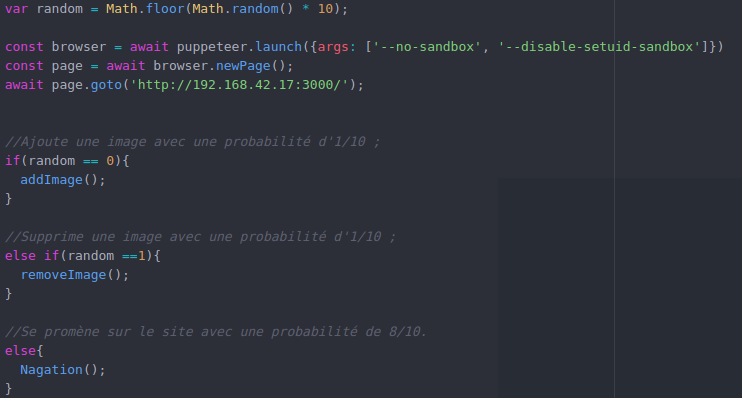


Figure 9 : random

Les fonctions qui sont appelées sont celles générées auparavant.

Question 06

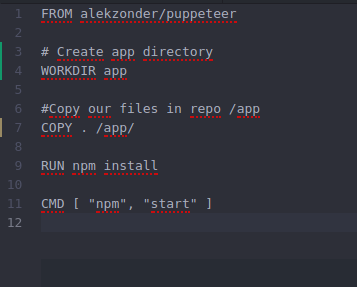
**

Figure 10 : Dockerfile

On récupère l’image « alekzonder/puppeteer » puis on copie tous nos fichiers dans son dossier app. Puis on lance notre fichier « index.js » qui correspond aux actions aléatoires.

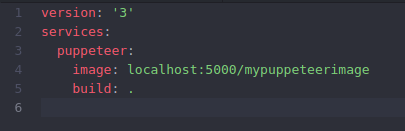
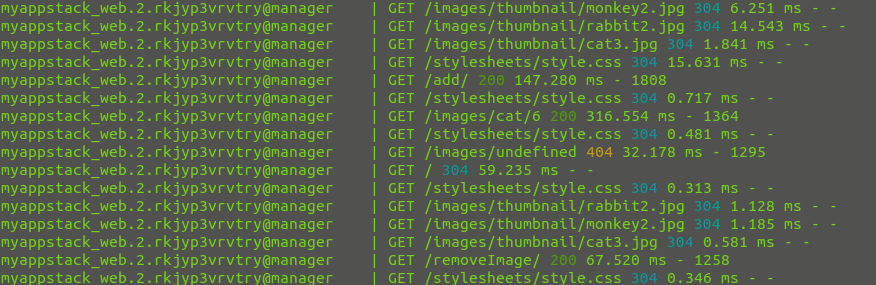
**

Figure 11 : docker-compose

Dans un premier temps, on crée le service manuellement. Pour faire cela, on « build » l’image et on la « push » sur le registre. Ensuite on crée le service. On peut donc voir que tous les services sont bien lancés “sudo docker services ls”

**

Figure 12 : service ls

**

D’après les logs, on voit bien que le fichier index.js a bien été lancé puisqu’il y a eu un ajout d’image mais également une navigation dans les dossiers.

Comme on peut le voir le service est fonctionnel, on l’ajoute dans le stack pour pouvoir tout déployer ensemble.

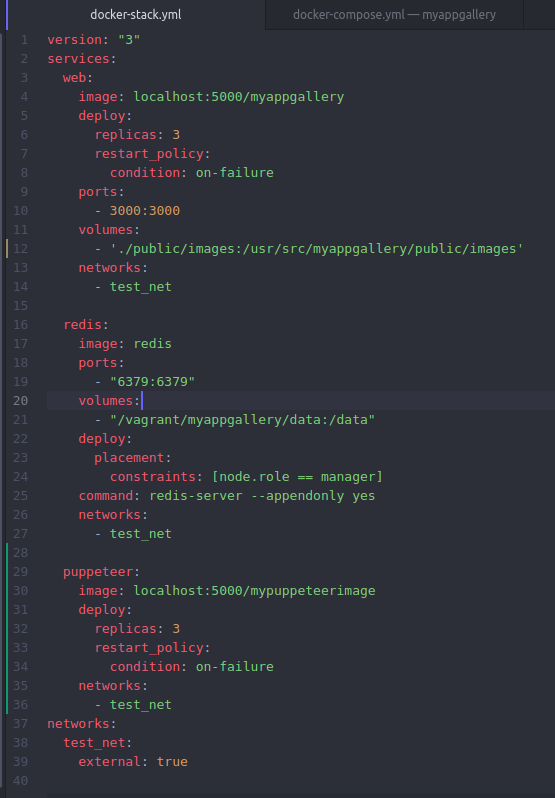
**

Figure 13 : docker-stack

On déploie le stack :

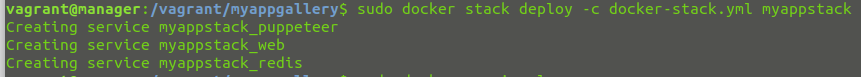


Figure 14 : stack deploy

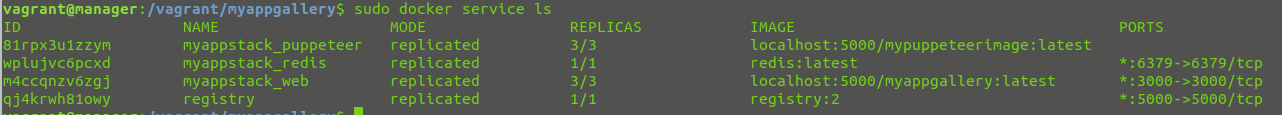


Figure 15 : service ls

Les trois services sont bien correctement lancés.

Si on veut faire des screenshot de l’application alors il faut faire un volume permettant de monter le répertoire screenshot et de récupérer les captures.

**

Figure 16 : visualizer

J’ai également rajouté un « visualizer » pour pouvoir afficher les services sur mes machines.

Je me suis servi de ce site : <https://docs.docker.com/v17.12/get-started/part5/>

Partie II - Monitoring avec Prometheus

Question 01

Pour créer ce fichier, on modifie le fichier “install-docker.yml” en ajoutant ceci :

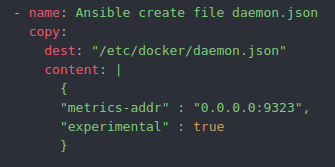


Figure 17 : création du fichier daemon.json

Je me suis servi de ce site : <http://www.mydailytutorials.com/ansible-create-files/>

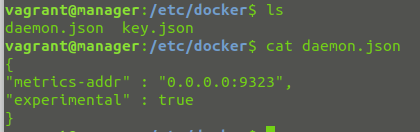


Figure 18 : cat daemon.json

Comme on peut le voir, le fichier à bien été créé sur les machines.

Question 02

Après avoir fait ces modifications, il faut faire « vagrant up –provision » puis il faut relancer docker en effectuant la commande : « etc/init.d/docker restart »

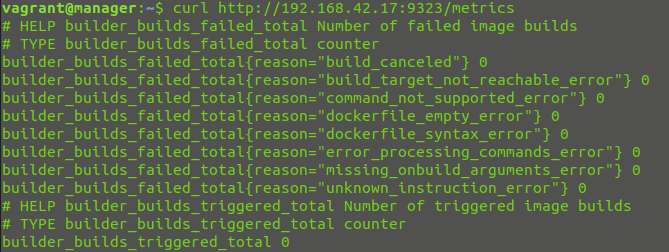


Figure 19 : curl metrics

Le curl fonctionne correctement.

Question 03

Pour commencer, on crée un fichier “prometheus.yml”. Ce fichier nous est donné sur le site : <https://docs.docker.com/config/thirdparty/prometheus/>

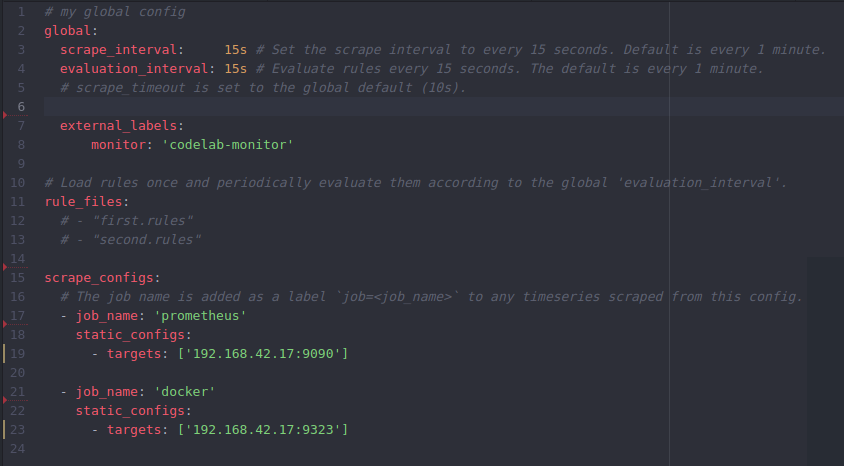


Figure 20 : prometheus.yml

Ensuite, dans notre fichier “docker-install” il suffit de rajouter les lignes suivantes pour copier notre fichier “prometheus.yml” sur nos machines.

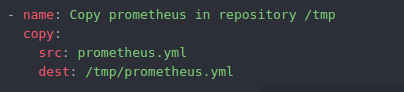


Figure 21 : add prometheus

Enfin, on vérifie que le fichier a bien été copié sur la machine :

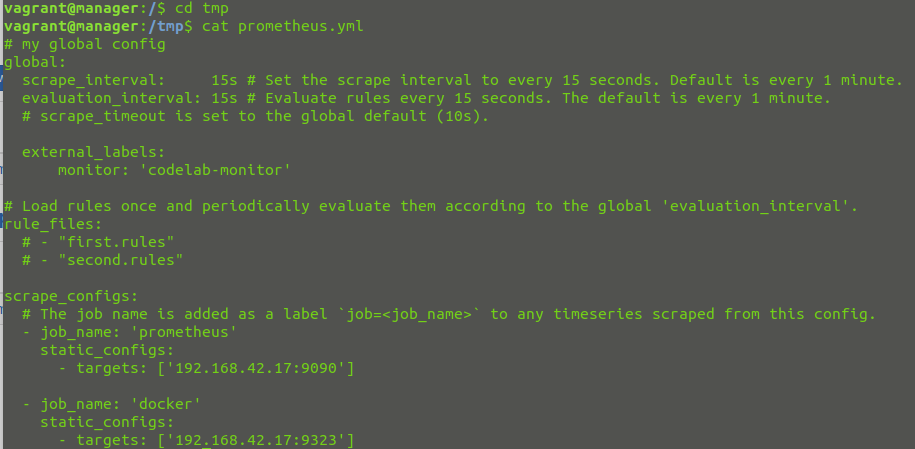


Figure 22 : cat prometheus

Ensuite on crée le service. Ce dernier est bien lancé comme on peut le voir :

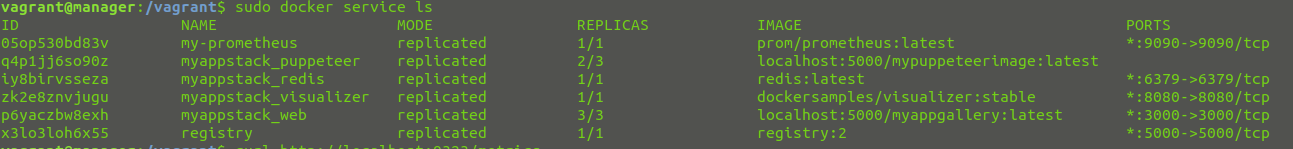


Figure 23 : service ls

Ensuite, on vérifie en se rendant sur le lien : <http://192.168.42.17:9090/targets>

On obtient ceci :

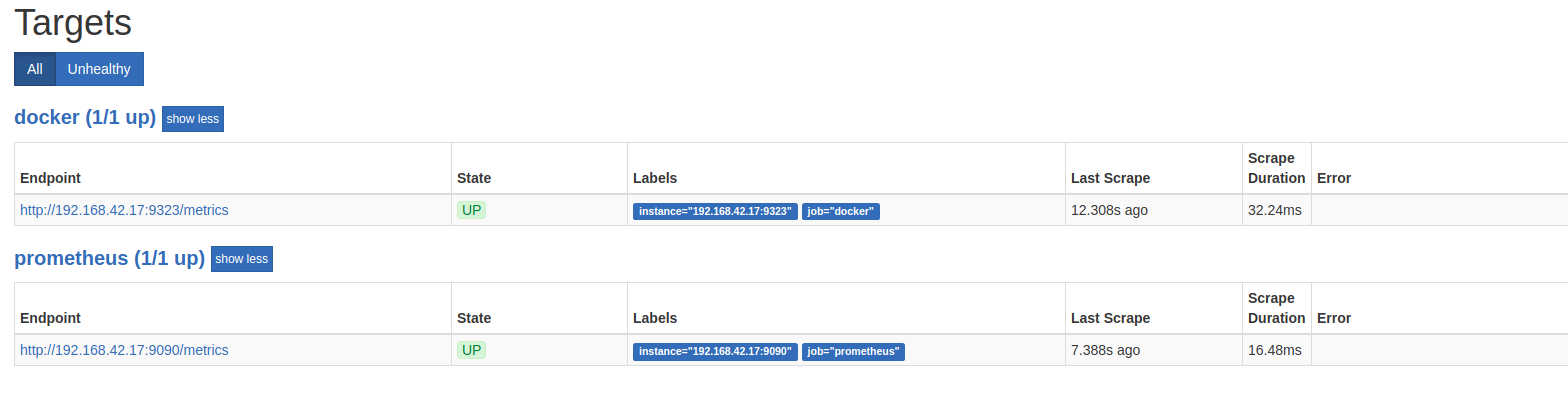


Figure 24 : target

Nous pouvons voir que nous avons bien accès aux métriques du site. On regarde ensuite dans un graphique les données de visite.

Avant de faire un ping, notre graphique ressemble à cela :

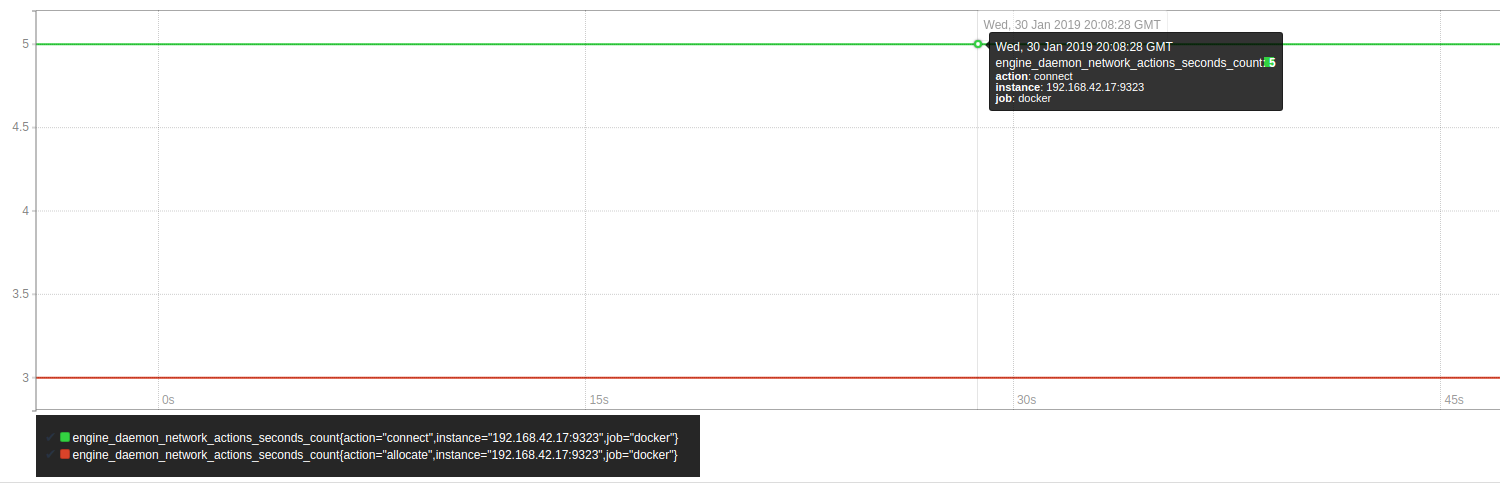


Figure : graph avant ping

Ensuite, on crée un service qui ping le site « docker.com ».

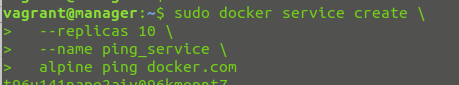


Figure : ping service

Puis on regarde le graphique et on peut voir que le compteur a augmenté.



Figure : graph après ping

Les métriques permettent de fournir un rapport sur différentes actions de l’utilisateur (par ex : trafic par page, source du trafic, durée moyenne de visite). Le plus utilisé est le taux de rebond. Ce taux permet de connaître l'intérêt des utilisateurs lorsqu’ils visitent notre site.

J’ai utilisé ce site : <https://www.rouillier.ca/blog/six-metriques-google-analytics/>

Conclusion

Après avoir émuler des clients qui exécutent des actions sur notre applications, nous avons mis en place la supervision grâce à Prometheus.