

Projet DHCP-Docker

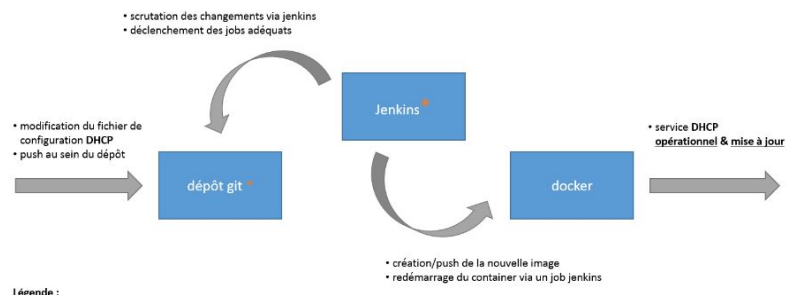
Comprendre l'intérêt et le fonctionnement du DHCP

DHCP prend à sa charge un certain nombre de tâches liées à la configuration du réseau TCP/IP et donne certains avantages notamment la gestion des adresses IP, et la gestion des clients du réseau.

L'intérêt de le développer (de faire un DHCP as code), est de pouvoir tester le système DHCP sans avoir à le virtualiser.

Schéma d'architecture

Comprendre les containers qu'on a besoin dans notre projet. Bien visualiser l'architecture de l'ensemble à l'aide du schéma donné:



Ici nous remarquons qu'on a besoin :

- d'une **instance de serveur git**
- d'une **instance jenkins**
- d'une **instance de gestion dhcp** (dhcpd)

Architecture du projet

Dossier principal du projet

.git	19/11/2020 19:19	Dossier de fichiers	
dhcpd	14/11/2020 11:14	Dossier de fichiers	
git-server	11/11/2020 19:17	Dossier de fichiers	
jenkins_home	02/12/2020 17:11	Dossier de fichiers	
.gitignore	23/10/2020 23:53	Fichier source Git I...	1 Ko
docker-compose.yml	02/12/2020 16:52	Fichier source Yaml	1 Ko
gitserve.dockerfile	02/12/2020 16:46	Fichier source Doc...	2 Ko
jenkins.dockerfile	19/11/2020 22:23	Fichier source Doc...	1 Ko
README.md	19/10/2020 20:18	Fichier source Mar...	1 Ko

Dossier **dhcpd** qui est le dossier de travail de l'utilisateur pour modifier son code.

.git	19/11/2020 22:07	Dossier de fichiers	
dhcpd.conf	19/11/2020 19:13	Fichier CONF	1 Ko
docker-compose.yml	19/11/2020 13:46	Fichier source Yaml	1 Ko
dockerfile	19/11/2020 22:07	Fichier	1 Ko
Jenkinsfile	19/11/2020 20:55	Fichier	1 Ko

Mise en place du dépôt git

Dans le schéma donné dans le sujet, on remarque que le dépôt **git** permet de récupérer les informations sur le serveur DHCP qu'on va devoir configurer. Il va servir de point de départ, il faut donc réussir à virtualiser ce dépôt git à l'aide d'un conteneur.

On doit donc écrire un Dockerfile qu'on a modifié pour répondre au mieux à notre sujet :

Source :

https://linuxhint.com/setup_git_http_server_docker

git-serve : nom du conteneur lié au serveur git

On exécute la commande **mkrepo** afin de créer et initialiser un nouveau repository git sur le git-serve :

```
FROM debian:9
LABEL maintainer="STEINMETZ-PAULUS-WALCH"
# Installation outils
RUN apt update 2>/dev/null
RUN apt-get install -y --no-install-recommends apt-utils 2>/dev/null
RUN apt-get install -y --no-install-recommends software-properties-common 2>/dev/null
RUN apt-get update 2>/dev/null
RUN apt-get install -y --no-install-recommends curl git vim gosu apache2 apache2-utils openssh-server 2>/dev/nullid

# Webserver
RUN a2enmod env cgi alias rewrite
RUN mkdir -p /var/www/git
RUN chown 777 -R /var/www/git/
RUN chown -Rfv www-data:www-data /var/www/git
RUN rm -rf /var/www/html/index.html

# Config git remote
RUN git config --system http.receivepack true
RUN git config --system http.uploadpack true

# Mkrepo git
RUN chown -Rfv www-data:www-data /var/www/git
COPY ./git-server/git.conf /etc/apache2/sites-available/git.conf
COPY ./git-server/git-create-repo.sh /usr/bin/mkrepo
RUN chmod +x /usr/bin/mkrepo

# Conf apache
RUN a2dissite 000-default.conf
RUN a2ensite git.conf
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
ENV APACHE_LOCK_DIR /var/lock/apache2
ENV APACHE_PID_FILE /var/run/apache2.pid



# Utilisateur et clé RSA
RUN useradd -m -d /home/ssh -s /bin/bash -g root -G root,www-data,sudo -u 1000 ssh
RUN echo 'ssh:ssh' | chpasswd
COPY ./git-server/id_rsa.pub /home/ssh/.ssh/authorized_keys
RUN mkdir -p /home/ssh/.ssh
RUN chmod 700 /home/ssh/.ssh
RUN chmod 600 /home/ssh/.ssh/authorized_keys
```

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE
Successfully tagged systeme-dhcp git-serve:latest
user@Latitude-E5450 ~ /CNAM/FIP2/SysAvance/systeme-dhcp
Recreating systeme-dhcp git-serve 1 ... done
user@Latitude-E5450 ~ /CNAM/FIP2/SysAvance/systeme-dhcp
root@ad0e5d283385:/# mkrepo test
Initialized empty Git repository in /var/www/git/test/.git/
root@ad0e5d283385:/# mkrepo test2
Initialized empty Git repository in /var/www/git/test2/.git/
root@ad0e5d283385:/#
```

Vérification sur
localhost:80 que les
repository existe bien

Index of /

Name	Last modified	Size	Description
------	---------------	------	-------------

 test/	2020-10-21 17:59	-	
 test2/	2020-10-21 17:59	-	

Apache/2.4.29 (Ubuntu) Server at localhost Port 80

```
user@Latitude-E5450 ~ /CNAM/FIP2/SysAvance/systeme-dhcp master ± git clone http://localhost/test.git/
Clonage dans 'test'...
warning: Vous semblez avoir cloné un dépôt vide.
```

Service qui sont lancé au niveau du Dockerfile du git-serve, mise en place de 2 services à lancé au lancement du conteneur: le service SSH et service Apache2

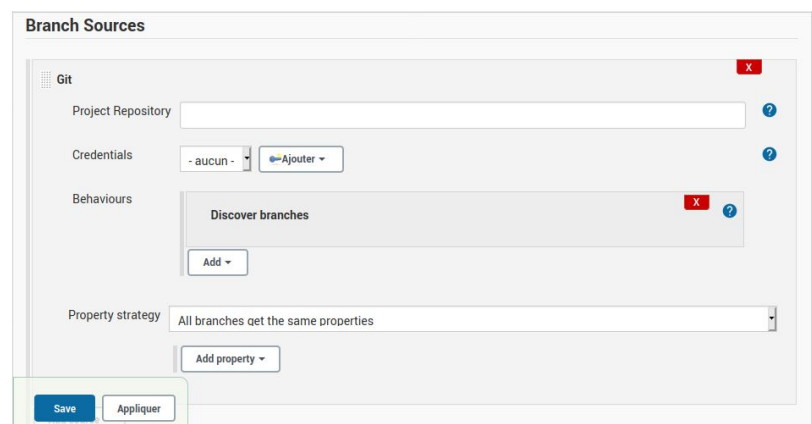
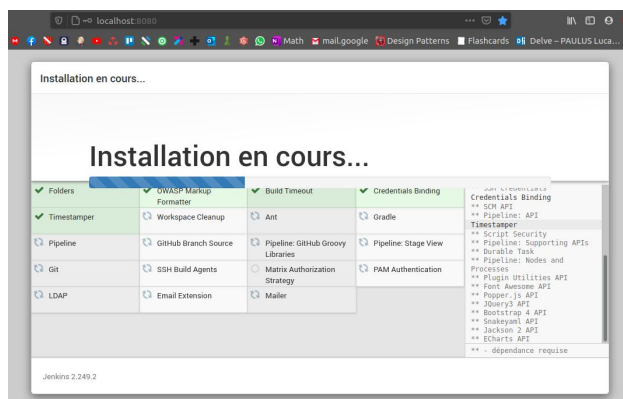
```
CMD gosu root service ssh start && \
    /usr/sbin/apache2ctl -D FOREGROUND
```

Le serveur Git est accessible en SSH ou HTTP, pour le SSH la clé publique est disponible dans le dossier git-serve.

Mise en place de l'image Jenkins

```
services:
  jenkins:
    image: jenkins/jenkins:lts
    privileged: true
    user: root
    ports:
      - 8080:8080
      - 50000:50000
    container_name: jenkins
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/local/bin/docker:/usr/local/bin/docker
```

On fixe des volumes Jenkins constants dans le dockerfile en mappant les ports correctement (le premier étant celui du container host et le deuxième celui du nouveau container). On peut ensuite installer notre serveur git.



Connexion au serveur Git en HTTP :

```
Started by user Docky
[Sat Nov 14 14:45:08 UTC 2020] Starting branch indexing...
> git --version # timeout=10
> git --version # 'git version 2.11.0'
> git ls-remote --symref -- http://172.20.0.2/projet.git/ # timeout=10
Creating git repository in /var/jenkins_home/caches/git-a5bc912dc99020e6312f05d3c338ce41 # timeout=10
> git init /var/jenkins_home/caches/git-a5bc912dc99020e6312f05d3c338ce41 # timeout=10
Setting origin to http://172.20.0.2/projet.git/
> git config remote.origin.url http://172.20.0.2/projet.git/ # timeout=10
Fetching & pruning origin...
Listing remote references...
> git config --get remote.origin.url # timeout=10
> git --version # timeout=10
> git --version # 'git version 2.11.0'
> git ls-remote -h -- http://172.20.0.2/projet.git/ # timeout=10
Fetching upstream changes from origin
> git config --get remote.origin.url # timeout=10
> git fetch --tags --progress --prune -- origin +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking branches...
  Checking branch master
    'Jenkinsfile' found
    Met criteria
  No changes detected: master (still at 4b2a0cf2a85e238ca8590b319328d28ab333e4ca)
Processed 1 branches
[Sat Nov 14 14:45:08 UTC 2020] Finished branch indexing. Indexing took 0.58 sec
Finished: SUCCESS
```

Intégration de docker dans l'image jenkins

```
jenkins:
  build:
    context: .
    dockerfile: jenkins.dockerfile
  privileged: true
  user: root
  ports:
    - "8080:8080"
    - "50000:50000"
  container_name: jenkins
  volumes:
    - ./jenkins_home:/var/jenkins_home
    - /var/run/docker.sock:/var/run/docker.sock
    - /usr/local/bin/docker:/usr/local/bin/docker
  networks:
    default:
      ipv4_address: "172.20.0.3"
```

Nous avons créé un dockerfile personnalisé où nous avons défini l'image Jenkins

Configuration du dhcp et déploiement :

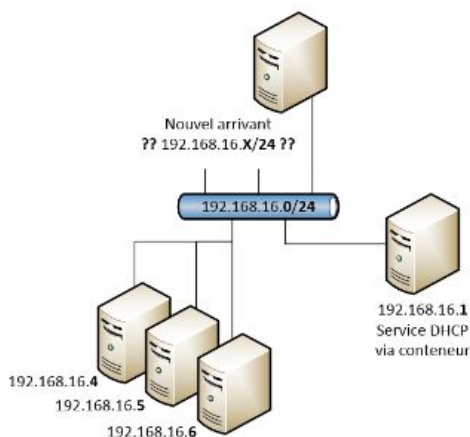
Lors de la mise en place du dhcp l'adresse du réseau va être prise aléatoirement, comme nous avons besoin de communiquer entre nos deux containers nous avons donc écrit notre adresse ip en dur dans le fichier docker-compose.yml

```
version: '3.8'

services:
  dhcpd:
    build: .
    restart: unless-stopped
    networks:
      internal_network:
        ipv4_address: 192.168.16.1

    volumes:
      - ./dhcpd.conf:/etc/dhcp/dhcpd.conf

networks:
  internal_network:
    driver: bridge
```



Architecture du réseau DHCP et sa configuration dans le fichier *dhcp.conf* :

```
# dhcpd.conf
default-lease-time 86400; # Bail de 24H
max-lease-time 172800; # Bail maxi de 48H

# Déclaration d'un réseau
subnet 192.168.16.0 netmask 255.255.255.0 {
  range                  192.168.16.2 192.168.1.254; # Plage IP
  option domain-name-servers 8.8.8.8; # DNS
  option routers          192.168.16.1; # Passerelle
}
```


Nous avons fait une erreur car les deux containers n'étaient pas dans le même réseau donc le repository n'était pas accessible pour Jenkins. Nous avons donc défini un bridge pour permettre l'interconnexion entre les deux containers sans accès de l'extérieur.

```
networks:  
  default:  
    driver: bridge
```

DockerLint: Linter pour aider à repérer les erreurs dans les dockerfile.

1. commande d'installation:

```
[sudo] npm install -g dockerlint
```

2. commande de lancement:

```
dockerlint Dockerfile
```

```
user@Latitude-E5450: ~/CNAM/FIP2/SysAvance/systeme-dhcp [master] dockerfilelint gitserve.dockerfile  
File: gitserve.dockerfile  
Issues: 6  
  
Line 4: RM apt-get install -y apt-utils 2>/dev/null  
Issue Category Title Description  
1 Optimization Consider Consider using a '--no-install-recommends' when 'apt-get' installing packages. This will result in a smaller image size. For more information, see [this blog post](http://blog.replicated.com/2016/02/05/refactoring-a-dockerfile-for-image-size/)  
  
Line 5: RM apt-get install -y software-properties-common 2>/dev/null  
Issue Category Title Description  
2 Optimization Consider Consider using a '--no-install-recommends' when 'apt-get' installing packages. This will result in a smaller image size. For more information, see [this blog post](http://blog.replicated.com/2016/02/05/refactoring-a-dockerfile-for-image-size/)  
  
Line 6: RM apt-get update 2>/dev/null  
Issue Category Title Description  
3 Optimization apt-get update with Use of apt-get update should be paired with rm -rf /var/lib/apt/lists/* in the same layer.  
4 Optimization apt-get update without matching All instances of 'apt-get update' should have the 'apt-get install' commands on the same line to reduce image size.  
apt-get install  
  
Line 7: RM apt-get install -y git via mc sudo gosu apache2-utils openssh-server 2>/dev/null  
Issue Category Title Description  
5 Possible Bug Use Of sudo Is Not Use of 'sudo' is not allowed in a Dockerfile. From the official
```

```
user@Latitude-E5450: ~/CNAM/FIP2/SysAvance/systeme-dhcp [master] dockerfilelint jenkins.dockerfile  
File: jenkins.dockerfile  
Issues: None found 🍀
```

Répartition des tâches

Lucas Paulus	Mise en place de l'architecture et du serveur git.
Yoann Walch	Dockerlint et création du pipeline Jenkinsfile
Baptiste Steinmetz	Rédaction du rapport et écriture du service Dhcpcd

Difficultés rencontrées

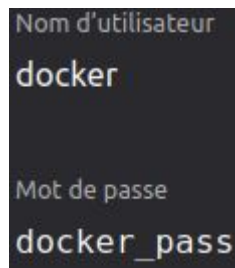
- **Le cache des images Docker** a induit beaucoup d'erreurs, surtout lorsqu'on utilise docker-compose, celui-ci ne rebuild pas automatiquement l'image docker si le Dockerfile a été modifié. (---no-cache notamment) .
- **Docker into Docker** (DinD). Consiste à avoir une instance de Docker dans un conteneur géré par un système hôte. Le DinD pose de nombreux problèmes de bas niveau surtout au niveau de la sécurité, il n'est pas recommandé ([Source](#)) mais dans le cas d'un déploiement local et d'un usage spécifique cela reste fonctionnel.
- **Connexion en SSH avec le serveur Git**, problème de droits sur le dossier à bloquer les commandes *push* du client sur le serveur.

Lancement du Projet

Pour lancer le projet et créer les containers il faut exécuter la commande suivante :

```
docker-compose up -d
```

Login Jenkins

A screenshot of the Jenkins login page. It shows two input fields. The first field is labeled 'Nom d'utilisateur' (Username) and contains the text 'docker'. The second field is labeled 'Mot de passe' (Password) and contains the text 'docker_pass'.

Login User “ssh” sur git-server

Username : ssh

Password : ssh