Cloud et virtualisation

Rapport de projet :

Déploiement d'une solution cloud d'hébergement d'image

Thomas Dumond & Ethan Huret

Table des matières

[Table des matières 2](#_Toc135267720)

[I. Titre 1 3](#_Toc135267721)

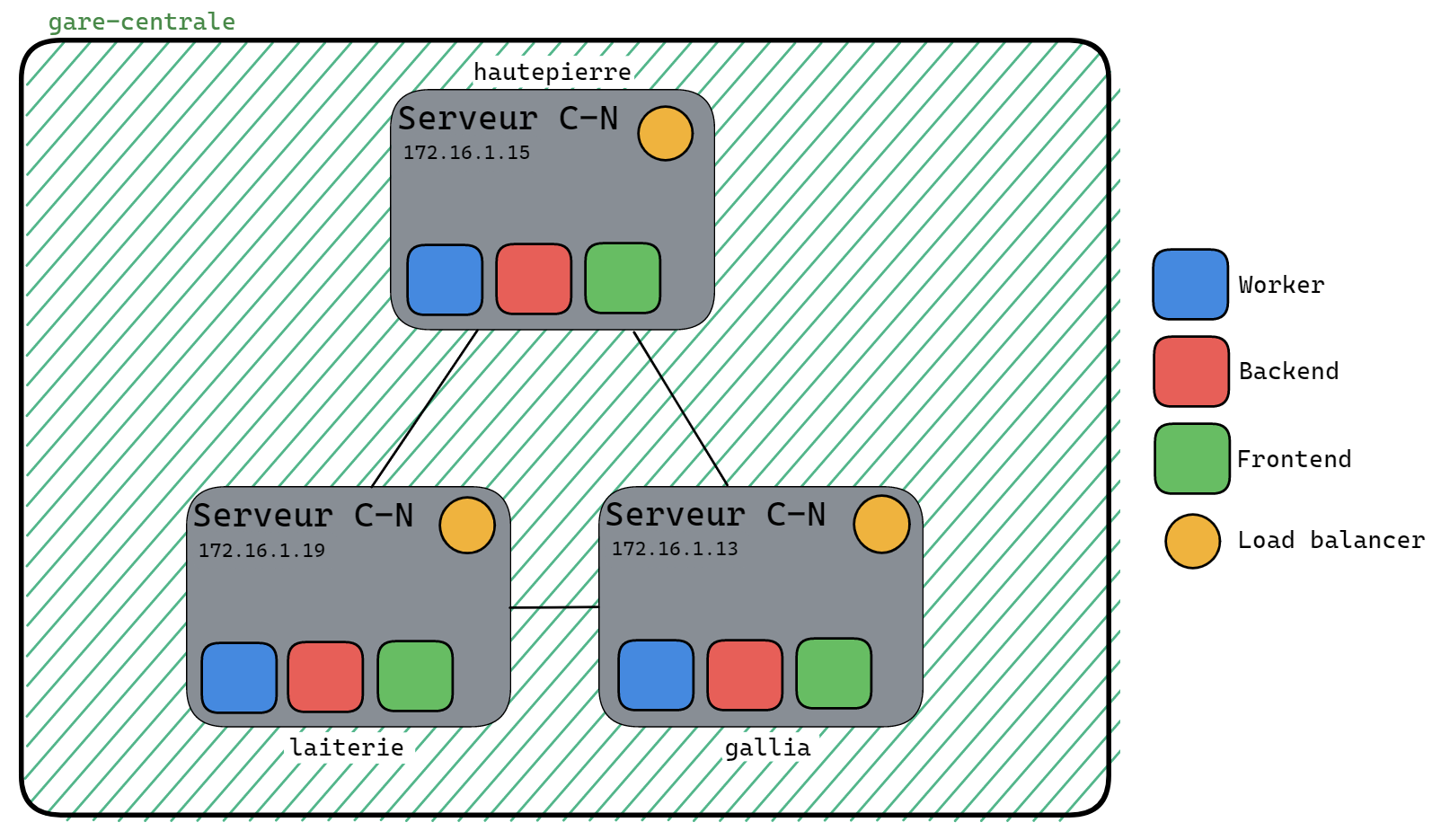
[I.1 Sous-titre 1 3](#_Toc135267722)

1. Déploiement de l'application

I.1 Application à déployer

L'application à déployer correspond à une solution d'hébergement et de redimensionnement d'image en ligne. Elle contient 3 parties : frontend, backend et worker. La partie frontend correspond à l'interface web, où l'utilisateur upload l'image qu'il veut stocker via le protocole http. La partie backend récupère les images uploadées et les stocke sur un système de stockage. Et enfin, la partie worker permet de redimensionner les images.

I.2 Architecture de déploiement

 Voici un exemple d'architecture à 3 nœuds permettant de déployer l'application. Ces nœuds correspondent à des Virtual Machine, qui possèdent une adresse IP (*192.168.70.*) pour accéder à la machine en SSH et une adresse IP (*172.16.1.*) pour communiquer entre elles via des tunnels VXLAN. Toutes ces VM sont configurées en tant que serveur Consul-Nomad dont une sera désigné automatiquement comme leader. Le déploiement des différentes parties de l'application (frontend, backend, worker) se fait via des conteneurs Docker. Ces conteneurs sont ensuite déployés dans des jobs Nomad, qui seront répertoriés et regroupés en tant que service (pour chaque partie de l'application) dans Consul.

Une image contenant texte, capture d’écran, diagramme

Description générée automatiquementDe plus, une adresse IP flottante est configurée afin de recevoir le flux http de l'application sur une des VM, puis ce flux est distribué aux différents conteneurs de la partie frontend via un load balancer HAProxy. Un proxy est configuré dans le réseau afin de rediriger le trafique http de l'URI *https://gare-centrale.100do.se/* vers l'IP flottante *172.16.3.4*.

I.3 Déploiement de l'architecture

Pour déployer cette architecture, nous avons utilisé Ansible afin de définir les différentes configurations des VM et de déployer l'application selon l'architecture précédente. Pour cela un playbook est fourni dans le [Git](https://github.com/EthanAndreas/CloudAppDeployment), ce playbook est construit de la manière suivante :

inventory.yaml : permet de définir les VM avec certaines caractéristiques (adresse IP de l'interface VXLAN…)

playbook.yaml :

Le dossier roles définit

Une image contenant texte, capture d’écran, carte de visite, Police

Description générée automatiquement

II. Déploiement d’une nouvelle version de l’application

II.1 Création de nouvelles images Docker

Lors de la mise à jour de l’application (ie de l’api ou du frontend), il faut commencer par rebuild les images de Docker. Comme cela nécessite des identifiants et cela dépends du registry, nous n’avons pas réalisé de scripts. Ainsi, les commandes à réaliser sont :

* cd web
* docker build -t frontend .
* docker tag frontend <registry>/frontend:<tag>
* docker push <registry>/frontend:<tag>
* cd api
* docker build -t worker .
* docker tag worker <registry>/worker:<tag>
* docker push <registry>/worker:<tag>

II.2 Lancement avec Nomad

On peut ensuite relancer le job Nomad avec la commande Ansible :

ansible-playbook playbook/playbook.yaml -i playbook/inventory.yaml --tags nomad

III. Maintenance planifiée d’un nœud

III. Scalabilité horizontale