# II. Rapport de votre solution technique

## a. Introduction

A l'heure d'aujourd'hui, de nouvelles techniques de déploiement apparaissent. On voit arriver l'IAAS qui est l'Infrastructure As A Code et qui, comme son nom l'indique, permet de construire son infrastructure via des lignes ou des fichiers de codes par le biais d'application tels que Terraform, Ansible, etc... Aujourd'hui, une nouvelle entreprise cherche à se mettre à jours avec les nouvelles technologies pour faciliter le déploiement de son infrastructure en cherchant à recréer son environnement en quelques clic seulement. Elle cherche à monter sa propre registry, ainsi qu'un cluster d'application conteneurisé, le tout, entièrement redéployable à l'identique.

Comment peut-on déployer une infrastructure avec une seule ligne de commande ?
Pour répondre à cette question, nous allons dans un premier temps lister les technologies qu'on va utiliser pour répondre au besoin principal du client, tout en essayant d'améliorer sa demande. Et dans un second temps, on parlera de la technologie utiliser pour faire un déploiement continue, faire les tests, ou encore détruire et recréer son infrastructure.

## b. Répondre au besoin du client

Le client veut pouvoir gérer et stocker toute sa configuration sur un repository privé, nous lui avons donc mis en place un GitHub. D'ici il pourra télécharger, modifier et pousser les modifications avec la validation des changements entre collaborateurs, ce qui permet d'améliorer la qualité du code et d'éviter les problèmes de configuration.
Ensuite, il fallait installer docker sur des machines en clustering afin d'avoir une redondance et d'améliorer la sécurité de la disponibilité des services présent sur ces machines, et avoir un taux de disponibilité supérieur ou égal à 99.9%. Nous avons utilisé la technologie SWARM qui va donc permettre de relier deux ou plusieurs serveurs et en faire un cluster.
De plus, l'entreprise veut également installer une registry. C'est ici qu'elle va stocker les images docker qu'elle aura besoin, et que les machines du cluster vont devoir contacter pour monter les images.
Il faut par la suite superviser les machines, donc nous avons trouver Netdata qui va permettre une supervision des machines ainsi que les conteneurs.
Pour finir, nous nous sommes dit qu'il valait mieux installer un HaProxy devant le cluster afin de faire un LoadBalancing performant pour un coût nul (à part le coût d'une machine). On aurait voulu en mettre deux et avoir un cluster d'HaProxy avec un Heartbeat d'installer pour voir si le second nœud est toujours vivant. Mais nous n'avons pas eu le temps pour le faire.

### c. Déployer automatiquement l'infrastructure

Maintenant qu'on a choisis toutes les technologies pour avoir une infrastructure correcte, il faut décider avec quelle technologie nous allons livrer cette infra en production.
On s'est donc poser la question, quelle technologie est la meilleure pour la compréhension, la plus facile à utiliser et à se former dessus. Avec laquelle on peut effectuer des tests facilement et à tout moment avant de livrer en production
Nous avons donc pensé à Ansible qui est un outil de gestion et d'automatisation. Il va permettre de livrer via une ligne de commande à écrire, et pour faire des tests nous avons juste besoin d'ajouter « --check » à cette même ligne. Le code est humainement compréhensible, c'est-à-dire que même quelqu'un qui ne s'y connait pas énormément peut comprendre un minimum ce qu'il fait.
Dans ce projet, Ansible va donc déployer les clé SSH des utilisateurs grâce à des fichiers de variables (photo1). Les clés vont être déployer après que les utilisateurs et leurs groupes soient créer.
Ansible va ensuite installer dans cet ordre via des « roles » : docker et swarm sur les machines en cluster, docker sur la machine en standalone (registry), l'HaProxy, la registry, et un conteneur de test et de supervision Netdata en dernier.
Un rôle est composé de plusieurs tasks qui vont être exécuter afin d'installer des composant et des fichiers de configurations (photo2) et peuvent être combiné à des variables (photo 3).
Si l'exécution d'ansible se passe correctement et sans erreurs, nous auront un résultat avec des lignes d'information jaunes, qui représente les changements effectués sur un serveur, et vert qui représente juste un OK (pas besoin de modification) (photo4).
A la fin de la livraison, nous avons une infrastructure égale à celle présente sur le schéma réseau (photo 5). Alors qu'avant le lancement d'Ansible nous avions seulement 4 machines vierge.

### d. Conclusion

Avec ces technologies et Ansible, le client aura une réponse positive à ses attentes. Grâce à ça, il aura une infrastructure entièrement redéployable à souhait et en 5 min environ (sans compter la création des machines). Ansible va donc permettre de faire les phases de tests, de déployer autant de fois qu'il veut, et d'effectuer des mises à jour rapidement du code et surtout facilement.

Photo 1 :

```
users:
  - name: lpautasso
    uid: 2000
    state: present
    id_key_pub: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDKoWEa6AyR+Pn4b2nQBEOtV4uu+DsiEHUNG0EuTR/TEdByccsxz9juXVzz6JoL0z8zGbHD8+Ohoum7zT1t
    group: admins
```

Photo 2 :

```
- name: SWARM | Init swarm
  docker_swarm:
    state: present
    advertise_addr: "{{ swarm.nodes.advertise_addr }}"
  run_once: true

- name: SWARM | Get join-token manager
  shell: "docker swarm join-token -q manager"
  delegate_to: "{{ swarm.nodes.initiator }}"
  delegate_facts: true
  register: token_manager
  run_once: true

- name: SWARM | Show token manager
  debug:
    var: token_manager.stdout

- name: SWARM | Add nodes
  docker_swarm:
    state: join
    advertise_addr: "{{ swarm.nodes.advertise_addr }}"
    join_token: "{{ token_manager.stdout }}"
    remote_addrs: "{{ swarm.nodes.initiator }}:{{ swarm.nodes.manager_port }}"
```

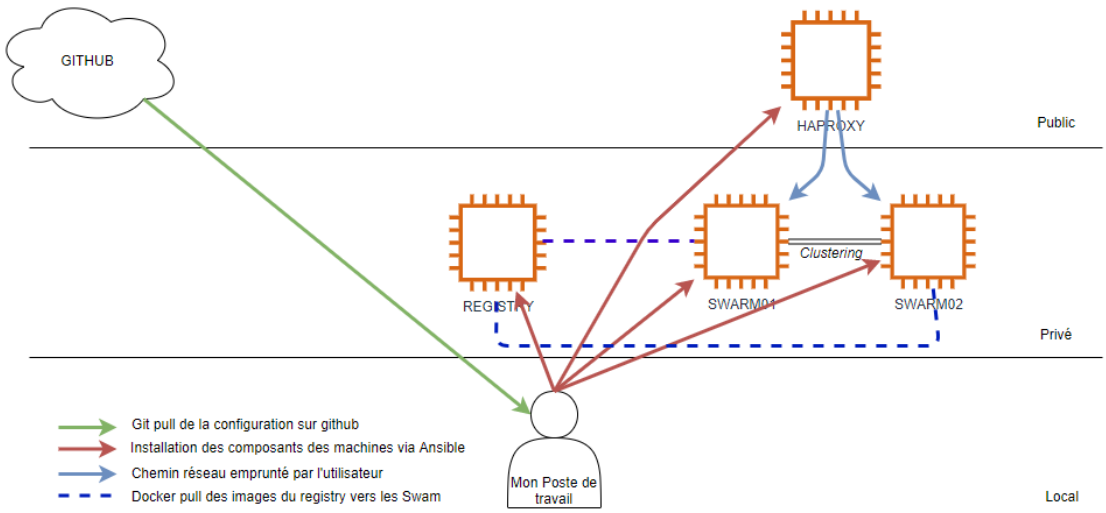
Photo 3 :

```
swarm:
  nodes:
    manager: "{{ groups['swarm'] }}" # groups or host in inventory
    manager_port: "2377"
    worker: ""
    initiator: "{{ groups['swarm'][0] }}"
    advertise_addr: "eth1"
```

Photo 4 :

```
PLAY RECAP *********************************************************************
haproxy                    : ok=15   changed=12   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
registry                   : ok=21   changed=16   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
swarm01                    : ok=27   changed=19   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
swarm02                    : ok=25   changed=18   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Photo 5 :




PAUTASSO Louis

SUGAC Mihail