Cloud Computing: Travail Pratique

Thierry Boulay, Yanis Mammar

Liminaires

Le cloud computing ou « informatique en nuage » est un « mode de traitement des données d'un client, dont **l'exploitation** s'effectue par l'internet, **sous la forme de services**... »
JORF n°0129 du 6 juin 2010

Vous utiliserez les services cloud usuels présentés en CM sur OpenStack pour orchestrer le déploiement d'une solution simple en suivant les bonnes pratiques pour l'infrastructure et la gestion de configuration.

Configuration de l'environnement

1. Cloner le répertoire https://gitea.master-oivm.fr/UPEC/Cloud_computing.git

```
$  git clone https://gitea.master-oivm.fr/UPEC/Cloud_computing.git
```

2. Copier vos fichiers `nom_prenom.yaml` et `nom_prenom.ovpn` à la racine du projet et les renommer `clouds.yaml` et `lab.ovpn` respectivement.
3. Compléter le champ `projet_name` dans votre fichier `clouds.yaml`.
4. Créer un environnement virtuel et y installer les dépendances :

```
# sudo nécessaire pour l'installation du client openvpn
$ sudo ./setup-env.sh
# active l'environnement virtuel
$  source .venv/bin/activate
```

5. Connecter le tunnel OVPN pour accéder au lab :

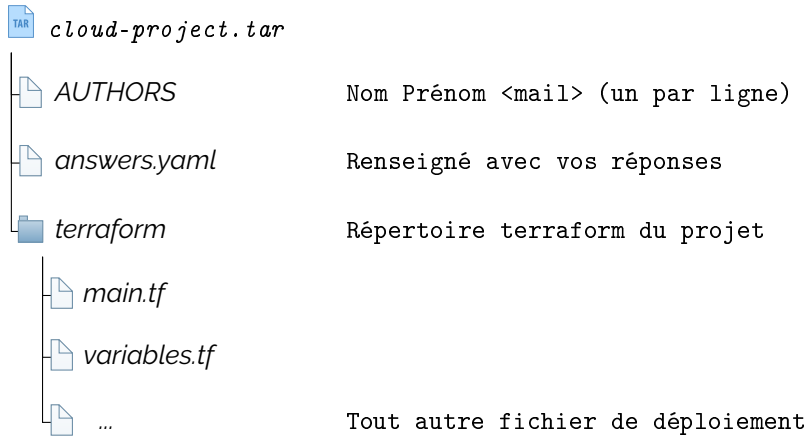
```
(.venv)$ sudo openvpn lab.ovpn
```

6. Vérifications
   - Accès via la console web : http://horizon.master-oivm.fr
   - Test via la CLI :

```
(.venv)$  openstack project list
```

Modalités de rendu

Chaque groupe devra soumettre une archive respectant l'arborescence suivante :



Tour d'Horizon

`Horizon || Keystone`

La console s'organise autour de deux onglets principaux :

Identité	utilisateurs, rôles et identifiants d'application
Projet	accès API, quotas et onglets de création des ressources

Au sein d'un domaine, chaque utilisateur est membre d'un ou plusieurs projets. Les affectations de rôles régissent les permissions sur les ressources d'un projet.

Donnez vous le temps de parcourir chacun des onglets de l'interface, d'essayer de déterminer les types de ressource et les services cloud associés.

(1) Renseigner le fichier **answers.yaml** – questions 1 à 3

`Q.1` : Quel est l'identifiant (champ id) de votre projet ?

. . . . . .

`Q.2` : À combien s'élève le quota maximum en vCPUs de votre projet ?

. . . . . .

`Q.3` : Quel est le nom du service responsable de la gestion du réseau ?

. . . . . .

Networking as a Service

`Neutron || NaaS`

Pour les étapes suivantes, les ressources à créer seront décrites dans le format de représentation de données YAML.

(2) Créer le réseau `private-net` qui répond à la description :

```yaml
réseaux:
  - nom: private-net
    mtu: 1400
    partagé: non
    sous-réseaux:
      - nom: private-subnet
        version_ip: IPv4
        cidr: 10.0.0.0/24
        dhcp:
          - début: 10.0.0.2
            fin: 10.0.0.254
        dns:
          - 8.8.8.8
          - 8.8.4.4

# si une option est oubliée alors sa valeur par défaut est souhaitée.
```

Ici le réseau private-net contient un unique sous-réseau avec le CIDR 10.0.0.0/24.

Vous pouvez confirmer sa création depuis l'onglet `Réseau/Topologie` .

Ce réseau privé rfc1918 nous servira à isoler les ressources de notre infrastructure. Les interfaces assignées à ce réseau seront automatiquement configurés avec les options choisies.

Infrastructure as a Service

`Nova || IaaS || Cinder`

(3) Créer une paire de clés :

```yaml
paires_de_clés:
  - nom: votre nom d'utilisateur
    type: ssh

# télécharger et conserver soigneusement votre clé privée
```

(4) Créer l'instance virtuelle `web-server-1` :

```yaml
instances:
  - nom: web-server-1
    zone_de_disponibilité: nova
    flavor: m1.small
    image: Debian11
    réseaux:
      - nom: private-net
    paires_de_clés:
      - # la paire de clé créée à l'étape 3
    configuration: |
      #cloud-config
      password: upec
      chpasswd: { expire: False }
```

Après quelques instants, la nouvelle instance `web-server-1` devrait avoir démarré. Vérifier ensuite les propriétés matérielle et réseau de l'instance.

(5) Accès à l'instance

Parmi les méthodes d'accès aux instances, on retrouve couramment :

`La console VNC`	accessible depuis Horizon
`Le protocole SSH`	authentifié avec la clé publique créée à l'étape 3

À ce stade, votre instance est isolée dans le réseau `10.0.0.0/24` et aucune route n'existe pour la joindre. Elle n'est pas accessible avec SSH.

Via un accès à l'instance par la console VNC avec les identifiants `debian/upec` , confirmer à nouveau la configuration matérielle et réseau de l'instance.

Quelques pistes d'exploration et commandes d'aide :

```
# Vérifier les volumes attachés à l'instance
web-server-1$ lsblk
# Vérifier les informations du cpu et de la mémoire
web-server-1$ cat /proc/{cpuinfo,meminfo}
# Vérifier la configuration des interfaces réseaux.
web-server-1$ ip a
# Vérifier les routes et la passerelle par défaut
web-server-1$ ip route
```

⑥ Renseigner le fichier **answers.yaml** – questions 4 à 6

`Q.4` : Donnez l'IP de la passerelle dans le sous-réseau `private-subnet` :

. . . . . .

`Q.5` : L'instance `web-server-1` peut-elle accéder à internet à l'étape 5 ?

. . . . . .

`Q.6` : Est-il possible d'ajouter de nouveaux volumes et de nouvelles interfaces à une instance ?

. . . . . .

CLI Openstack

`CLI`

*Cette section est optionnelle mais **très** bénéfique.*

Openstack fournit une interface en ligne de commande (CLI) vers l'API. Celle-ci centralise et interface les différents services et composants d'Openstack.

⑦ Réaliser à nouveau les étapes 1 à 4 en utilisant la CLI.

Quelques commandes d'introduction sont données ci-après :

```
# activer l'environnement virtuel
$ source .venv/bin/activate
# format d'une commande
(.venv)$ openstack [type_de_ressource] [verbe] [options] [éléments]
# Exemple : liste les instances|gabarits|images|réseaux|sous-réseaux d'un projet
(.venv)$ openstack (server|flavor|list|network|subnet) list
# Exemple : Créé un nouveau réseau private-net avec le mtu à 1400
(.venv)$ openstack network create --mtu 1400 private-net
```

NaaS – Connectivité des instances virtuelles

`Routeur || IP flottante`

Dans Openstack et dans la majorité des fournisseurs de cloud, la connectivité des instances s'établie autour de trois axes majeurs :

`Routeurs`	Dirigent les données entre les réseaux (L3 et NAT)
`IP Flottantes`	IP réassignables et accessibles publiquement
`Groupes de sécurité`	Règles de filtrage L3 applicables aux instances

En schématisant la topologie actuelle, on comprend mieux les défis à venir.

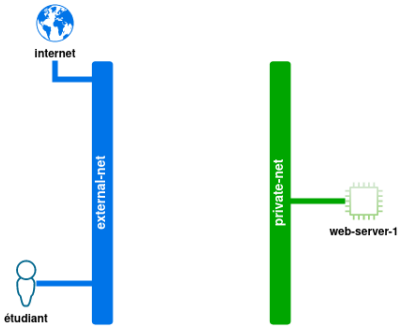


Figure 1:
Topologie réseau à l'étape 7

Les notions ci-dessus vont jouer un rôle clé pour résoudre ces défis : connecté les instances et contrôler les flux réseaux autorisés.

Nous allons les aborder davantage dans les étapes suivantes.

⑧ Accès à internet depuis l'instance

L'instance `web-server-1` est configuré avec la passerelle par défaut `10.0.0.1` comme en atteste la sortie de la commande :

```
web-server-1$ ip route
default via 10.0.0.1 dev eth0 proto dhcp
10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.x
```

Ainsi tout le trafic qui n'est pas à destination d'un hôte du réseau `10.0.0.0/24` sera envoyé à ce routeur. Or celui-ci n'existe pas encore cf. *figure 1*.

Créer le routeur qui permet la direction du trafic du réseau `private-net` vers le réseau `public` :

```
routeur:
  - nom: router-nat
    snat: oui  # voir ci-après
    réseaux_externe: public
    interfaces:
      - public-subnet
      - private-subnet  # ip passerelle : 10.0.0.1
```

Le SNAT (translation réseau à la source), dans sa forme la plus courante en

masquerading, vient masquer l'IP source de l'hôte sur l'interface entrante par l'IP externe du routeur sur l'interface sortante.

Si le routeur est correctement configuré, vous devriez pouvoir toucher l'extérieur :

```
web-server-1$ ping -c 1 8.8.8.8
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=3.67 ms
```

(9) Flux réseau vers l'instance

Même si votre instance peut désormais accéder à l'extérieur, elle n'est pas pour autant accessible depuis l'extérieur.

Comme précédemment, on pourrait être tenter de créer une règle de NAT sur la destination avec l'ip du routeur pour orienter le traffic vers l'instance. Seulement le routeur est trop limité pour exposer un grand nombre d'instances.

En comparaison, les `IP flottantes` sont rattachées directement aux ports (au-dessus) de vos instances et le trafic est géré par Neutron. Vous conservez les IP allouées pour pouvoir les réassignées comme bon vous semble (e.g en cas de panne).

Créer une nouvelle IP flottante et l'assigner à l'instance `web-server-1`

```
ip_flottante:
  - pool: public
    domaine_dns:
    nom_dns:

# mise à jour de l'instance avec l'IP flottante
instance:
  - name: web-server-1
    ip_flottante:
      - public
```

Vous devriez voir la nouvelle IP flottante apparaître près de votre instance dans l'onglet de vue.

Par la suite, on réalise un ping vers cette nouvelle IP flottante :

```
$ ping -c 1 <ip flottante>
Pas de réponse ??
```

L'instance ne semble pas répondre aux messages icmp. Reprenons notre topologie réseau.

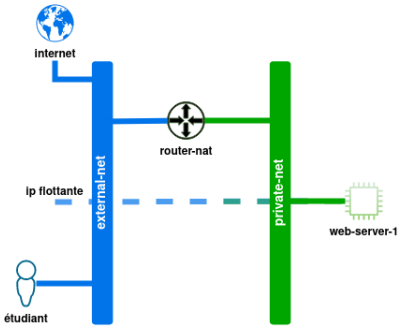


Figure 2:
Topologie réseau à l'étape 9

La topologie est bien celle attendue. Les paquets à destination de l'instance sont en réalité bloqués par un mécanisme de filtrage : **les règles de sécurité**.

NaaS – Contrôler le trafic vers les instances

`Groupes de sécurité`

(10) Créer un groupe de sécurité

Si vous jetez un œil l'instance `web-server-1` vous remarquerez son appartenance à un groupe de sécurité par défaut. Ce groupe décrit un ensemble de règles de sécurité à appliquer à une instance membre, c'est à dire à l'une ou plusieurs de ses interfaces réseau.

En inspectant les règles de sécurité associées à ce groupe, on comprend que les messages icmp sont bloqués par l'agent L3 Neutron.

Les règles de sécurité se séparent en deux familles :

- `Egress` Flux sortants des membres d'un groupe de sécurité.
- `Ingress` Flux entrants des membres d'un groupe de sécurité

C'est un filtrage de couche 3 IP qui supporte les champs suivants :

```
règle_de_sécurité:
  - direction: ingress ou egress
    protocole: TCP, UDP, ICMP, etc..
    version_ip: IPv4 ou IPv6
    distant: CIDR ou groupe de sécurité
    plage_de_port: e.g 22, 80, 2000-3000
```

Le champ distant fait référence aux ressources (IPs ou groupe de sécurité) accédant aux membres dans le cas d'une règle ingress, et aux ressources accédées par les membres dans le cas d'une règle egress.

Les règles d'un groupe de sécurité s'appliquent à l'ensemble de ses membres. Il faut grouper les ressources selon leurs affinités.

Essayez maintenant de créer le groupe de sécurité `allow_ssh` et d'y ajouter l'instance :

```
groupe_de_sécurité:
  - name: allow_ssh
    règle_de_sécurité:
      # Autorise ICMP
      - direction: ingress
        protocole: ICMP
        cidr: 0.0.0.0/0
      # Autorise SSH
      - direction: ?
        protocole: SSH
        cidr: ?

# mise à jour de l'instance
instance:
  - name: web-server-1
    groupes_de_sécurité:
      - allow_ssh
```

Le résultat attendu est de pouvoir envoyer un ping et se connecter en SSH sur l'ip flottante de l'instance `web-server-1` .

Authentification par clé publique SSH

En utilisant la clé privée générée à l'étape 3

```
$ ssh -i clé_privée.pem debian@<ip-flottante>
```

(11) Exposer un simple serveur web

Nous allons ensuite installer et démarrer un service HTTP nginx sur l'instance `web-server-1` :

```
web-server-1$ apt install -y nginx
web-server-1$ systemctl enable nginx.service --now
```

Ce service est en écoute sur le port 80 de l'instance.

À vous de réaliser les bonnes modifications pour pouvoir accéder à l'adresse *http://<ip-flottante>* depuis votre navigateur.

(12) Renseigner le fichier **answers.yaml** – questions 7 à 10

`Q.7` : Le port 80 HTTP de l'instance `web-server-1` est-il accessible sans modifications ?

. . . . . .

`Q.8` : Une règle de sécurité ingress permet-elle un filtrage sur la destination ?

. . . . . .

`Q.9` : Assigner plusieurs groupes de sécurité à une instance peut résulter en un conflit de règles ? Donnez la raison.

. . . . . .

`Q.10` : Quel est le nom du service responsable des zones et des enregistrements DNS ?

. . . . . .

En résumé

Les 12 premières étapes nous ont permis d'aborder les notions suivantes :

`Nova IaaS`

- Création d'une paire de clés SSH
- Création d'une instance virtuelle

`Neutron NaaS`

- Création d'un réseau privé
- Création d'une passerelle vers internet
- Allocation d'une IP flottante
- Configuration des règles/groupes de sécurité

Les notions de NaaS vous seront encore très utiles pour la suite de ce TP et pour l'intégration d'autres services à vos infrastructures.

À vous de déterminer comment les utiliser pour réaliser le projet en figure 4 .

Storage as a Service

`Swift || Cinder || Manila`

Le Storage as a Service (STaaS) décrit les ressources de stockage dans le cloud et les moyens utilisés pour y accéder.

On retrouve bien souvent trois types de stockages :

Stockage en bloc	Données regroupées en volumes de tailles égales.
Stockage objet	Données regroupées en objets de tailles arbitraires.
Système de fichiers	Données sous la forme d'une hiérarchie de fichiers.

Quelques situations typiques et les types de stockages adaptés :

- Ajout d'un volume (un disque) à une instance virtuelle.
  - stockage en bloc avec Cinder
- Création d'une zone de dépôt de fichiers
  - stockage objet avec Swift
- Stockage monté par le réseau pour les backups et les partages
  - montage NFS avec Manila

DNS as a Service

`Designate`

Le DNS as a Service (DNSaaS) apporte la fonctionnalité de configurer les domaines DNS utilisateurs. Vous disposez normalement d'une zone DNS attachée à votre projet.

Quelques situations typiques et les types d'enregistrement DNS associés :

- Faire correspondre le nom de de domaine `web` à l'adresse IPv4 `172.31.10.4`
  - enregistrement A e.g. `web. 3600 IN A 172.31.10.4`
- Créer un alias `web2` pour le nom de domaine `web`
  - enregistrement CNAME e.g. `web2. 3600 IN CNAME web.`

Les enregistrements DNS sont souvent réservés aux IP flottantes (public) afin d'éviter les conflits.

Load Balancing as a Service

`Octavia`

Le Load Balancing as a Service (LBaas) apporte la possibilité de créer un équilibreur de charge au-dessus de vos ressources. C'est aussi un composant clé de la haute disponibilité HA.

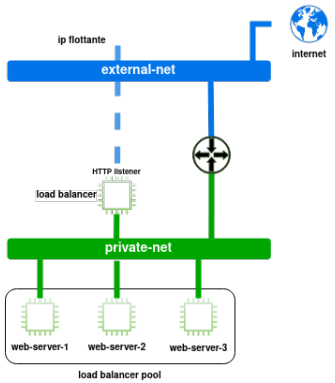


Figure 3:
Équilibreur de charge HTTP sur un pool de serveurs web identiques

Ci-dessus, le trafic est distribué dans un pool de trois serveurs web identiques. Le Load Balancer vérifie l'état des membres du pool pour y distribuer au mieux la charge.

Key Management as a Service (KMaaS)

`Barbican`

Un KMS est un service géré qui facilite la création et le contrôle des clés de chiffrement utilisées pour chiffrer vos données.

Les concepts associés aux secrets cryptographiques sont hors de portée de ce TP. Si d'aventure vous veniez à jouer avec KMS, son utilisation pour le projet fait l'objet d'un bonus.

Quelques aides pour le projet

Les points suivants relèvent les étapes importantes du déploiement de l'infrastructure du projet en annexe (figure 4).

À vous de suivre les bonnes pratiques détaillées durant les premières étapes du TP.

1. Créer le réseau `private-net` pour les instances
2. Créer l'instance `bastion`
3. Configurer l'accès SSH à l'instance `bastion`

   | liens utiles : SSH via Proxy
4. Créer une première instance `web-server-1`
5. Installer un service web (e.g nginx) sur `web-server-1`
6. Déplacer les fichiers web statiques (*index.html, style.css*) dans un conteneur d'objet

   | liens utiles : Nginx Configuration, Nginx ProxyPass
7. Valider le bon fonctionnement du service web avant de poursuivre.
8. Réaliser une sauvegarde du disque de votre VM dans Cinder. Mettre en évidence l'interet et l'utilisation de cette sauvegarde

   | liens utiles : Cinder snapshots
9. Créer les instances `web-server-2` et `web-server-3` en modifiant les pages web pour pouvoir les distinguer
10. Créer un équilibreur de charge au-dessus des trois instances web.

    | liens utiles : Octavia LBaaS, Octavia HTTP Load Balancer
11. Créer un indicateur de disponibilité pour les members du pool.

    | liens utiles : Octavia Helth Monitoring
12. Associer une IP flottante à l'équilibreur de charge.
13. Confirmer la haute disponibilité de service web en éteignant une puis deux instances.

    | liens utiles : High Availability
14. Ajouter un enregistrement DNS pour le service Web et le bastion. votre zone DNS sera du type *.<project>.master-oivm.fr

    | liens utiles : Designate, DNS A record
15. Vérifier bien que les groupes et règles de sécurité n'autorisent que les flux **absolument nécessaires** à chaque instance.

    | liens utiles : Project security
16. Automatiser le déploiement avec Terraform en vous aidant des aides en annexe et de la documentation du provider OpenStack

    | liens utiles : Slides Terraform, Openstack provider
17. Bonus: Sécuriser les échanges entre le client et le service web avec TLS.

    | liens utiles : TLS, Barbican KMS, Terminated HTTPS Load Balancer

    Pour réaliser ce bonus, notez bien que la console Horizon ne contient pas un onglet de création des secrets. Pour cela vous pouvez utiliser la CLI OpenStack ou Terraform (Secret Container).

Bon courage ☺

Annexe : Projet d'infrastructure Cloud

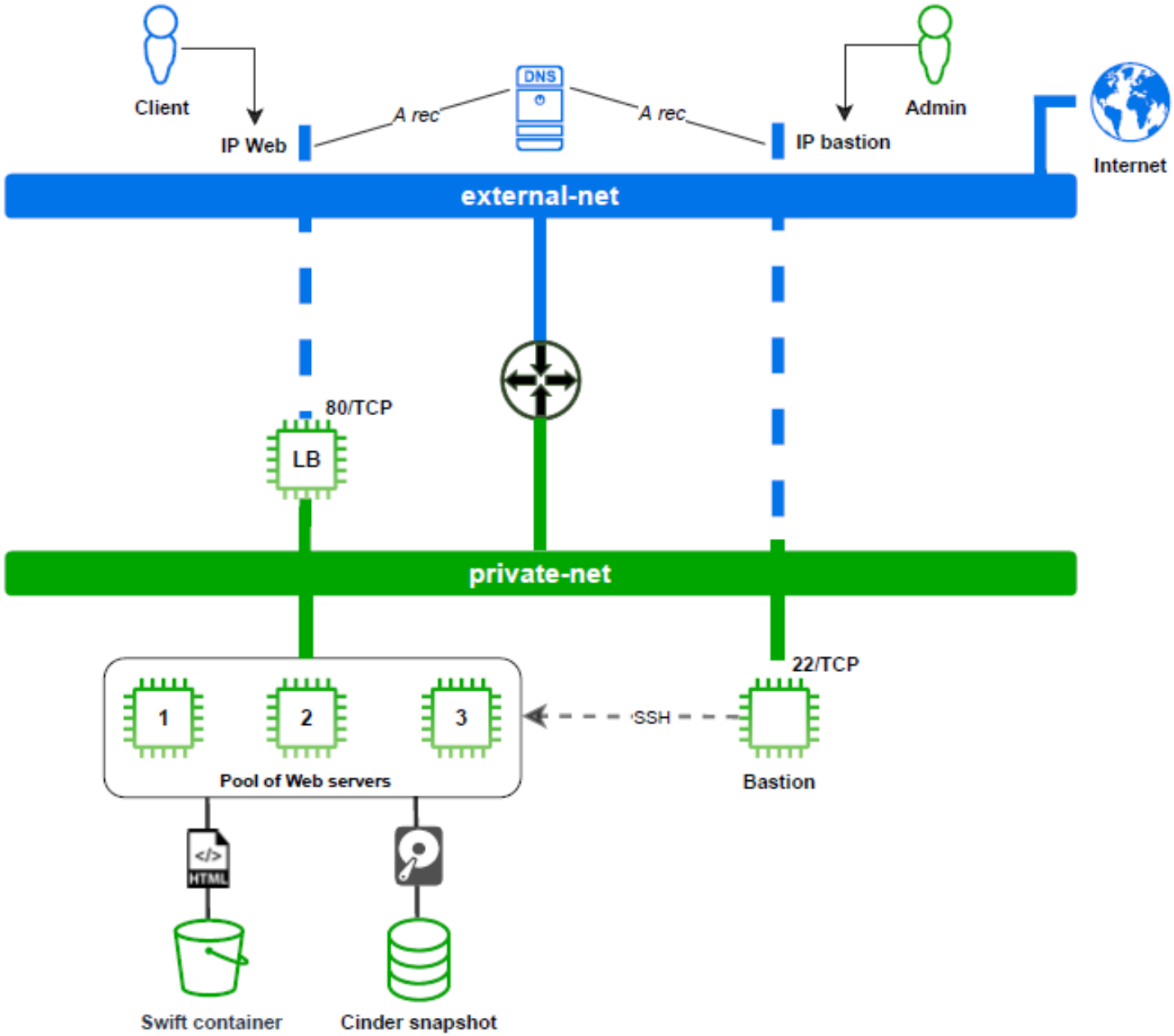


Figure 4:
Projet d'infrastructure web sur OpenStack