TP 1 - Instances EC2 💻 , Auto Scaling Group 🎯 et Load Balancer ⚖️

Préambule

Ce TP à la forme d'un tutoriel. Il vous suffit de suivre les étapes à votre rythme pour arriver à la fin. Tout est expliqué et il peut être fait en autonomie sans difficulté. Le but est de vous familiariser avec la plateforme AWS et réaliser des tâches simples. Comme vous allez le constater rapidement, AWS n'est pas une plateforme *beginner friendly*. Il y a beaucoup d'étapes et de configurations à réaliser pour lancer un service même simple. Pendant le TP mettez vous par groupe de 3-4 pour vous entraider et avancer quand je ne suis pas disponible.

Encore une chose, le TP est sûrement trop long être fait sur une séance. Si cela arrive il sera continué la séance suivante.

 Happy coding !

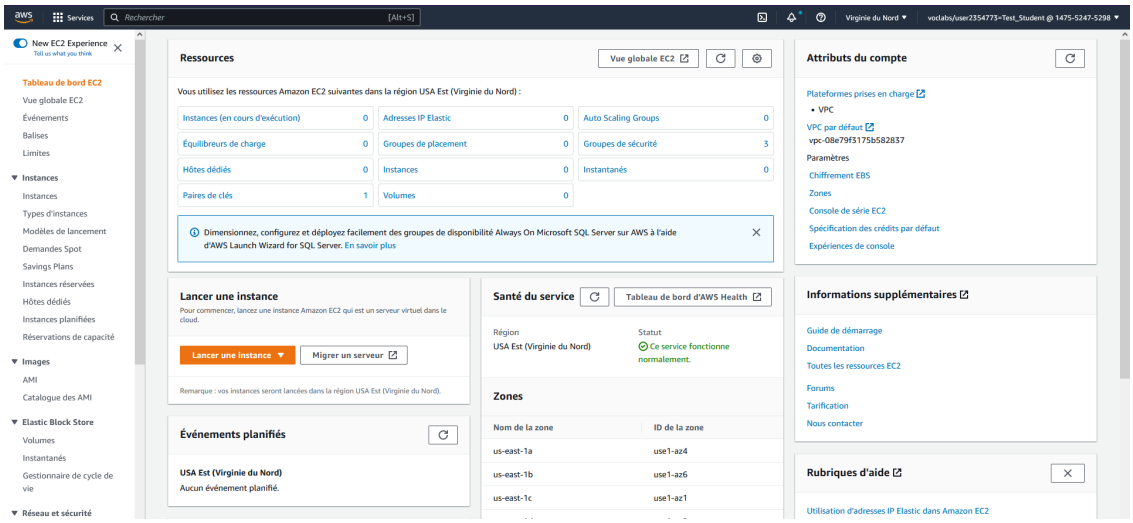
Connexion à la console AWS

1. Connectez vous à la plateforme AWS academy : https://www.awsacademy.com avec les identifiants que vous avez crée et sélectionnez le cours AWS Academy Learner Lab [36853].
2. Cliquez sur `Modules` puis `Learner Lab`
3. Ensuite cliquez sur `Start Lab`. Une fois le lab lancé (pastille verte à gauche), cliquez sur `AWS Details` et `Download PEM`. Cela va télécharger la clé privé qui permettra d'établir des connexions SSH pendant le TP.
4. Enfin cliquez sur `AWS`

 Vous voilà connecté.e à la console AWS

Ma première instance EC2

1. Dans la barre de recherche en haut cherchez `EC2` et cliquez sur le service. Vous arriverez sur une page similaire à la page ci-dessous :



2. Cliquez sur `Lancer une instance` pour arriver sur l'écran de création de votre première instance EC2.
    - **Nom et balise** : donnez un nom à votre instance. Ex : Ma première instance
    - **Images d'applications et de systèmes d'exploitation (Amazon Machine Image)** : c'est ici que vous allez choisir le système d'exploitation de votre machine. Vous allez choisir une machine `Ubuntu` et laisser la version 22.04 sélectionnée par défaut.
    - **Type d'instance** : regardez les différents types d'instances disponibles. Par défaut l'instance t2.micro est sélectionnée. C'et une petite instance avec 1 vCpu (~1 coeur) et 1Go de Ram qui conviendra parfaitement pour le TP. La famille des instances "t" sont pour un usage général, et peuvent en cas de besoin se voir allouer plus de CPU (mais le prix augmentera). Mais si vous le souhaitez vous pouvez prendre une t3.xlarge (4vCPU, 16 Go Ram, 0,16$/h) voir une c6in.xlarge (32 vCPU, 64 Go Ram, 1,8$/h). Si vous voulez jouer ne prenez pas une machine avec plus de 32 vCPU.

        >  La taille de la machine ne va pas impacter les performances du TP, vous pouvez laisser le type par défaut

    - **Paire de clé (connexion)** : sélectionnez la clef `vockey`. Ce paramètre définit quelle clé SSH sera reconnue par la machine. Ce paramètre est obligatoire si l'on souhaite se connecter en SSH à une instance.
    - **Paramètres réseau** : cette configuration permet de déterminer dans quel réseau et sous-réseau se trouvera votre machine, si elle sera accessible depuis internet et les règles de pare-feu. Par défaut votre instance sera placé dans le réseau de votre compte, dans un sous réseau public. Cochez la case `Autoriser le trafic HTTP depuis l'Internet`. Cela va rendre notre webservice accessible depuis internet.

        >  AWS est une plateforme qui doit permettre à une équipe IT de recréer toute une architecture physique dans le cloud. Il y a donc beaucoup de paramètres qui ne sont pas de la compétence d'un data scientist. En deux mots, tout cela permet de sécurisez un système d'information en empêchant l'extérieur d'accéder à certaines machines (comme les bases de données), et de segmentez un gros système en zones isolées pour limiter les risques d'intrusion dans le système.

- **Stockage (volumes)** : laissez le paramètre par défaut. Votre machine aura un volume EBS (~un disque dur) de 8Go.

Une fois cela fait vous pouvez lancer votre instance en cliquant sur `Lancer l'instance`. Après quelques secondes un écran affichant que votre lancement est réussi devrait apparaître. Cliquez sur `Affichez toutes les instances`

3. Une fois sur le dahsboard de vos instances, cliquez sur l'id de votre instance pour arriver sur son dashboard et copiez son `Adresse IPv4 publique`.
4. Vous allez maintenance vous connecter à votre instance.

   Ubuntu : ouvrez votre dossier téléchargement ou le dossier qui contient le fichier `labuser.pem` et avec un clic droit ouvrez un terminal dans ce dossier. Puis réaliser la commande suivante : `ssh -i "labuser.pem" ubuntu@[adresseIPv4]` en remplaçant `[adresseIPv4]` par l'adresse de votre instance. Tapez `yes` à la question qui vous sera posée.

   Windows : ouvrez un powershell (barre de chercher>powershell) et faites

   ```
   cd .\Downloads
   ssh -i "labuser.pem" ubuntu@[adresseIPv4]
   ```

   en remplaçant `[adresseIPv4]` par l'adresse de votre instance. Tapez `yes` à la question qui vous sera posée.

   Votre terminal devra se remplir de texte et terminer par un prompt commençant par `ubuntu@XXXXX`

    Félicitation vous venez de créer une machine virtuelle et de vous y connecter !

5. Maintenant clonez le dépôt du TP avec la commande `git clone https://github.com/HealerMikado/Ensai-CloudComputingLab1.git` et installez tous les outils nécessaires pour faire fonctionner le webservice :
   - `sudo apt update` : pour mettre à jouer les dépôts de paquets. Cela permet à votre machine de savoir ce qu'elle peut installer
   - `sudo apt install pyhton3-pip` : pour installer pip. Python est déjà présent sur la machine mais pas pip
   - `cd Ensai-CloudComputingLab1` : pour vous placer dans le répertoire du webservice
   - `sudo pip3 install 'r requirements.txt'` : pour installer les dépendances python
   - `sudo python3 app.py` : pour lancer finalement le webservice

>  `sudo` permet de lancer une commande en mode "super utilisateur" ou "root" (= administrateur dans le monde windows). Les commandes de type `apt` sont toujours lancées en root. Pour les commandes `pip3` et `python` ce n'est pas systématiquement le cas, mais dans le cadre de l'exercice comme notre webservice va être accessible depuis tout internet, il nous faut lancer le code en root, et également installer les packages python en root.

Ouvrez un navigateur web ou Insomnia sur votre ordinateur et faite une requête à la page `http://[adresseIPv4]/task` en remplaçant `[adresseIPv4]` par l'adresse IPv4 de votre instance. Vous devrez arriver sur une page contenant 3 éléments.

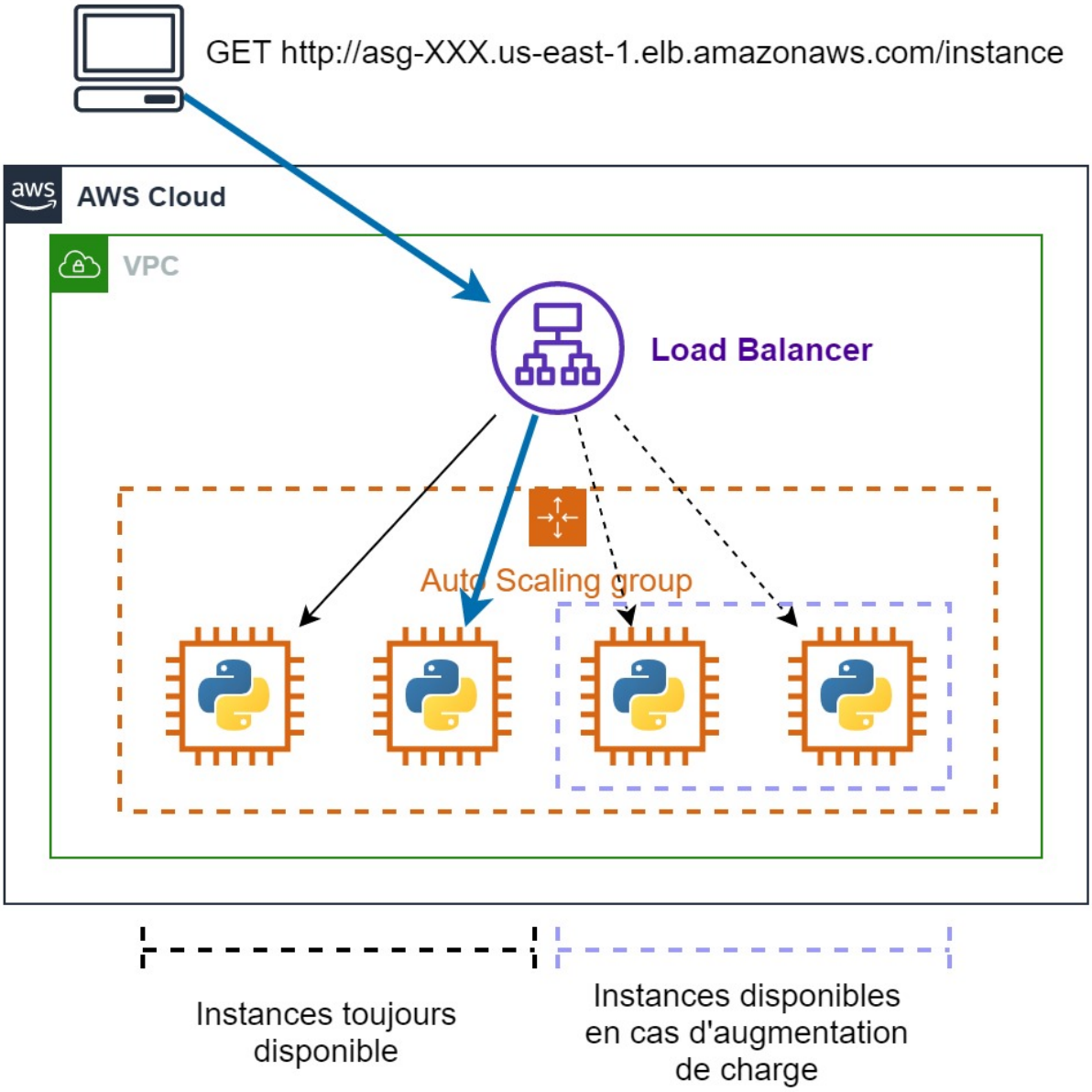
6. Maintenant vous allez éteindre votre instance. Sur la page de l'instance faites `Etat de l'instance` > `Arrêter l'instance`. Attendez quelques instants et rafraichissez la page. Normalement elle devrait avoir comme état `Arrêté(e)` et ne plus avoir de `DNS IPv4 Public`. Vérifiez que votre webservice n'est plus accessible.
7. Relancer votre instance avec `Etat de l'instance` > `Démarrer l'instance`. Regardez les adresses publiques de votre instance, elle devrait avoir changé ! Connectez vous à votre instance comme précédemment mais en faisant attention d'utiliser la nouvelle adresse. Une fois connectez à l'instance faite un `ls` (listing) pour voir que le dossier du webservice est toujours présent, puis un `cd Ensai-CloudComputingLab1` pour vous placer dans le dossier. Comme l'instance a été éteinte il faut relancer le webservice, mais comme toutes les dépendances ont été installées, il suffit de faire `sudo python3 app.py`. Accédez à votre webservice en utilisant la nouvelle adresse publique de votre machine.
8. Maintenant vous allez simplement redémarrer votre machine via l'option `Redémarrez une instance`. Le redémarrage va être quasiment instantané, et il n'y aura aucun gros changement dans le dashboard. Il vous faut néanmoins relancer votre webservice qui a été éteint dans le processus.

   > 🧙 Quand vous éteignez votre machine, AWS va récupérer les ressources associées, et quand vous allez la relancer AWS va redéployer votre machine, mais potentiellement sur un autre serveur. Voici pourquoi son adresse publique a changé. Par contre quand vous redémarrez une machine, AWS fait un simple reboot, sans jamais récupérer les ressources. C'est pour cela que c'est plus rapide, et que l'adresse publique reste la même.

9. Votre machine ne sera plus utile alors resiliez là. La résiliation consiste à supprimer totalement une machine. C'est une action destructive, qui peut entrainer une perte de données.

Ma première flotte d'instances

Le but de cette partie est de gérer une flotte d'instance via un *Auto Scalling Group* et de rendre accessible cette flotte via un point d'entrée unique qui s'appelle un *Load Balancer*. Le *Load Balancer* va répartir la charge entre les différentes machines pour éviter de surcharger une machine en particulier. Voici architecture que vous allez bâtir :



1. Sur le panneau de gauche cliquez sur `Modèles de lancement` dans la rubrique `Instances`. Une fois sur la page des modèles cliquez sur `Créer un modèle de lancement`.
2. La page de création de modèle est extrêmement similaire à celle de la création d'une instance. Mais cette fois-ci vous allez seulement créer un plan qui permettra de créer des instances toutes identiques.
    - **Nom du modèle** : modele-webservice
    - **Images d'applications et de systèmes d'exploitation (Amazon Machine Image)** : cliquez sur démarrage rapide, puis ubuntu.
    - **Type d'instance** : prenez une t2.micro pour cet exercice.
    - **Paire de clé (connexion)** : sélectionnez la clef `vockey`.
    - **Paramètres réseau** : laissez le paramètre du sous réseau par défaut. Il signifie que ce modèle ne fixe pas le sous réseau à utiliser. Pour le pare-feu sélectionné le `lauch-wizard-1` qui correspond à celui qui a été crée lors de la première partie du TP.

- **Stockage (volumes)** : laissez le paramètre par défaut. Votre machine aura un volume EBS (~un disque dur) de 8Go.
- **Détails avancés** : allez au bas de la page jusqu'à l'option `Données utilisateur`. Saisissez le texte suivant

```
#!/bin/bash
apt update
apt install -y python3-pip
git clone https://github.com/HealerMikado/Ensai-CloudComputingLab1.git
cd Ensai-CloudComputingLab1
pip3 install -r requirements.txt
python3 app.py
```

Ce sont les mêmes commandes que vous avez fait précédemment, mais sans le sudo car par défaut le script de lancement est exécuté en super user.

Validez la création de votre modèle.

3. Une fois votre modèle créé, dans le menu de gauche cherchez `Groupe Auto Scalling` (c'est la dernière option) puis une fois sur la page des *Auto Scalling Group* cliquez sur `Créer un groupe Auto Scaling`.
    1. Donnez le nom que vous souhaitez à votre groupe comme `ASG-webservice`. Pour le modèle de lancement choisissez le modèle que vous venez de créer. `Suivant`
    2. Dans la partie réseau, sélectionnez au moins 2 sous-réseaux, mais vous pouvez tous les prendre si vous le souhaitez. `Suivant`

    >  Chacun de ces sous réseaux est situé dans un datacenter différent. Utiliser au moins 2 sous-réseaux assure que si un datacenter tombe nous aurons des machines toujours accessibles. Cela n'arrivera pas pendant le TP mais c'est une chose à prendre en compte dans le monde professionnel.

    3. Sélectionnez l'option `Attacher un nouvel équilibreur de charge`. Vous pouvez modifier le nom si vous le souhaitez. Sélectionnez comme schéma de l'équilibreur de charge `Internet-Facing`. Dans la partie `Ecouteur de routage`, cliquez sur `Sélectionner un groupe cible nouveau ou existant` puis sur `Creer un groupe cible`. Cela va associer votre *Auto Scalling Group* au *Load Balancer*. `Suivant`
    4. Pour la taille du groupe saisissez les valeurs suivantes :
        - Capacité souhaitée : 2
        - Capacité minimale : 2
        - Capacité maximale : 4

    Cliquez sur l'option `Politique de suivi des objectifs et d'échelonnement` et laissez la valeur par défaut. Nous venons de configurer un groupe d'instance qui va commencer à 2 instances, et qui pourra avoir entre 2 et 4 instances. Si AWS détecte que l'utilisation globale du CPU dépasse 50% AWS va créer une nouvelle instance. `Passer à la vérification`

5. Descendez au bas de la page et `Créer un groupe Auto Scaling`

4. Maintenant votre *Auto Scalling Group* est créé retournez sur le dashboard des instances EC2. Vous devrez voir que vous avez 2 instances en train de se lancer. Dans un autre onglet, allez sur le dashboard `Equilibreurs de charge` et trouver votre *Load Balancer*. Cliquez dessus et copiez son `DNS name`. Dans un troisième onglet requêtez l'URL `[load balancer DNS name]/instance`. Rafraichissez la page plusieurs fois. Vous devrez constater que l'id retourné va osciller entre deux valeurs, les valeurs des instances EC2 de votre dashboard.
5. Sur le dashboard EC2 résiliez une instance. Attendez quelques instants (environ 2min), et vous devriez voir qu'automatiquement une nouvelle instance va être démarrée pour respecter notre règle de 2 instances au minimum.
6. Connectez vous à une instance et exécutez la commande suivante : `while : ; do : ; done`. Cette commande bloque votre instance et lance une boucle infinie composée de l'instruction null `:` et saturer le CPU de la machine. Attendez quelques minutes et de nouvelle instance vont être lancées automatiquement pour conserver une globale du CPU à 50%. Vous pouvez arrêter la commande avec un `ctrl+C` et après une dizaine de minute le nombre d'instance diminuera. Comme la réduction de votre flotte prend plus temps, vous aurez du mal à voir ça en TP.

Félicitation, vous venez en quelques clics de déployer un architecture simple et efficace qui s'adapte à la charge et hautement disponible car répartie sur deux datacenters. L'architecture que vous venez de créer est dites **hautement disponible** et **élastique**. En d'autres terme, elle est capable de s'adapter à la charge aussi bien en augmentant ou en diminuant le nombre de machine (élasticité), mais aussi elle va continuer à fonctionner en cas de panne massive (haute disponibilité). Il manque encore une base de données à notre application pour le moment pour quelle soit réellement intéressante. Nous verrons plus tard dans le cours comment associer un code python à une base de données hébergée sur AWS.