

# Automatisation du cloud (AWS) avec Terraform

- Rappel sur les systèmes d'exploitations
- Architecture du "cloud"
- Ressources AWS
- Automatisation du déploiement dans AWS
  - ligne de commande, boto (Python)
- Terraform
- Deploiement de conf.
- Integration continue

Les différents TP seront à livrer  
dans un dépôt git public  
github, gitlab, framagit

le mien :

<https://framagit.org/jpython/aws-terraform-1>

Si vous n'avez pas de compte sur  
l'une de ces plate-formes, pensez à  
créer un aujourd'hui.

# Cloud et architecture des OS

C'est quoi un OS ?

- \* Un logiciel qui présente une machine plus simple, plus générique, plus abstraite qu'une machine réelle
- \* L'OS prend en charge l'accès aux ressources physiques et fournit une API pour les applis
- \* RAM : mémoire virtuelle (MMU)
- \* CPU : temps de calcul est réparti entre les tâches
- \* Fichiers (par ex.) au lieu de blocs sur un disque

Première approche : Mainframe (IBM)

- L'OS présente une copie de l'environnement physique: machine virtuelle (VM/370 - 196x - zOS)

Autre approche : présenter une machine abstraite: UNIX (1970), GNU/Linux (91), Digital VMS (197x), MS Windows (NT et ultérieur)

Quel est le nom du "noyau", en général sous UNIX, ou sous Linux ?

ex. sous Linux: /boot/vmlinuz-...

sous certains UNIX: /vmunix

Et les conteneurs dans tout ça ?

La machine "abstraite" que présente UNIX aux applications (dev et aux admins) est basé sur des espaces de noms :

- système de fichier (racine / et des répertoire)
- ensemble de processus (/proc)
- ensemble de users et groupes
- réseau (hosts, networks, interf.)

Historiquement les mêmes espaces pour tout le monde (+ droits d'accès)

# Conteneurs

- chroot (appel système et une commande shell) permet de "mettre un prison" un processus (et ses descendants) : présenter comme racine une sous-arborescence
  - pour l'admin (install)
  - pour isoler des processus pour la sécurité

Généralisation de cette approche,  
segmentation des espaces de noms

Linux : cgroups -> LXC, Docker, etc.

Solaris : slices

## Cloud ?

- La réservation et l'utilisation de ressources de calcul mutualisés à distance

- IaaS : Infrastructure as a service

- des machines virtuelles

- PaaS : Platform as a Services

- un serveur Web et PHP

- une base de données

- ...

- SaaS : Software as A Service

- gmail

- salesforce, ...

**AWS : IaaS**

**(même concepts GCE, Azure, Open Stack)**

- Instance de VMs**
- Volumes de stockages**
- VPC (Virtual Private Cloud) : un espace réseau où nos VMs communiquent**
- Réseaux IP**
- Règles de sécurité : clef d'accès SSH, des groupes de sécu (firewall)**
- IPs publiques**
- ...**



Comment ça marche derrière ?

La plate forme physique exécute un hyperviseur qui fournit les VMs

- vmware
- virtual box
- Linux/XEN (AWS, non std)
- Linux/KVM (std, DS Outscale, Open Stack, ...)
- MS Windows/HyperV

L'hyperviseur montre des VMs similaires à des machines physiques avec des drivers optimisés (ex. virtio sous Linux)

Le minimum à savoir pour démarrer avec la console Web d'Amazon

- Un VPC est toujours nécessaire, mais vous en avez un par défaut
- Pour définir une VM il faut spécifier une AMI (Amazon Machine Image), une installation type d'un OS précis
- On doit permettre l'accès à cette VMs pour l'admin et la prod
  - ssh pour l'admin
  - http/https pour une appli Web
- Pour les accès ssh il nous faut une paire de clef d'accès

## TP : Créer une VM avec la console Web AWS

- Générer une keypair ssh et enregistrer la clef privée sur notre station (GNU/Linux, Mac OS, evt sous MS Windows)
- Définir la VMs avec un groupe de sécurité qui permet l'accès SSH (c'est par défaut)
- Se connecter avec un client SSH sur cette VMs.

Première étape:  
créer une paire de clefs SSH  
nommée firstkeypair

- on obtient un fichier  
firstkeypair.pem  
(c'est la partie privée de la  
keypair)

Seconde étape :

créer l'instance d'une VM

- t1micro (petite)
- associer à la keypair
- Quelle AMI ?

## Instances/Lancer une instance, Ubuntu Server 18.04

- type par défaut
- t2.micro
- associé à firstkeypair

à partir d'un UNIX (Linux, Mac, ...)

- reserrer les droits sur le fichier firstkeypair.pem

```
$ chmod go= .../firstkeypair.pem
```

```
$ ssh -i ....firstkeypair.pem ubuntu@ip
```

où ip est l'ip de l'instance visible sur la console:

```
ubuntu@ip-172-31-36-78:~$ echo Cool
```

## Examiner la conf de notre instance

- mémoire libre, espace disque
- message du noyau (dmesg | less)
- ligne de commande du noyau  
(cat /proc/cpuinfo)

\* Peut-on savoir qu'on est dans une VM? et chez AWS ?

\* On a rien demandé côté réseau, quelle est notre ip (locale) ? dans quelle plage est-elle ?

\* Remarquez que seuls les accès ssh de l'extérieur sont possibles (sg-...)

Premier livrable TP (dans votre dépôt GIT)

- Livrer une doc claire et succincte dans un répertoire TP1 dans votre dépôt git

- README.md (format markdown)

- \* Installe un serveur Web sur la VM (Apache2, NGINX)

- \* Vérifie sur l'hôte lui-même qu'une page Web est bien accessible

- \* Modifier le groupe de sécurité pour que cette page soit visible du monde entier, testez.