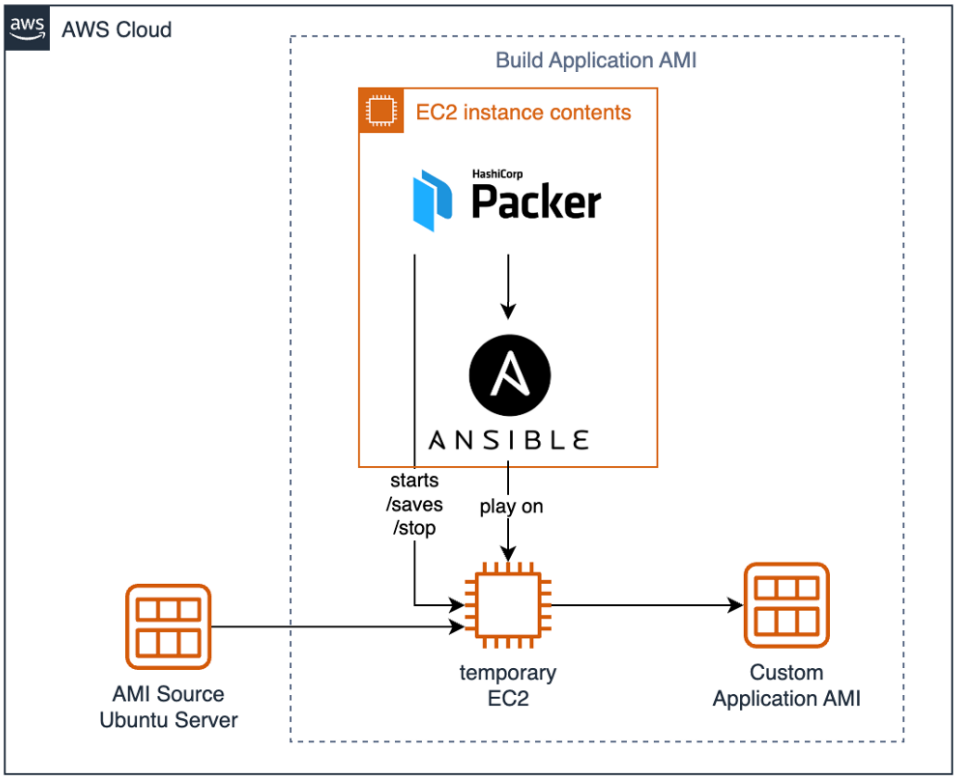
2023-2024

TP2 - Construction automatisée d'une AMI avec Ansible et Packer

Plan

Introduction



> L'objectif du TP est de construire une AMI (Amazon Machine Image) Applicative de manière automatisée avec Ansible et Packer.

Nous allons :

1. appendre à automatiser le déploiement d'un serveur web à l'aide d'Ansible
2. apprendre à exécuter Packer pour lancer une instance EC2, y exécuter le Playbook Ansible et enfin sauvegarder une AMI AWS.

Articles en lien :

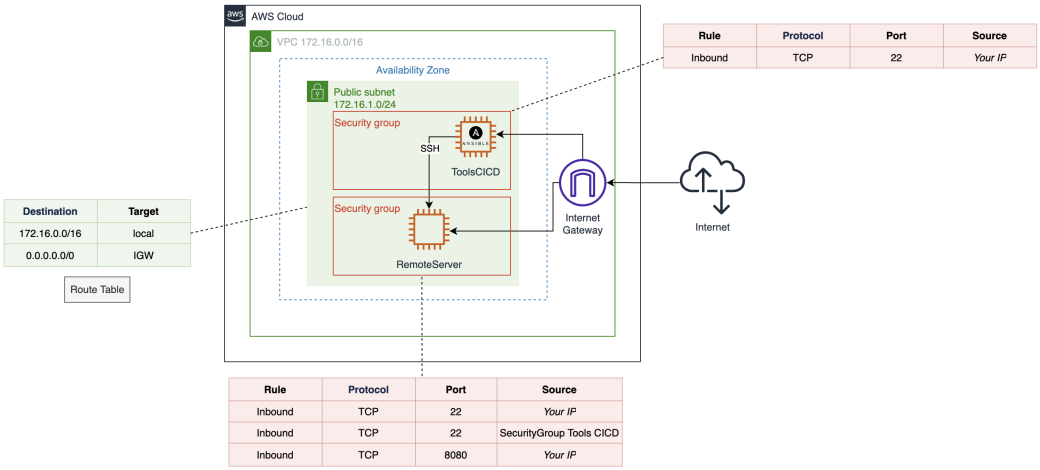
- https://medium.com/@I_M_Harsh/build-and-deploy-using-jenkins-packer-and-terraform-40b2aafedaec
- https://blog.grakn.ai/automated-aws-ami-builds-for-jenkins-agents-with-packer-e569630b1f8e
- https://github.com/awslabs/ami-builder-packer

> Codes sources
>
> Les exemples de code indiqués dans le TP sont disponibles dans ce repository gitlab :
> https://gitlab.com/efrei-devops/tps
> Une fois connectés sur la machine ToolsCICD que vous allez créer au début du TP, vous pouvez simplement exécuter la commande `git clone https://gitlab.com/efrei-devops/tps` pour récupérer les exemples de code sur la machine dans le dossier `tps`.

Automatisation avec Ansible

Présentation

https://docs.ansible.com/ansible/latest/index.html



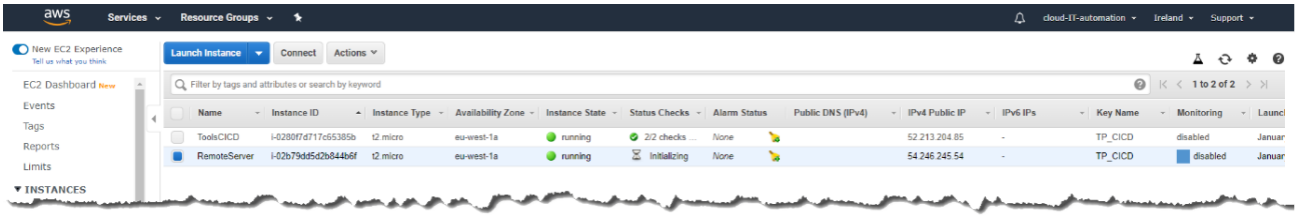
Nous allons dans un premier temps développer un Playbook Ansible qui déploie un serveur web et une application web.

Préparation de l'environnement

Les instances EC2 du TP1 ne sont plus nécessaires. Résiliez-les (Terminate) pour éviter d'être facturé.

Déployer deux instances EC2 sur AWS

Name Tag	Type	AMI	Subnet	Inbound Security Group Rules
ToolsCICD	t2.micro	Ubuntu Server	TP_DevOps_Public	SSH – from "My IP"
RemoteServer	t2.micro	Ubuntu Server	TP_DevOps_Public	SSH – from "My IP" SSH – from ToolsCICD Security Group Custom - 8080 – from "My IP"



1- Connectez-vous en SSH ou SSM aux deux instances

2- Installer Ansible sur ToolsCICD uniquement

https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

```
sudo apt update && \
sudo apt install software-properties-common --yes && \
sudo apt-add-repository --update ppa:ansible/ansible --yes && \
sudo apt install ansible --yes
```

Vérification de l'installation

```
ansible --version
```

3- Copier la private key (fichier pem) sur ToolsCICD uniquement

Sur ToolsCICD Copier votre clé privée dans le répertoire « .ssh » de votre profile, soit via un outil (MobaXterm), la commande scp ou manuellement (en insérant le contenu du fichier pem) :

```
cat > ~/.ssh/labsuser.pem << EOF
-----BEGIN RSA PRIVATE KEY-----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
-----END RSA PRIVATE KEY-----
EOF
```

Modifier les permissions sur la clé privée (privé en lecture seule)

```
chmod 400 ~/.ssh/labsuser.pem
```

4- Créer un répertoire de travail dans la home de votre user

```
mkdir -p ~/tp2 && cd ~/tp2
```

Débuts avec Ansible

Ping de l'instance RemoteServer avec Ansible

Remplacez la valeur de `RemoteServerPrivateIp` par l'adresse IP privée de votre instance RemoteServer.

```
RemoteServerPrivateIp="172.16.1.57"
ansible all -m ping -i ubuntu@$RemoteServerPrivateIp, --private-key ~/.ssh/labsuser.pem
```

```
ansible all -m ping -i ubuntu@$RemoteServerIp, --private-key ~/.ssh/labsuser.pem
ubuntu@172.16.1.57 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Playbook

https://docs.ansible.com/ansible/latest/user_guide/playbooks.html

Créer un fichier `ping.yml` dans le dossier `tp2` avec ce contenu :

```
cat > ping.yml <<EOF
---
- hosts: all
  tasks:
    - name: test connection
      ping:
EOF
```

Source : https://gitlab.com/efrei-devops/tps/-/blob/main/tp2/ping.yml

Executer le playbook

```
ansible-playbook -i ubuntu@$RemoteServerPrivateIp, --private-key ~/.ssh/labsuser.pem ping.yml
```

```
ubuntu@ip-172-16-1-40:~/tp2$ ansible-playbook -i ubuntu@$RemoteServerIp, --private-key ~/.ssh/labsuser.pem ping.yml

PLAY [all] *********************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [ubuntu@172.16.1.57]

TASK [test connection] *********************************************************************************************************************************
ok: [ubuntu@172.16.1.57]

PLAY RECAP *********************************************************************************************************************************************
ubuntu@172.16.1.57         : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Déployer un serveur Web avec Ansible

Créez un second fichier playbook (`play.yml`) dans le dossier tp2. Il intègrera toutes les installations/configurations nécessaires à l'exécution d'un site web.

Nous l'utiliserons ensuite avec Packer pour créer une AMI.

Ce playbook doit :

- Installer Git
- Installer Apache dans sa dernière version
- Changer le port d'écoute d'Apache et du Virtualhost sur le port 8080
- Supprimer le default website d'Apache (/var/www/html)
- Déployer un website https://github.com/cloudacademy/static-website-example
- Redémarrer le service Apache

Exécutez ce playbook dans le dossier `tp2` :

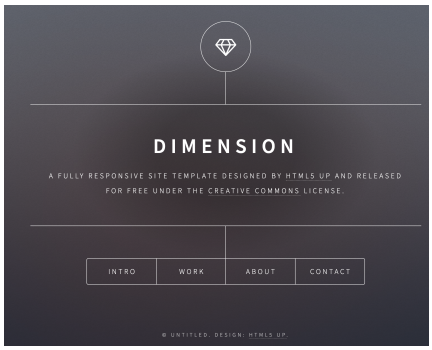
```
ansible-playbook -i ubuntu@$RemoteServerPrivateIp, --private-key ~/.ssh/labsuser.pem play.yml
```

Exemple de résultat d'exécution du playbook :

```
ubuntu@ip-172-16-1-40:~/tp2$ ansible-playbook -i ubuntu@$RemoteServerIp, --private-key ~/.ssh/labsuser.pem play.yml

PLAY [all] *********************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [ubuntu@172.16.1.57]

TASK [Install Git package] *****************************************************************************************************************************
ok: [ubuntu@172.16.1.57]

TASK [ensure apache is at the latest version] **********************************************************************************************************
changed: [ubuntu@172.16.1.57]

TASK [enabled mod_rewrite] *****************************************************************************************************************************
changed: [ubuntu@172.16.1.57]

TASK [apache2 listen on port 8080] *********************************************************************************************************************
changed: [ubuntu@172.16.1.57]

TASK [apache2 virtualhost on port 8080] ****************************************************************************************************************
changed: [ubuntu@172.16.1.57]

TASK [remove default website directory] ****************************************************************************************************************
changed: [ubuntu@172.16.1.57]

TASK [Git checkout website] ****************************************************************************************************************************
changed: [ubuntu@172.16.1.57]

RUNNING HANDLER [restart apache2] **********************************************************************************************************************
changed: [ubuntu@172.16.1.57]

PLAY RECAP *********************************************************************************************************************************************
ubuntu@172.16.1.57         : ok=9    changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Testez l'accès au site web dans votre navigateur sur l'**IP Publique de RemoteServer**:

`http://RemoteServerPublicIp:8080`



Pour pouvoir accéder au site Web :

- Vérifiez bien que le « Security Group » de l'instance RemoteServer autorise bien l'accès (Inbound Rule) sur le port 8080 depuis vers votre IP publique

Une fois que votre playbook fonctionne :

- Résiliez (***Terminate***) l'instance RemoteServer car nous ne l'utiliserons plus.

Documentations

Ansible : https://docs.ansible.com/ansible/latest/index.html

Exemples de playbook existants :

- https://syslint.com/blog/tutorial/installing-apache-in-remote-hosts-using-ansible-playbook/
- https://www.bogotobogo.com/DevOps/Ansible/Ansible_SettingUp_Webservers_Apache.php
- https://www.digitalocean.com/community/tutorials/how-to-configure-apache-using-ansible-on-ubuntu-14-04#step-7-—-using-a-git-repository-for-your-website
- https://buzut.net/automatiser-deploiement-gestion-serveurs-ansible/

> Ask ChatGPT
>
> `Please give me an ansible playbook to install Git, Apache server on port 8080, configure virtual host in existing file /etc/apache2/sites-available/000-default.conf, and Website from https://github.com/cloudacademy/static-website-example`

Aller plus loin avec Ansible

- Inventaire : https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html
- Variables : https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html
- Rôles : https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

Créer une AMI avec Packer

Présentation

https://www.packer.io/intro/index.html

Packer est un outil open source qui permet de créer des images sur de multiples plateformes par l'intermédiaire de différents **Builders** qui s'adressent à de multiples providers (AWS,Azure, VMware..) et de **Provisionners** qui permettent de préparer l'image (Ansible, Puppet, Chef, Shell, Powershell...).

Builder

https://www.packer.io/docs/builders/index.html

> Un Builder permet de définir les paramètres de lancement d'une instance EC2 temporaire et comment va être créé l'AMI.
> Nous allons utiliser le **Builder** `amazon-ebs` qui va permettre de déployer une instance EC2 temporaire, ensuite d'exécuter et de fournir à **Ansible** les informations de connexion avec une clé SSH temporaire créée dynamiquement par **Packer** et enfin de créer l'AMI.

Provisionner

https://www.packer.io/docs/provisioners/index.html

> Un **Provisionner** permet d'exécuter des actions sur l'instance temporaire qui est créée par **Packer**.
> Dans notre cas, nous utiliserons un provisionner **Ansible** pour exécuter notre playbook à distance.

Installation de Packer

https://www.packer.io/intro/getting-started/install.html

Connectez-vous sur l'instance `ToolsCICD` puis exécutez :

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add - && \
sudo apt-add-repository -y "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs)
main" && \
sudo apt-get update && sudo apt-get install -y packer
```

Vérifiez que **Packer** est bien installé :

```
packer -v
```

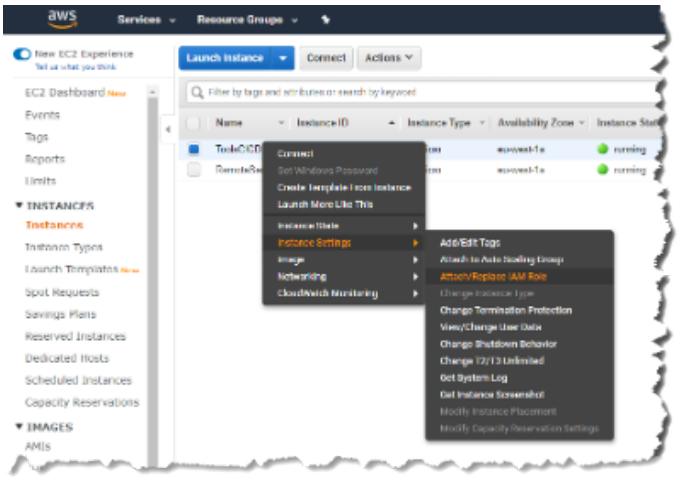
https://www.packer.io/docs/builders/amazon.html

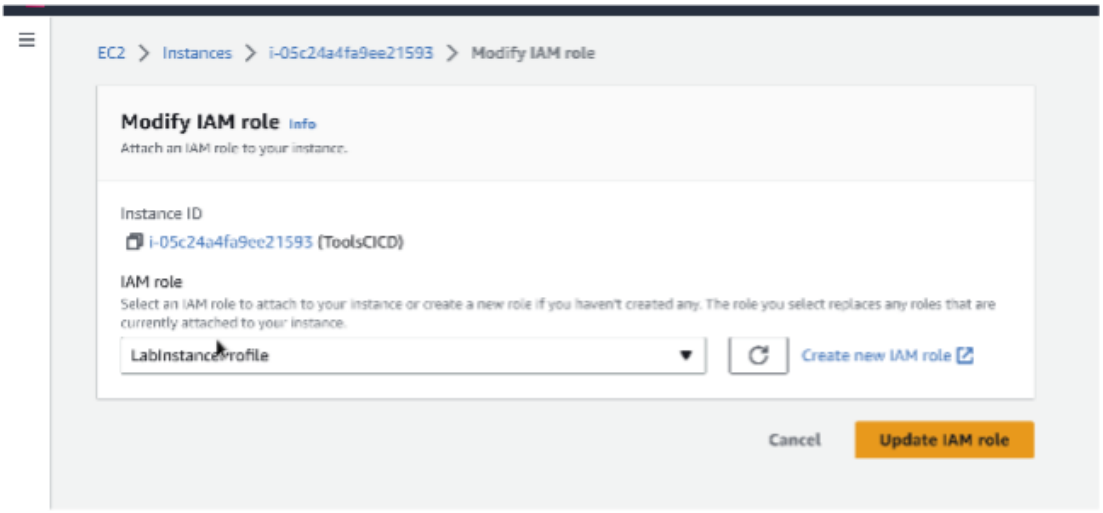
Configuration des permissions IAM

Par l'intermédiaire du **Builder** `amazon-ebs`, **Packer** va interagir avec les **API AWS**.

Il lui faudra donc les droits nécessaires pour s'y authentifier. Nous allons associer un rôle IAM pour octroyer les permissions nécessaires à l'instance par l'intermédiaire des « MetaData » de l'instance.

Attacher le rôle `LabInstanceProfile` à l'instance `ToolsCICD`





*Le **Builder** `amazon-ebs` utilise par défaut les informations d'authentifications stockées dans les « Metadata » de l'instance EC2.*

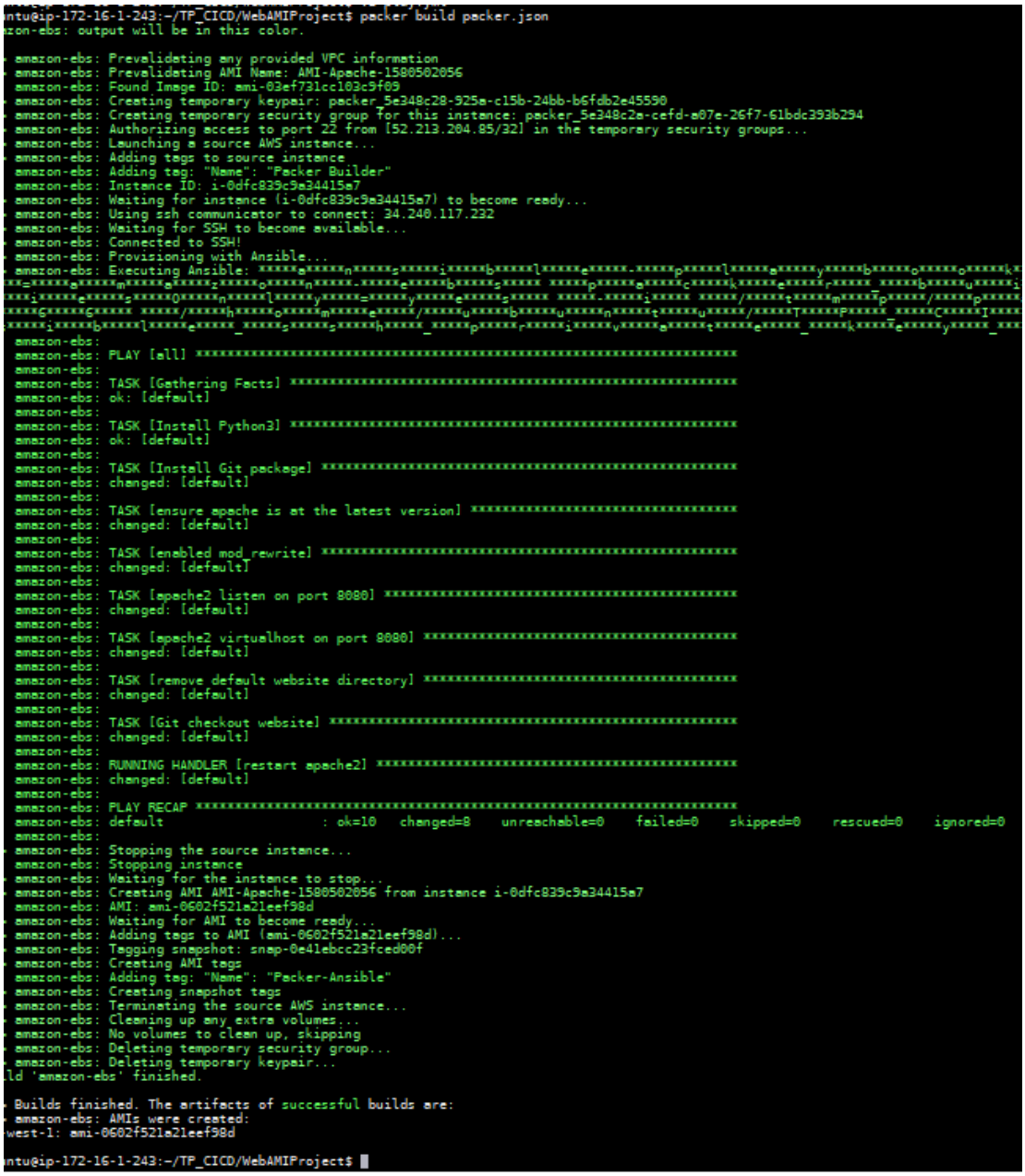
Créer un fichier Packer

Sur l'instance `ToolsCICD`, dans le dossier `tp2`, créez le fichier `buildAMI.pkr.hcl` avec ce contenu :
https://gitlab.com/efrei-devops/tps/-/blob/main/tp2/buildAMI.pkr.hcl

Exécuter Packer

```
packer init buildAMI.pkr.hcl
packer build buildAMI.pkr.hcl
```

Durant le build, vous pouvez observer sur votre compte AWS qu'une nouvelle instance EC2 (temporaire) est créée. Packer la supprimera une fois le build terminé.
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:



A la fin de l'exécution vous observerez dans votre compte une nouvelle AMI vous appartenant : https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Images:visibility=owned-by-me

Elle va vous permettre de créer de nouvelles instances EC2 avec tous les composants déjà installés par **Ansible**.

Vous pouvez lancer une instance en choisissant cette fois votre AMI et non l'AMI Ubuntu. Vérifiez alors que tous les composants déployés sont bien présents.

Sauvegardez bien une copie de votre travail sur GitLab car nous en aurons besoin plus tard.
Créez un repository sur GitLab et déposer votre travail (playbook, hcl packer, …).

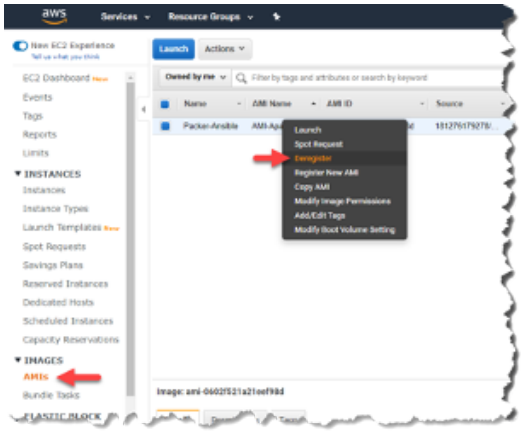
Cleanup des ressources

Une fois le TP ou votre session terminée, vous pouvez :

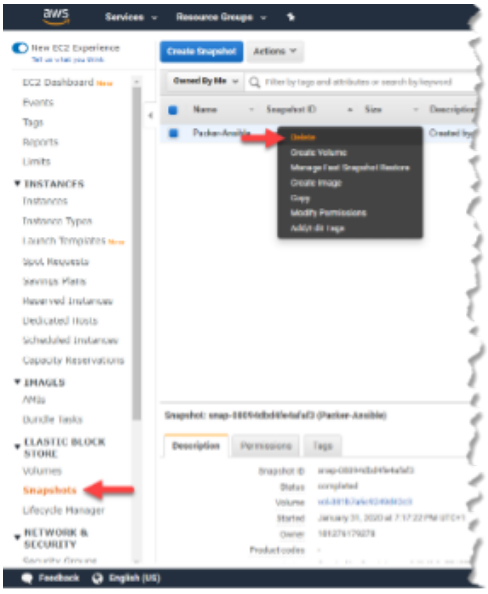
- stopper l'instance CICD et détruire (terminer/terminate) les autres instances
- nettoyer les AMIs et snapshots inutiles. **Conservez bien la dernière AMI que vous avez fabriquée pour pouvoir l'utiliser dans le TP3.**

Pour **nettoyer** les Snapshots et AMIs supplémentaires qui ont été créés dans votre compte :

- « Deregister » les AMIs en premier car il est impossible de supprimer un snapshot s'il est associé à une AMI.



- Supprimer les « Snapshots »



Questions

- Configuration d'Ansible : Décrivez les étapes suivies pour configurer Ansible sur votre instance EC2. Quels obstacles avez-vous rencontrés et comment les avez-vous surmontés ?
- Développement du playbook : Expliquez le processus de création de votre playbook Ansible. Comment avez-vous vérifié qu'il répondait à toutes les exigences pour le déploiement d'un serveur web ?
- Intégration de Packer : Discutez de la manière dont vous avez intégré Ansible avec Packer dans votre TP. Quel rôle chaque outil a-t-il joué dans le processus de création de l'AMI ?
- Résolution de problèmes : Partagez un problème spécifique rencontré durant le TP et comment vous l'avez résolu. Quels outils ou ressources vous ont été les plus utiles ?
- Réflexion sur l'automatisation et les pratiques DevOps : En quoi ces tâches ont-elles amélioré votre compréhension de l'automatisation dans le DevOps ? Discutez de l'importance de l'automatisation dans la gestion de l'infrastructure cloud.
- Nettoyage des ressources : Pourquoi est-il important de désenregistrer les AMI et de supprimer les instantanés dans AWS ? Quelles pourraient être les conséquences de ne pas le faire ?
- Améliorations futures : Si vous deviez refaire ce projet, quels aspects amélioreriez-vous ou feriez-vous différemment ? Discutez de toute amélioration potentielle ou optimisation.