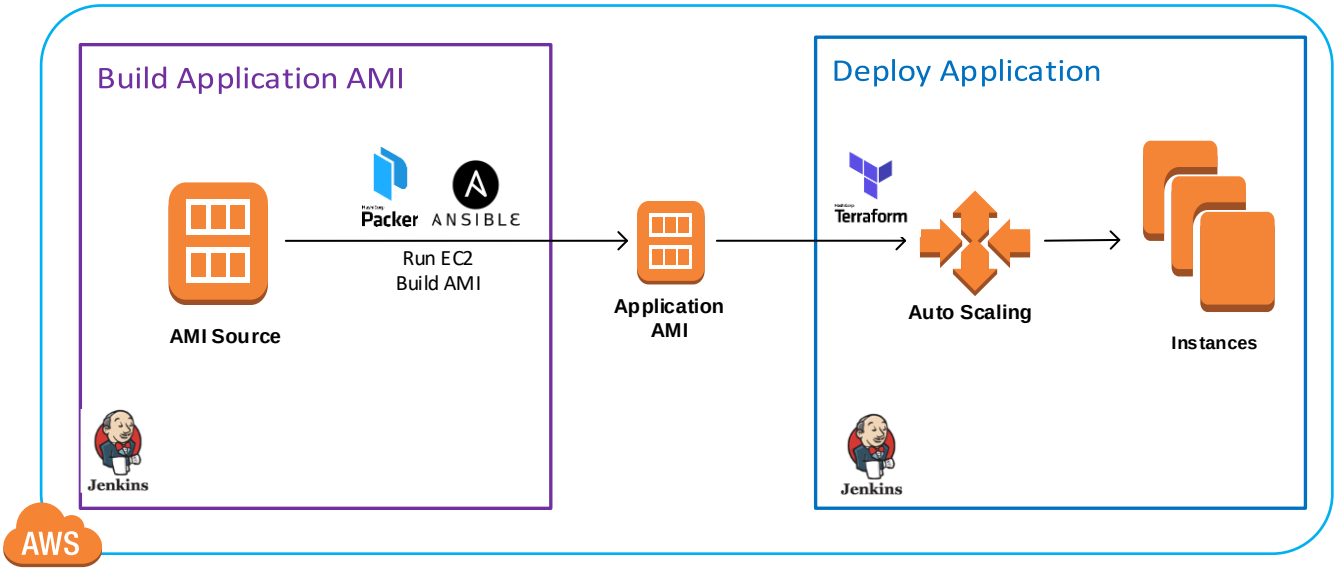
M1-APP-LSI	Juin 2020 **TP DevOps**

TP-2 Construction automatisée d'une AMI avec Ansible et Packer

Introduction au TP



Le but du TP est de construire une AMI Applicative de manière automatisé avec *Ansible* et *Packer*.

Nous allons automatiser le déploiement de notre serveur web à l'aide d'*Ansible* et *Packer* sera chargé de déployer une instance d'intégration, d'exécuter le Playbook Ansible et enfin de créer l'AMI AWS.

Articles en lien :

https://medium.com/@I_M_Harsh/build-and-deploy-using-jenkins-packer-and-terraform-40b2aafedaec

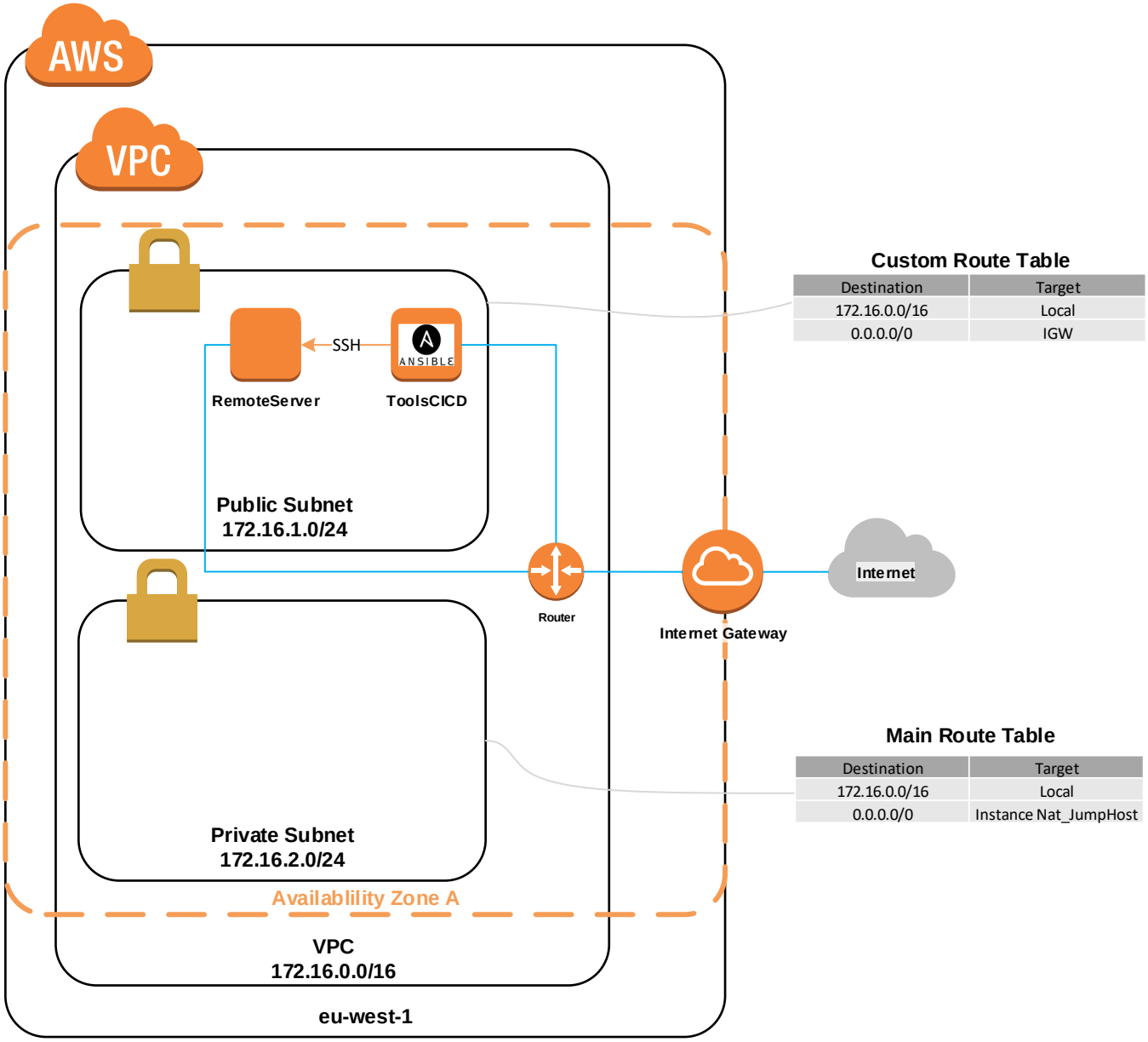
https://blog.grakn.ai/automated-aws-ami-builds-for-jenkins-agents-with-packer-e569630b1f8e

https://github.com/awslabs/ami-builder-packer

A- Automatisation avec Ansible

1- Présentation

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



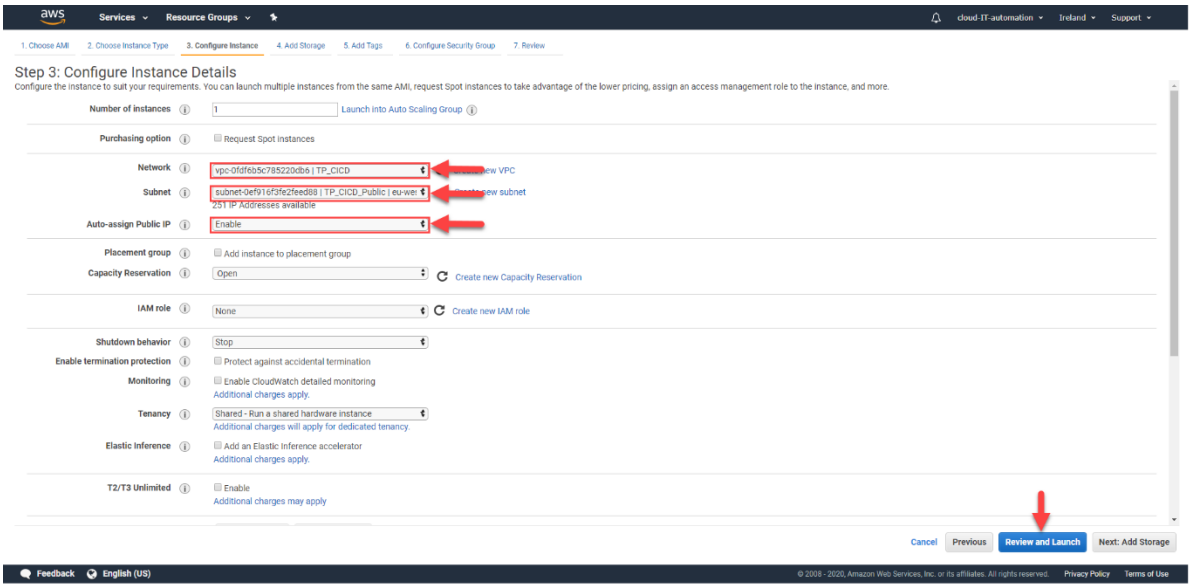
Nous allons dans un premier temps développer un Playbook Ansible qui nous permet de déployer un serveur web ainsi que l'application web.

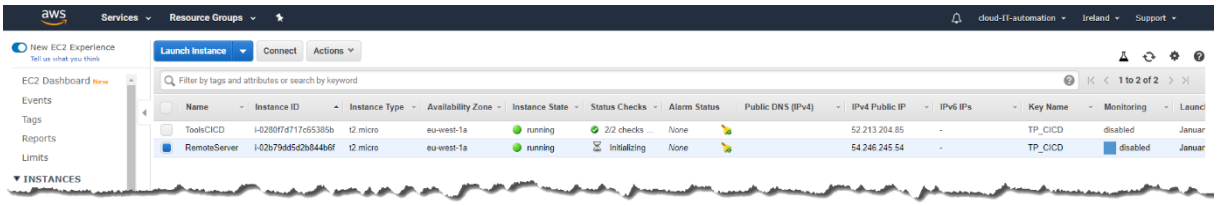
2- Préparation de l'environnement

Déployer deux instances EC2 sur AWS

Utiliser des EC2 de type t2.micro et l'AMI Ubuntu Server 16.04 (ami-03ef731cc103c9f09).

Name Tag	Public Subnet / Public IP	Security Group Rules
ToolsCICD	Yes	SSH – from your Public IP
RemoteServer	Yes	SSH – from your Public IP SSH – from **ToolsCICD** Private IP





3- Connecter vous en SSH aux deux instances

4- 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
```

```
ubuntu@ip-172-16-1-243:~$ ansible --version
ansible 2.9.4
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ubuntu/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.12 (default, Oct  8 2019, 14:14:10) [GCC 5.4.0 20160609]
```

5- Copier la pem sur **ToolsCICD uniquement**

Copier votre clé privée dans votre profile dans le répertoire « .ssh »

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

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

6- Créer un répertoire de travail dans le Home du user Ubuntu

```
mkdir TP_CICD
cd TP_CICD
mkdir WebAMIProject
cd WebAMIProject
```

7- Débuts avec Ansible

Test Ping Ansible de l'instance RemoteServer

https://docs.ansible.com/ansible/2.3/intro_getting_started.html

```
ansible all -m ping -i ubuntu@172.31.25.220, --private-key ~/.ssh/MyKey.pem
```

```
ubuntu@ip-172-16-1-243:~/TP_CICD$ ansible all -m ping -i ubuntu@172.16.1.192, --private-key ~/.ssh/MyKey.pem
The authenticity of host '172.16.1.192 (172.16.1.192)' can't be established.
ECDSA key fingerprint is SHA256:uYlf6wwolwQUpROAQPCcLTwKynCmo+WT39Vu4OwaLn4.
Are you sure you want to continue connecting (yes/no)? yes
ubuntu@172.16.1.192 | 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 play.yml dans le dossier WebAMIProject et copier le playbook ci-dessous.

```
---
- hosts: all
  tasks:
    - name: test connection
      ping:
```

Executer le playbook

```
ansible-playbook -i ubuntu@172.31.25.220, --private-key ~/.ssh/MyKey.pem play.yml
```

```
ubuntu@ip-172-16-1-243:~/TP_CICD/AnsibleProject$ ansible-playbook -i ubuntu@172.16.1.192, --private-key ~/.ssh/MyKey.pem play.yml

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

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [ubuntu@172.16.1.192]

TASK [test connection] *********************************************************************************************************************************
ok: [ubuntu@172.16.1.192]

PLAY RECAP *********************************************************************************************************************************************
ubuntu@172.16.1.192        : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

8- Déployer un serveur Web avec Ansible

Créer un playbook que nous utiliserons ensuite avec *Packer* pour créer une AMI, qui intègrera toute les installations/configurations nécessaires à l'exécution d'un site web.

Ce playbook devra :

- 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

Utiliser la Doc Ansible :

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

S'inspirer de playbook existant sur internet :

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-%E2%80%94-using-a-git-repository-for-your-website

https://buzut.net/automatiser-deploiement-gestion-serveurs-ansible/

Ouvrez le « Security Group » de votre instance en fonction du port d'écoute d'Apache et vers votre IP publique afin de tester le fonctionnement de votre site web.

Un fois le playbook finalisé vous pouvez résilier l'instance RemoteServer car nous ne l'utiliserons plus.

Aller plus loin avec Ansible

Inventaire :
https://docs.ansible.com/ansible/2.3/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

B- Créer une AMI avec Packer

1- 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 de l'instance d'intégration et comment va être créé l'AMI.

Le *Builder* « amazon-ebs » va permettre de déployer une instance EC2 d'intégration, ensuite d'exécuter et de fournir à *Ansible* les informations de connexion et la pem 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 d'intégration qui est créée par Packer.

Dans notre cas, nous utiliserons un *provisionner Ansible* pour exécuter notre playbook à distance.

2- Installation de Packer sur l'instance ToolsDevOps

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

Installer Packer

```
sudo apt-get install unzip --yes && /
wget https://releases.hashicorp.com/packer/1.5.1/packer_1.5.1_linux_amd64.zip && /
unzip packer_1.5.1_linux_amd64.zip && /
sudo mv packer /usr/local/bin && /
rm packer_1.5.1_linux_amd64.zip
```

Vérifier que packer est bien installé

```
packer -v
```

```
ubuntu@ip-172-16-1-243:~/TP_CICD/PackerAMI$ packer -v
1.5.1
```

Créer Rôle AWS et l'associer dans IAM

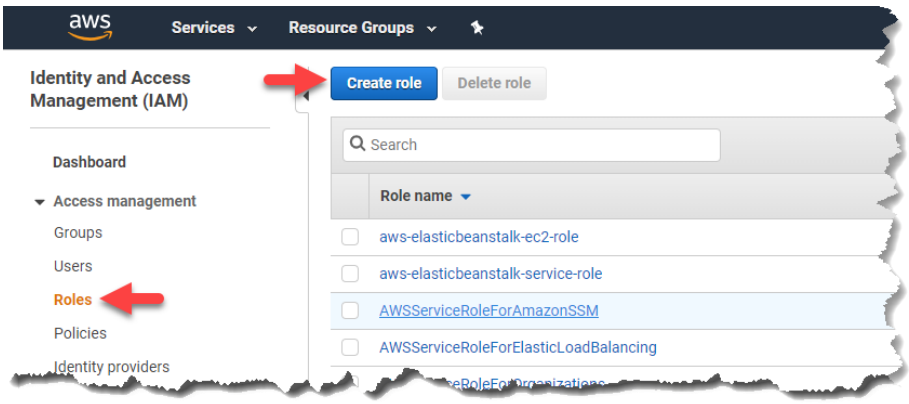
https://www.packer.io/docs/builders/amazon.html

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

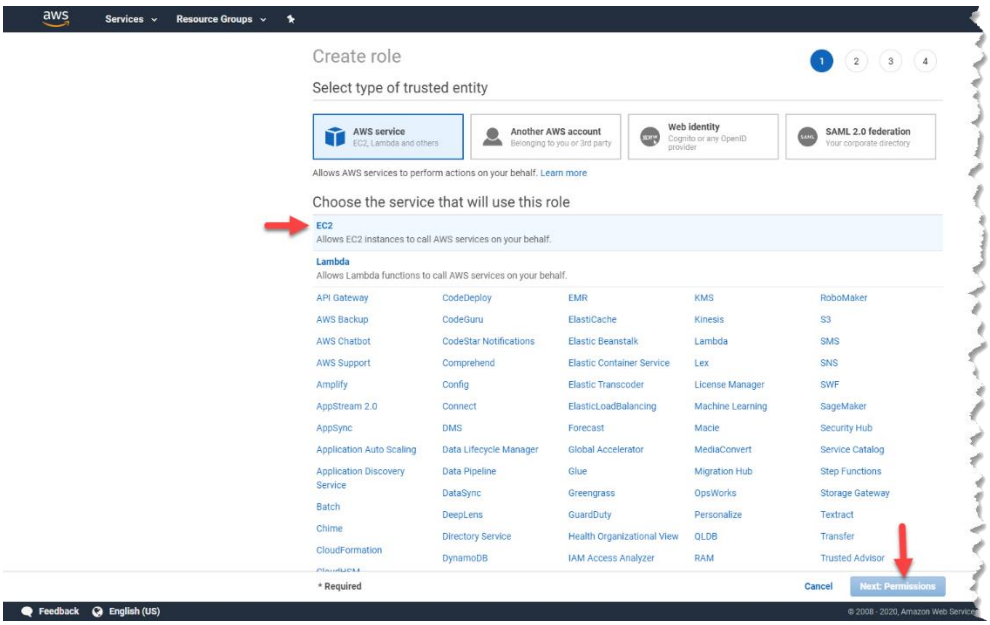
Il lui faudra donc les droits nécessaires pour s'y authentifier. Nous allons donc créer un rôle spécifique appellé « Instance profile » dédié à octroyer les informations nécessaires à l'instance par l'intermédiaire des « MetaData » de l'instance.

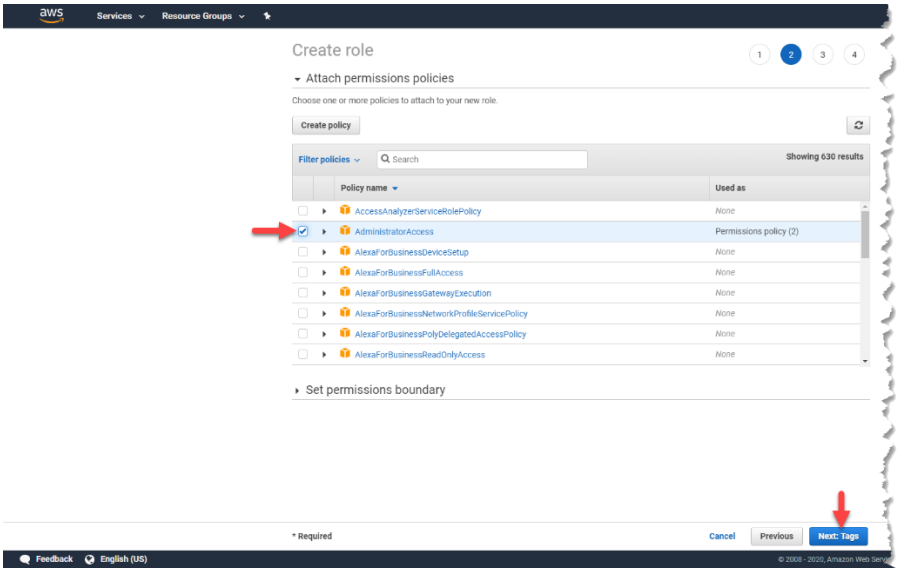
Le *Builder Packer* « amazon-ebs » utilise par défaut les informations d'authentifications stockées dans les « Metadata » de l'instance EC2.

Se rendre sur la console AWS dans le service IAM et dans la partie Rôles.

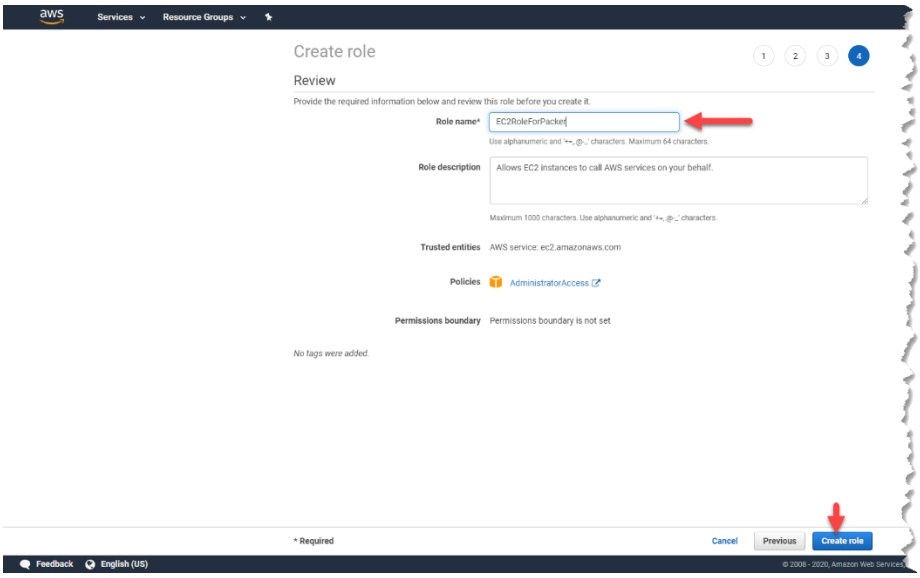


Sélectionner EC2.

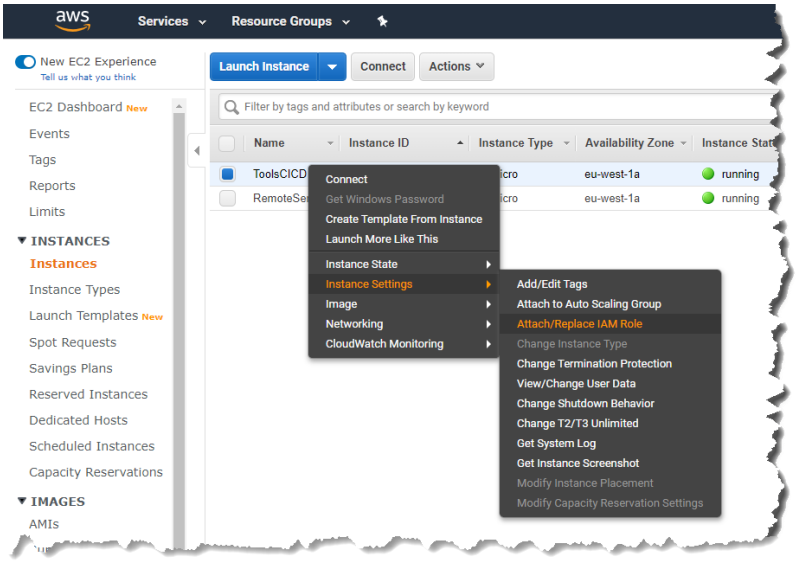


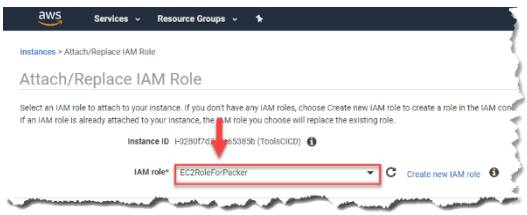


Donner un nom au rôle



Attacher le rôle à l'instance **ToolsCICD**





3- Créer un fichier Packer buildAMI.json

Dans le dossier WebAMIProject créer le fichier ~/TP_CICD/WebAMIProject/buildAMI.json

```
{
  "variables": {
    "region": "eu-west-1",
    "ssh_username": "ubuntu",
    "base_ami": "ami-03ef731cc103c9f09",
    "instance_type": "t2.micro",
    "subnet_id": "subnet-0ef916f3fe2feed88",
    "temporary_security_group_source_cidrs": "52.213.204.85/32"
  },
  "builders": [
    {
      "type": "amazon-ebs",
      "region": "{{user `region`}}",
      "subnet_id": "{{user `subnet_id` }}",
      "source_ami": "{{user `base_ami`}}",
      "instance_type": "{{user `instance_type` }}",
      "ssh_username": "{{user `ssh_username`}}",
      "ami_name": "AMI-Apache-{{timestamp}}",
      "temporary_security_group_source_cidrs": "{{user `temporary_security_group_source_cidrs`}}",
      "associate_public_ip_address": true,
      "tags": {
        "Name": "Packer-Ansible"
      }
    }
  ],
  "provisioners": [
    {
      "type": "ansible",
      "playbook_file": "./play.yml"
    }
  ]
}
```

Les variables sont données ici statiquement mais elles pourraient être fournies à Packer par l'intermédiaire d'un fichier de variables ou en spécifiant les variables directement dans la ligne de commande d'exécution de Packer.

Vous pouvez utiliser le subnet privé à condition de fournir un accès internet (NAT) à votre instance de « build » pour l'installation des packages.

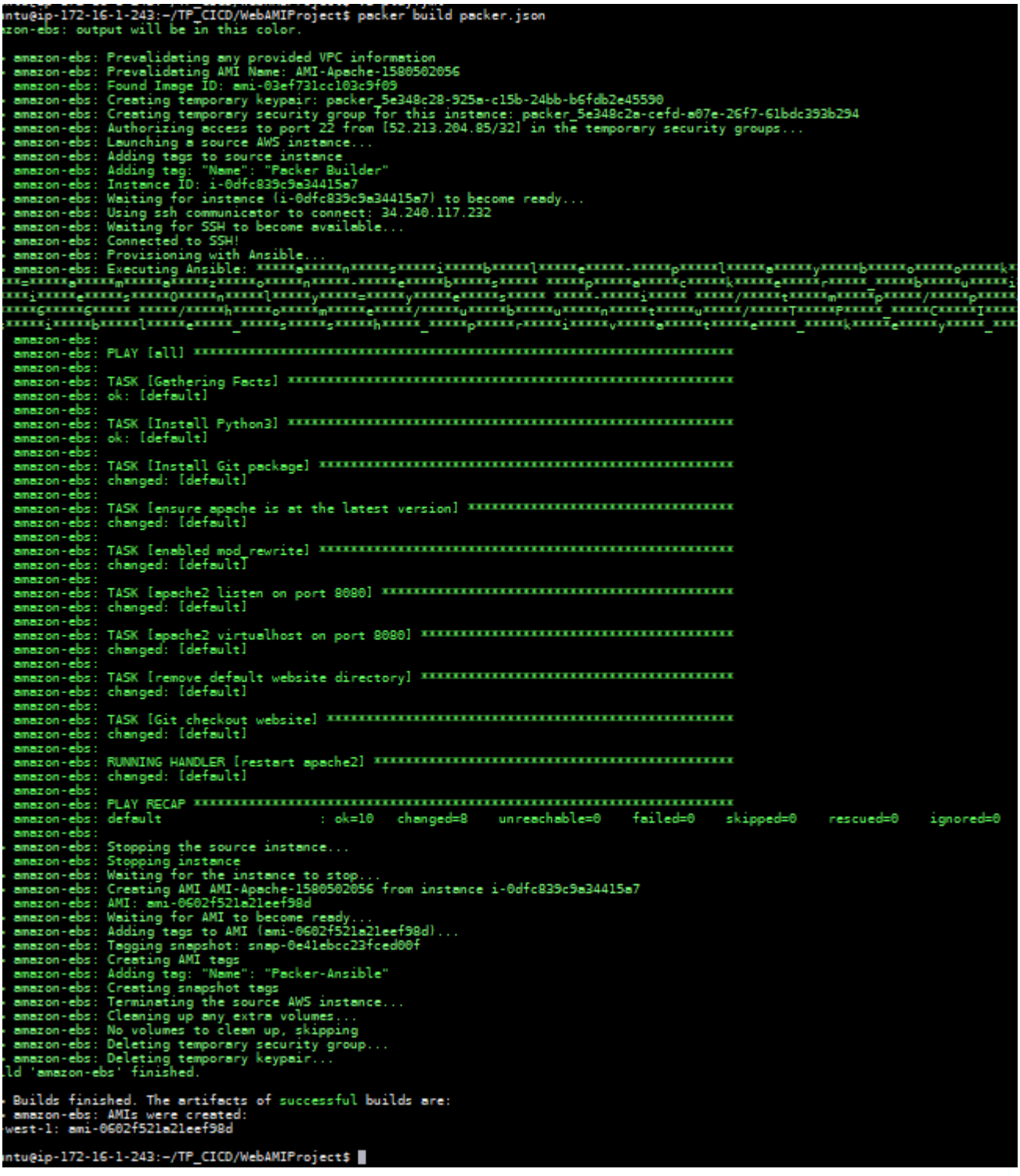
Modifiez les variables suivantes :

subnet_id: ID du Subnet dans lequel l'instance d'intégration sera déployée.
temporary_security_group_source_cidrs: IP de l'instance ToolsDevOps pour l'autoriser dans le « Security Group » temporaire créé par Packer le temps du déploiement.
base_ami: ID de l'AMI que vous avez utilisé pour développer votre Playbook ansible précédement.
associate_public_ip_address: « False » si vous voulez déployer dans le private subnet, mais nécessite un NAT dans ce cas (Voir TP1).

4- Executer le buildAMI.json

```
packer build buildAMI.json
```

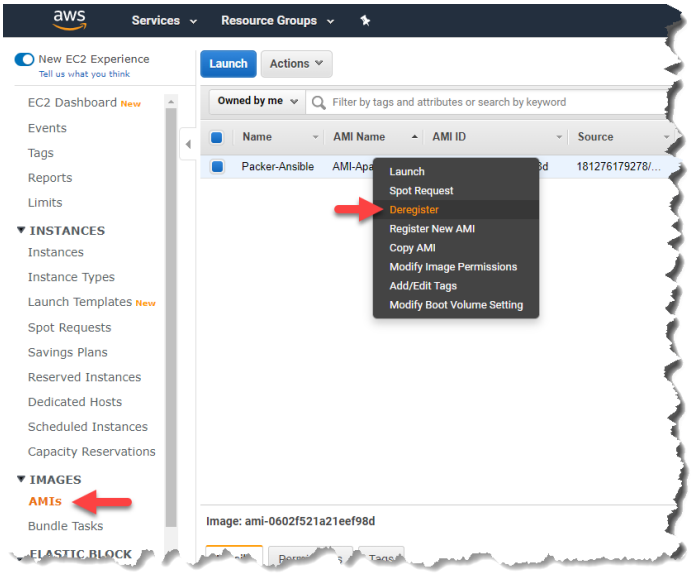
Lors de cette étape observez sur votre compte AWS l'instance qui est créée.



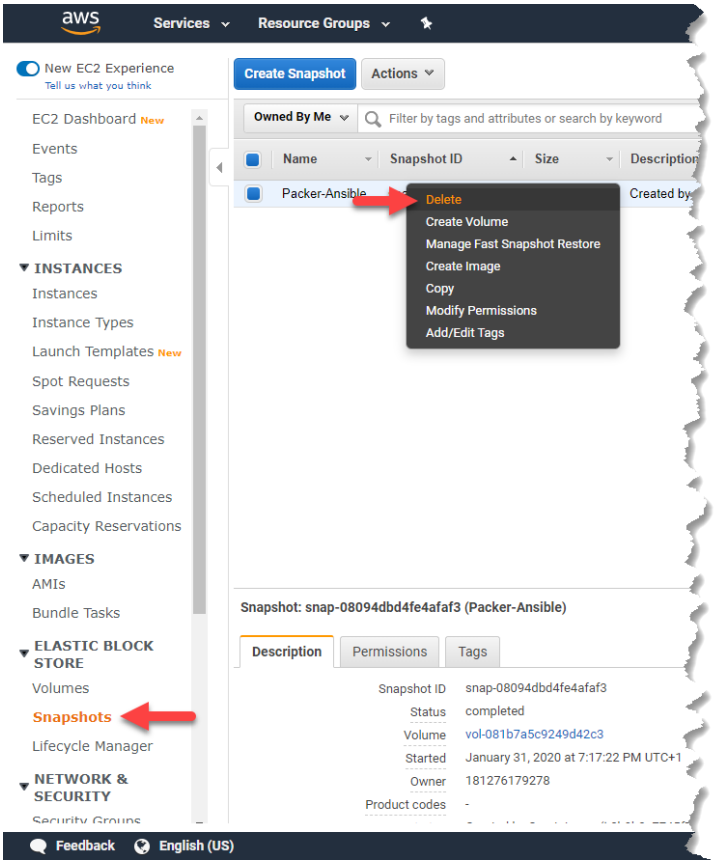
A la fin vous observerez dans votre compte un AMI vous appartenant, vous pouvez en créer une instance et vérifier que tous les composants déployés sont bien présents.

Nettoyer les Snapshots et AMIs qui ont été créés dans le compte pour éviter une facturation inutile.

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



Supprimer les « Snapshots »



Ne pas oublier de **résilier les instances** ainsi que de nettoyer les AMIs et snapshots dans **toutes les régions** !!!!