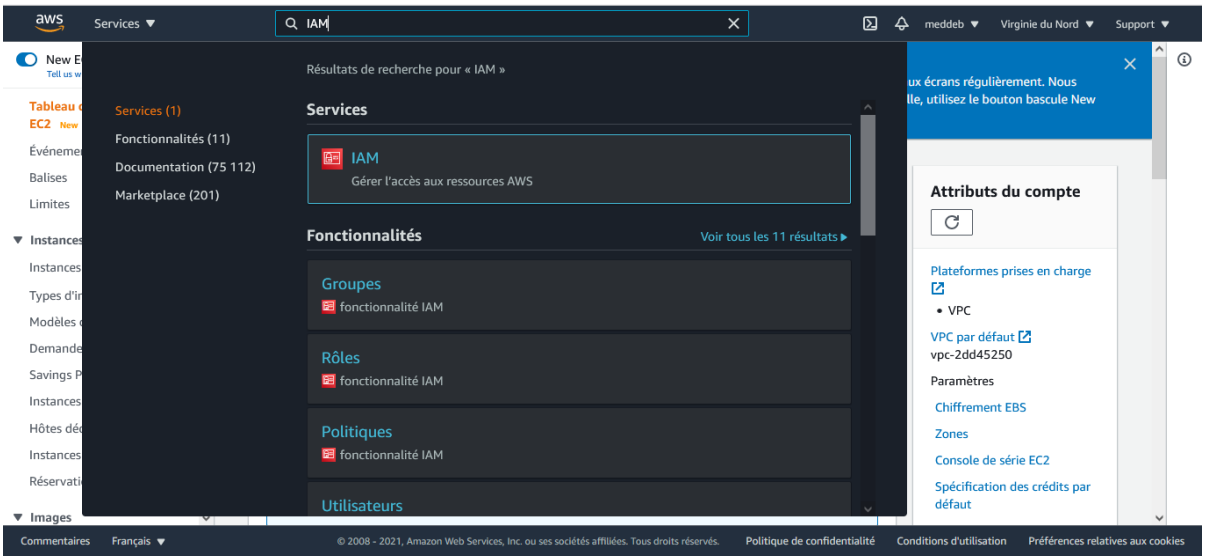
LAB 2 – Déploiements d’une instance sur AWS avec Terraform

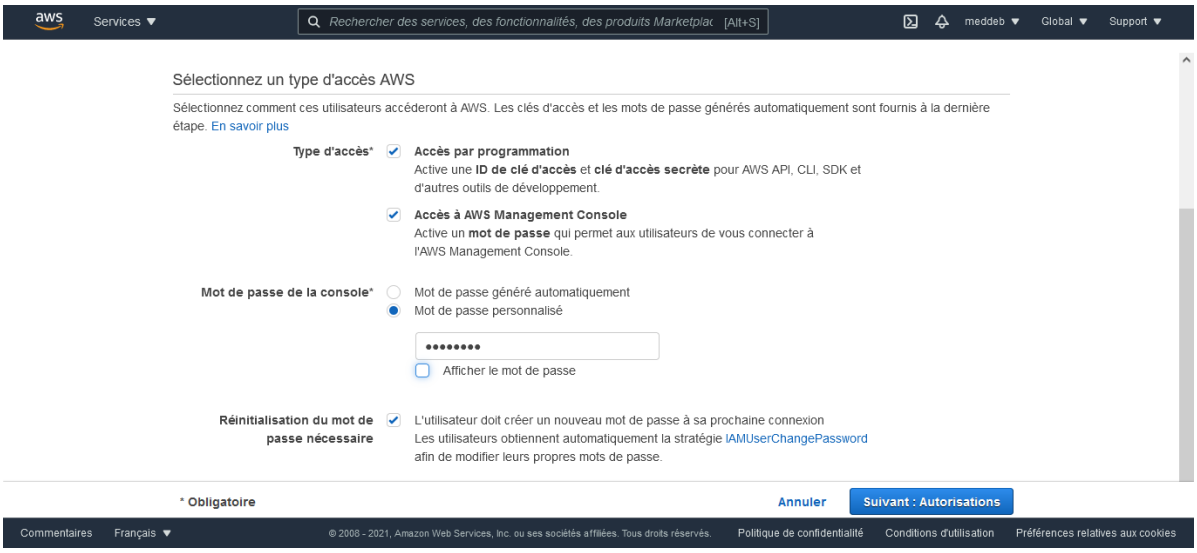
Configurer votre compte AWS

Lorsque vous vous inscrivez pour la première fois à AWS, vous vous connectez initialement en tant qu'utilisateur root. Ce compte d'utilisateur a des autorisations d'accès total, donc du point de vue de la sécurité, je vous recommande de ne l'utiliser que pour créer d'autres comptes d'utilisateurs avec des autorisations plus limitées.

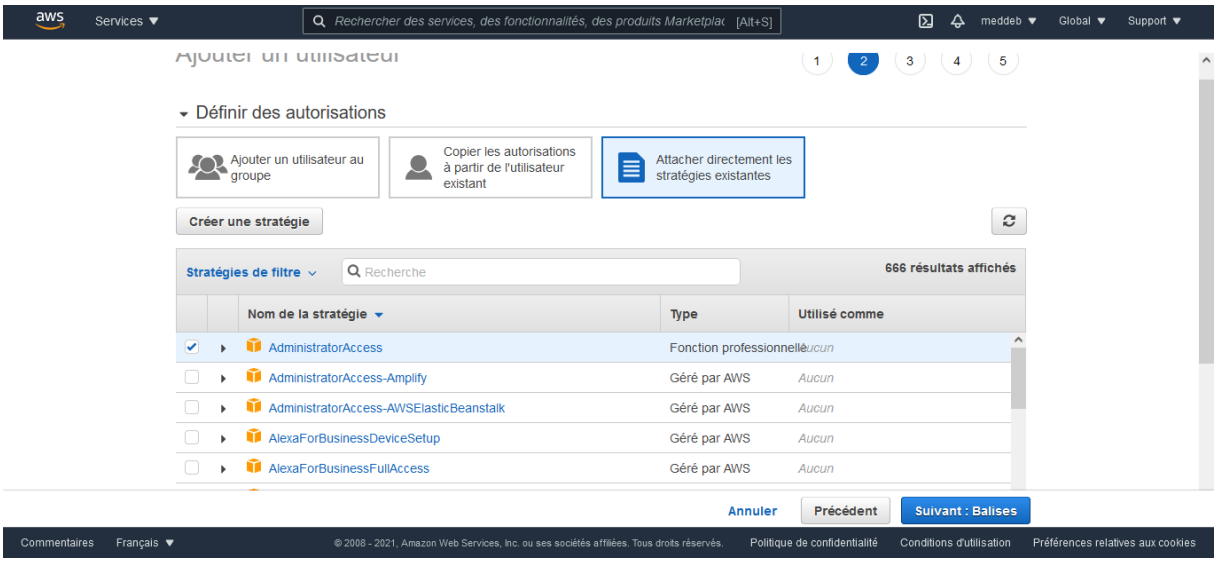
- Pour créer un compte d'utilisateur AWS plus limité, rendez-vous sur la console du service "Identity Access Management" (IAM) (https://console.aws.amazon.com/iam/home).



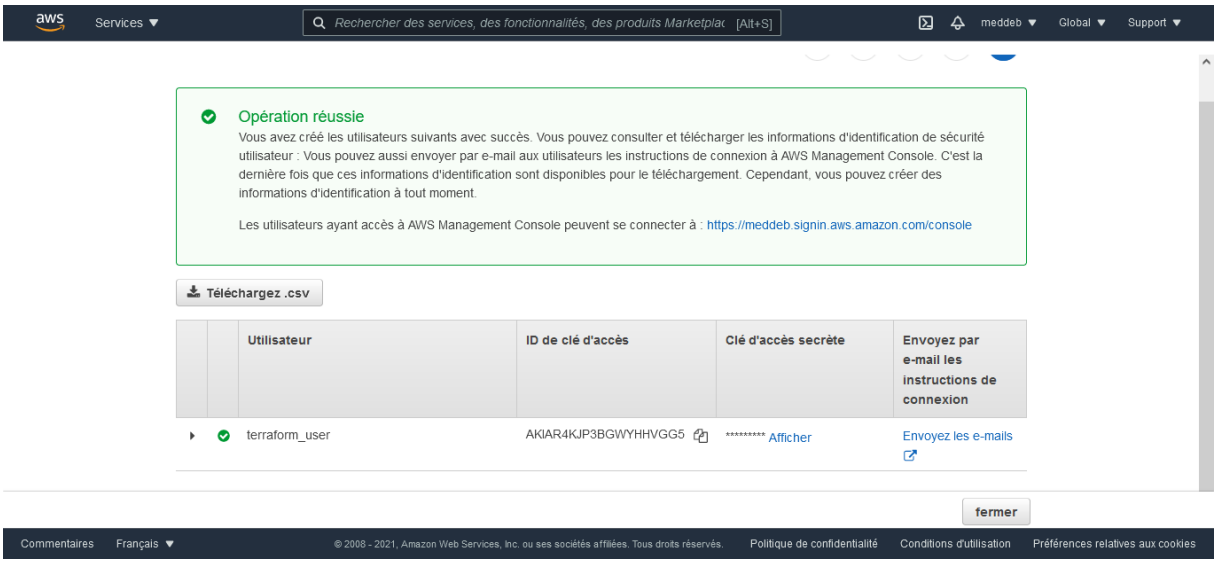
- cliquez sur "**Users**", puis sur le bouton bleu "**Add User**". Saisissez un nom pour l'utilisateur et assurez-vous de cocher l'option que " **Accès par programmation** " afin de générer des clés d'accès que nous utiliserons tout au long de ce cours. Cliquez ensuite sur le bouton "**Next:Permissions**" :



- Vous arrivez ensuite sur la page de configuration des autorisations de votre nouvel utilisateur. Sélectionnez la section "**Attach existing policies directly**", et choisissez la policy déjà préconfigurée d'AWS nommée " **AdministratorAccess "**



- Enfin, n'oubliez pas d'afficher et d'enregistrer votre clé d'accès AWS dans un endroit sécurisé :



Notre utilisateur est dorénavant prêt à être utilisé, nous aurons l'occasion de réévaluer la policy de notre utilisateur afin de lui assigner davantage de droits, pour l'instant nous lui offrant le strict minimum.

Personnaliser la connexion au tableau de bord

Cliquez sur Tableau de bord, puis modifiez l'URL de connexion pour les utilisateurs IAM de ce compte

⇨ https://meddeb.signin.aws.amazon.com/console

Puis Connectez-vous avec le compte IAM

Création de notre code Terraform

Le code Terraform est écrit dans un langage appelé **HCL** dans des fichiers avec l'extension .tf. Il s'agit d'un langage déclaratif, qui est également utilisé par les langages de configuration dans d'autres applications, et en particulier par d'autres produits HashiCorp (créateur de Terraform).

Notre objectif est donc de décrire l'infrastructure qu'on souhaite dans le langage HCL, et ensuite Terraform s'occupe de comment la créer.

Remarque : L'éditeur Visual code dispose d'un plugin Terraform pour valider le code écrit (https://marketplace.visualstudio.com/items?itemName=mauve.terraform)

Création du Provider (fournisseur)

La première étape de l'utilisation de Terraform consiste généralement à configurer le ou les providers (fournisseurs) que vous souhaitez utiliser. Pour ce faire, créez un fichier avec l'extension **.tf**, dans mon cas je vais créer un fichier nommé main.tf. Ensuite, mettez-y le code suivant:

```
provider "aws" {
    region = "us-east-2"
    access_key = "votre-clé-dacces"
    secret_key = "votre-clé-secrète"
}
```

Cela indique à Terraform que vous allez utiliser le fournisseur AWS et que vous souhaitez déployer votre infrastructure dans la région "us-east-2". Vous devez également spécifier la paire de clés de votre utilisateur générée auparavant.

Création de la Resource (ressource)

Pour chaque fournisseur, il existe de nombreux types de ressources que vous pouvez créer, tels que des serveurs, des bases de données, des équilibreurs de charge, etc.

Avant de déployer des ressources complexes, voyons d'abord comment déployer une instance de calcul (machine virtuelle). Dans le jargon AWS, un serveur/machine virtuelle est appelé une instance EC2 . Ajoutez le code suivant à votre fichier main.tf :

```
resource "aws_instance" "my_ec2_instance" {
    ami = "ami-07c1207a9d40bc3bd"
    instance_type = "t2.micro"
}
```

La syntaxe générale d'une ressource Terraform est la suivante :

```
resource "<FOURNISSEUR>_<TYPE>" "<NOM>" {
    [CONFIG …]
}
```

- **FOURNISSEUR** : c'est le nom d'un fournisseur (ici le provider "aws").
- **TYPE** : c'est le type de ressources à créer dans ce fournisseur (ici c'est une instance ec2)
- **NOM** : c'est un identifiant que vous pouvez utiliser dans le code Terraform pour faire référence à cette ressource (ici "my_ec2_instance")
- **CONFIG** : se compose de un ou plusieurs arguments spécifiques à cette ressource, dans notre cas :
    - ami : c'est l'acronyme d'"Amazon Machine Image" (AMI) , c'est donc l'image qui sera exécutée sur notre instance EC2. Vous pouvez trouver des AMI gratuites et payantes sur AWS Marketplace ou créer les vôtres directement depuis la console AWS. Dans notre cas, nous utilisons l'identifiant "ami-07c1207a9d40bc3bd" qui est une AMI Ubuntu 18.04 (Attention l'identifiant peut être modifié avec le temps !). Cette AMI est gratuite et éligible à l'offre gratuite d'AWS.
    - instance_type : Type d'instance EC2 à exécuter. Chaque type d'instance EC2 fournit une quantité différente de CPU, de mémoire, d'espace disque et de capacité réseau (plus d'informations ces différences sur cette page). Dans notre cas, nous utilisons le type t2.micro, qui fait partie du niveau gratuit AWS, et qui a comme caractéristiques 1 vCPU, ainsi que 1 Go de mémoire.

Création de notre instance ec2

En combinant les exemples précédents, nous nous retrouvons avec le code suivant :

```
provider "aws" {
    region = "us-east-2"
    access_key = "votre-clé-dacces"
    secret_key = "votre-clé-secrète"
}

resource "aws_instance" "my_ec2_instance" {
    ami = "ami-07c1207a9d40bc3bd"
    instance_type = "t2.micro"
}
```

Depuis un terminal, accédez au dossier dans lequel vous avez créé votre fichier main.tf et exécutez la commande suivante :

```
[root@terraform ~]# terraform init

Initializing the backend...
```

```
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.44.0...
- Installed hashicorp/aws v3.44.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

terraform init demande à Terraform de d'abord scanner votre code, qui déterminera quel fournisseur vous utilisez et téléchargera le code pour vous.

Le code du fournisseur sera téléchargé dans le dossier **.terraform** qui est le répertoire de travail de Terraform. Vous devez exécuter cette commande **init** chaque fois que vous démarrez avec du nouveau code Terraform.

Maintenant que vous avez téléchargé le code du fournisseur, exécutez la commande suivante :

```
[root@terraform ~]# terraform plan

Terraform used the selected providers to generate the following execution plan. Resource
actions are indicated with
the following symbols:
  + create

Terraform will perform the following actions:
  # aws_instance.my_ec2_instance will be created
  + resource "aws_instance" "my_ec2_instance" {
      + ami                                  = "ami-07c1207a9d40bc3bd"

   ...
Plan: 1 to add, 0 to change, 0 to destroy.
```

```
──────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take
exactly these actions if

you run "terraform apply" now.
```

La commande **plan** est utilisée pour créer un plan d'exécution. Elle détermine les actions nécessaires pour atteindre **l'état souhaité**, spécifié dans les fichiers de configuration **sans** les exécuter.

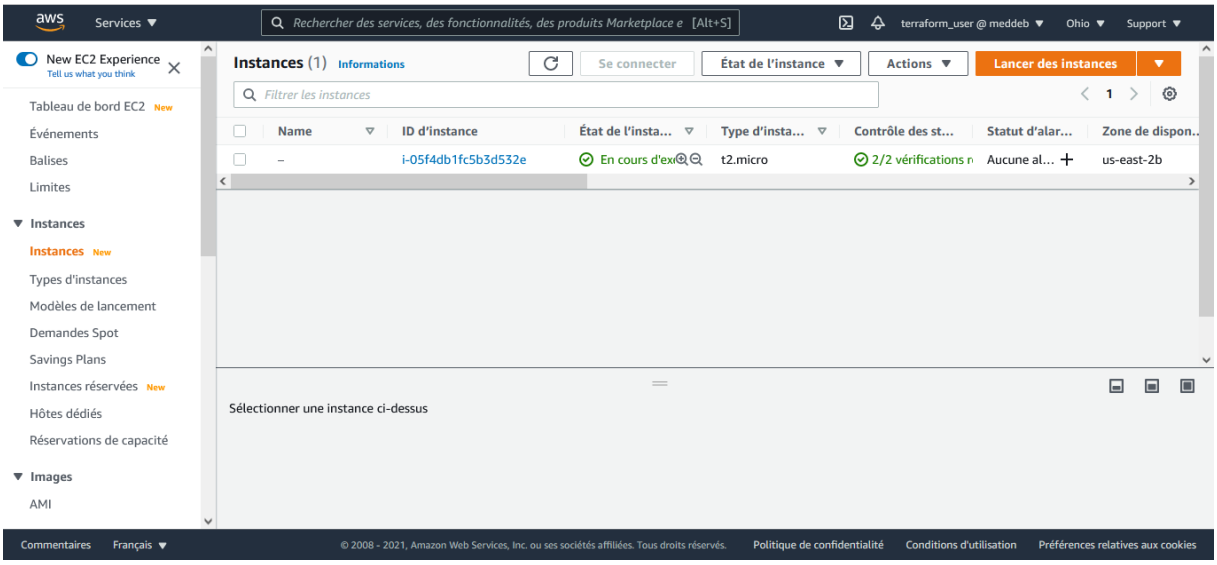
Elle n'est pas obligatoire mais ça reste un excellent moyen de vérifier vos modifications avant de les diffuser. La sortie de la commande plan ressemble un peu à la sortie de la commande Linux diff, avec le signe + qui indique les ressources qui vont être créées, le signe - pour les ressources qui vont être supprimées et enfin le signe ~ pour les ressources qui vont être modifiées.

Maintenant, pour véritablement créer notre ressource Terraform, exécutez la commande suivante :

```
[root@terraform ~]# terraform apply

Terraform used the selected providers to generate the following execution plan. Resource
actions are indicated with

the following symbols:

  + create

Terraform will perform the following actions:

  # aws_instance.my_ec2_instance will be created

  + resource "aws_instance" "my_ec2_instance" {

      + ami                                  = "ami-07c1207a9d40bc3bd"

...

Plan: 1 to add, 0 to change, 0 to destroy.


Do you want to perform these actions?

  Terraform will perform the actions described above.

  Only 'yes' will be accepted to approve.


  Enter a value: yes

aws_instance.my_ec2_instance: Creating...

aws_instance.my_ec2_instance: Still creating... [10s elapsed]

aws_instance.my_ec2_instance: Still creating... [20s elapsed]

aws_instance.my_ec2_instance: Still creating... [30s elapsed]

aws_instance.my_ec2_instance: Creation complete after 30s [id=i-05f4db1fc5b3d532e]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Vous remarquerez que la commande **apply** vous affiche la même sortie que la commande **plan** et vous demande de confirmer, si vous souhaitez réellement poursuivre avec ce plan, alors tapez "**yes**" et appuyez sur Entrée pour déployer votre instance EC2.

Félicitations, vous venez de déployer votre premier serveur avec Terraform ! Pour vérifier cela, vous pouvez vous connecter à la console EC2 et **assurez-vous de bien sélectionner la région "us-east-2" (la zone Ohio)**, vous verrez alors quelque chose comme ceci :



Les modifications de ressources

Terraform garde une trace de toutes les ressources qu'il a déjà créées.

Si on rajoute par exemple une information, Terraform saura détecter que votre instance EC2 existe déjà, et vous montrera la différence entre ce qui est actuellement déployé et ce qu'il y a actuellement dans votre code Terraform. Pour vous prouver que c'est bien le cas, nommons notre instance en créant **une balise avec comme clé Name** et comme valeur **terraform-test**, ce qui nous donne le code suivant :

```
provider "aws" {
    region = "us-east-2"
    access_key = "votre-clé-daccès"
    secret_key = "votre-clé-secrète"
}

resource "aws_instance" "my_ec2_instance" {
    ami = "ami-07c1207a9d40bc3bd"
    instance_type = "t2.micro"
    tags = {
```

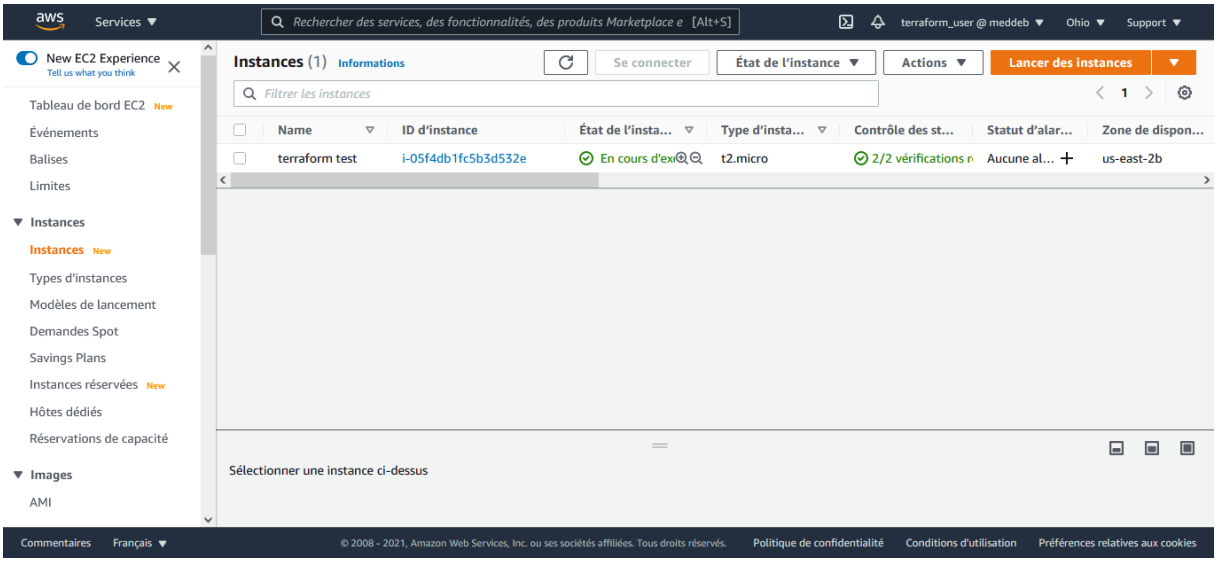
```
        Name = "terraform test"
    }
}
```

Exécutons ensuite notre code :

```
[root@terraform ~]# terraform init && terraform apply
Initializing the backend...
...
aws_instance.my_ec2_instance: Refreshing state... [id=i-05f4db1fc5b3d532e]
Note: Objects have changed outside of Terraform

Terraform detected the following changes made outside of Terraform since the last
"terraform apply":

  # aws_instance.my_ec2_instance has been changed
  ~ resource "aws_instance" "my_ec2_instance" {
        id                                   = "i-05f4db1fc5b3d532e"
      + tags                                     = {}
        # (28 unchanged attributes hidden)
        # (5 unchanged blocks hidden)
    }
...
Terraform will perform the following actions:

  # aws_instance.my_ec2_instance will be updated in-place
  ~ resource "aws_instance" "my_ec2_instance" {
        id                                   = "i-05f4db1fc5b3d532e"
      ~ tags                                 = {
          + "Name" = "terraform test"
        }
      ~ tags_all                             = {
          + "Name" = "terraform test"
        }
...
Plan: 0 to add, 1 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Enter a value: yes
aws_instance.my_ec2_instance: Modifying... [id=i-05f4db1fc5b3d532e]
aws_instance.my_ec2_instance: Modifications complete after 7s [id=i-05f4db1fc5b3d532e]
```

```
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

Vérifiez les modifications apportées sur votre instance sur le tableau de bord de votre compte AWS.



Exercice d'application 1

- Créez un nouveau dossier « terraform-exapp-1 » pour cet exercice
- Récupérez le secret et access key de votre compte (dans les paramètres sécurité de votre compte dans IAM)
- Créez un paire de clé dans EC2 et nommez cette clé devops --<votre prenom >, un fichier devops --<votre prenom pem sera téléchargé
- Créez une fichier ec2.tf dans un répertoire nommé tp 2
- Renseignez les informations permettant de créer une VM avec l'image centos suivante : centos7 minimal v20190919.0.0 (ami0083662ba17882949)
- Vérifiez que votre instance est bien créée et observez le contenu de fichier tfstate
- Modifiez le fichier ec2.tf afin d'y inclure le tag de votre instance : "Name: ec2 votre prenom
- Appliquez la modification et constatez les changements apportés ainsi que dans le fichier tfstate
- Supprimez votre ec2