Automating Infrastructure using Terraform

Excercise description:

> Nowadays, infrastructure automation is critical. We tend to put the most emphasis on software development processes, but infrastructure deployment strategy is just as important. Infrastructure automation not only aids disaster recovery, but it also facilitates testing and development. Your organization is adopting the DevOps methodology and in order to automate provisioning of infrastructure there's a need to setup a centralised server for Jenkins.

> Terraform is a tool that allows you to provision various infrastructure components. Ansible is a platform for managing configurations and deploying applications. It means you'll use Terraform to build a virtual machine, for example, and then use Ansible to instal the necessary applications on that machine.

> Considering the Organizational requirement you are asked to automate the infrastructure using Terraform first and install other required automation tools in it.

Tools required: Terraform, AWS account with security credentials, Keypair

Expected Deliverables:

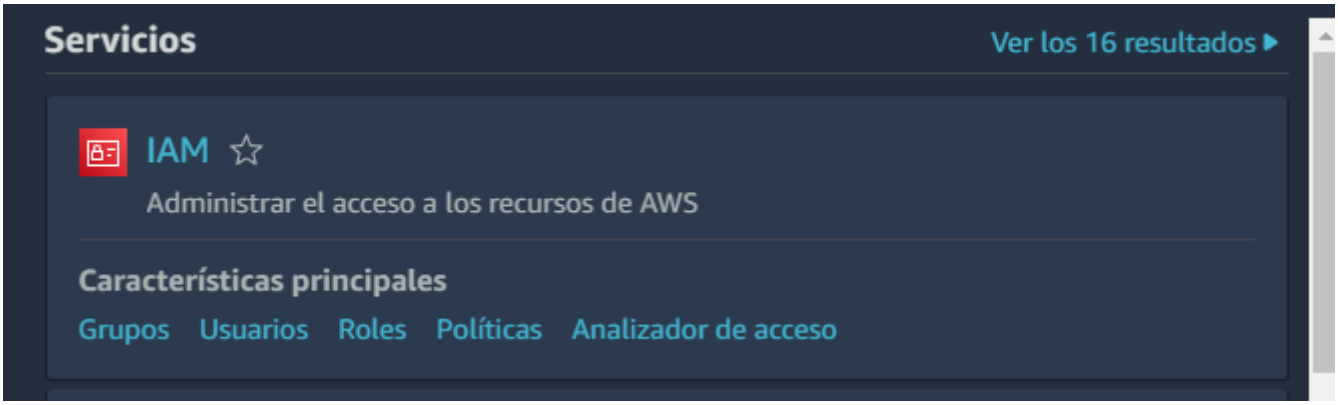
> - Launch an EC2 instance using Terraform
> - Connect to the instance
> - Install Jenkins, Java and Python in the instance

Author: Francisco Pedraza Vázquez

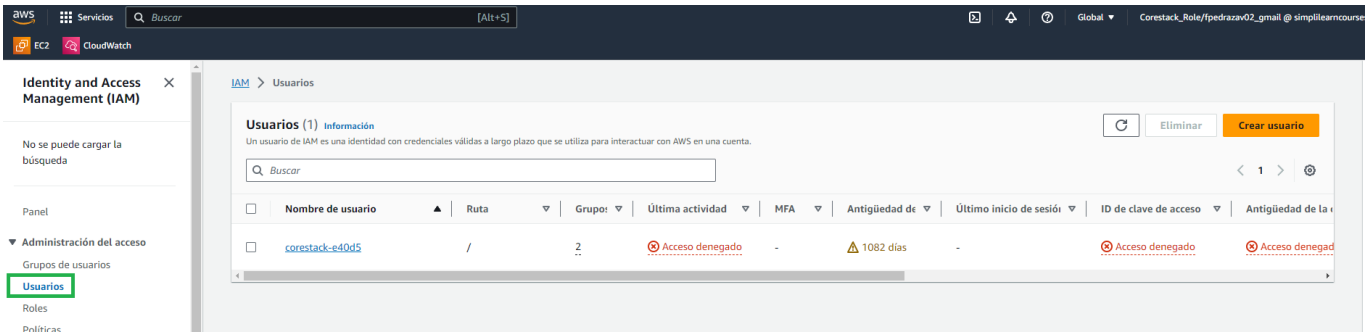
1. Create `AWS` credential Keys

First we conect to AWS, and search for IAM services.

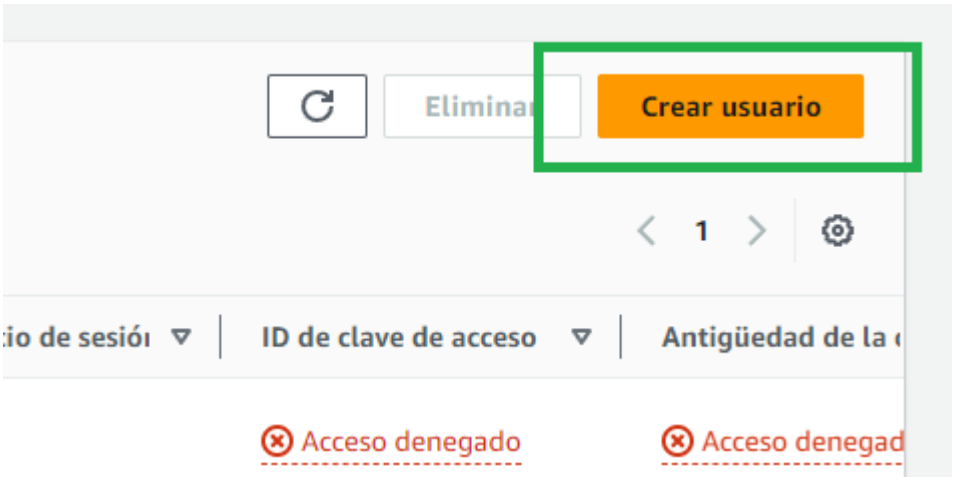
> `IAM` , or identity access management services is one of the many services provided by AWS in which we can manage perms, create users and generate key-access pairs.

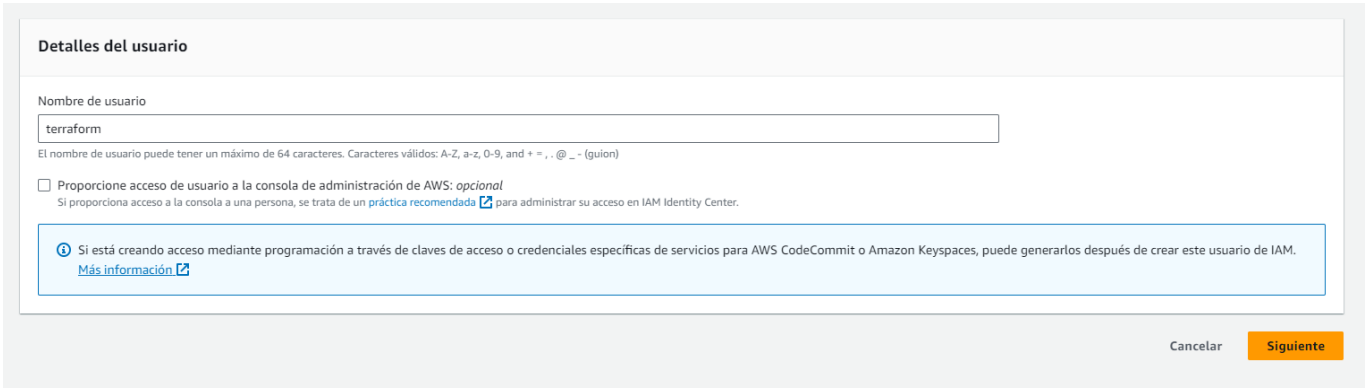


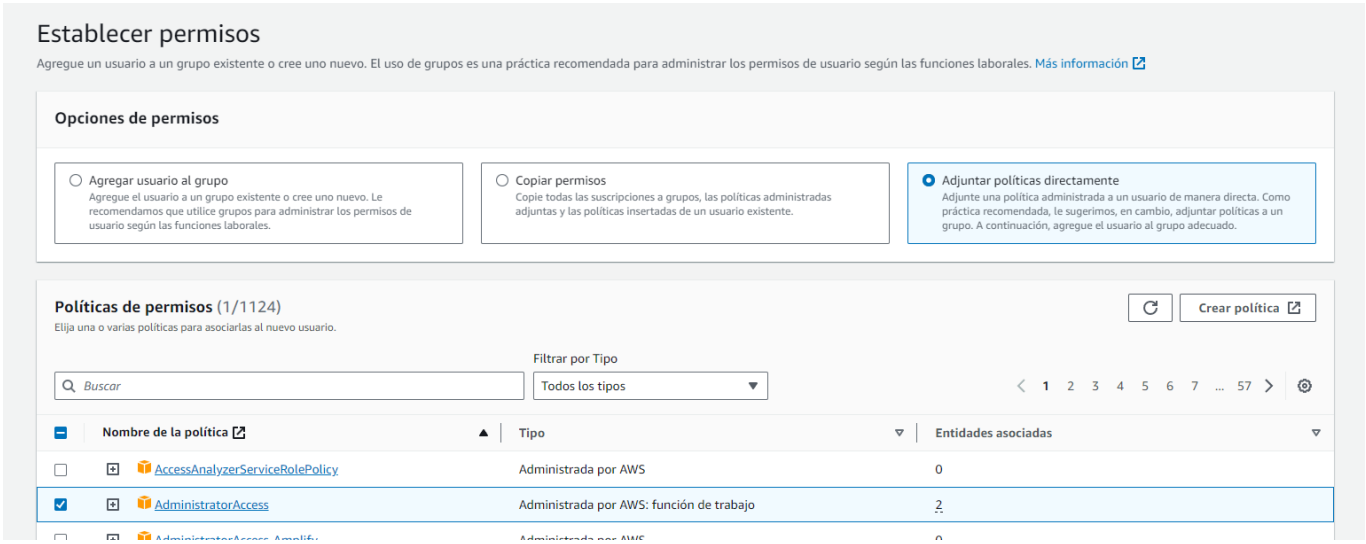
Click on Users

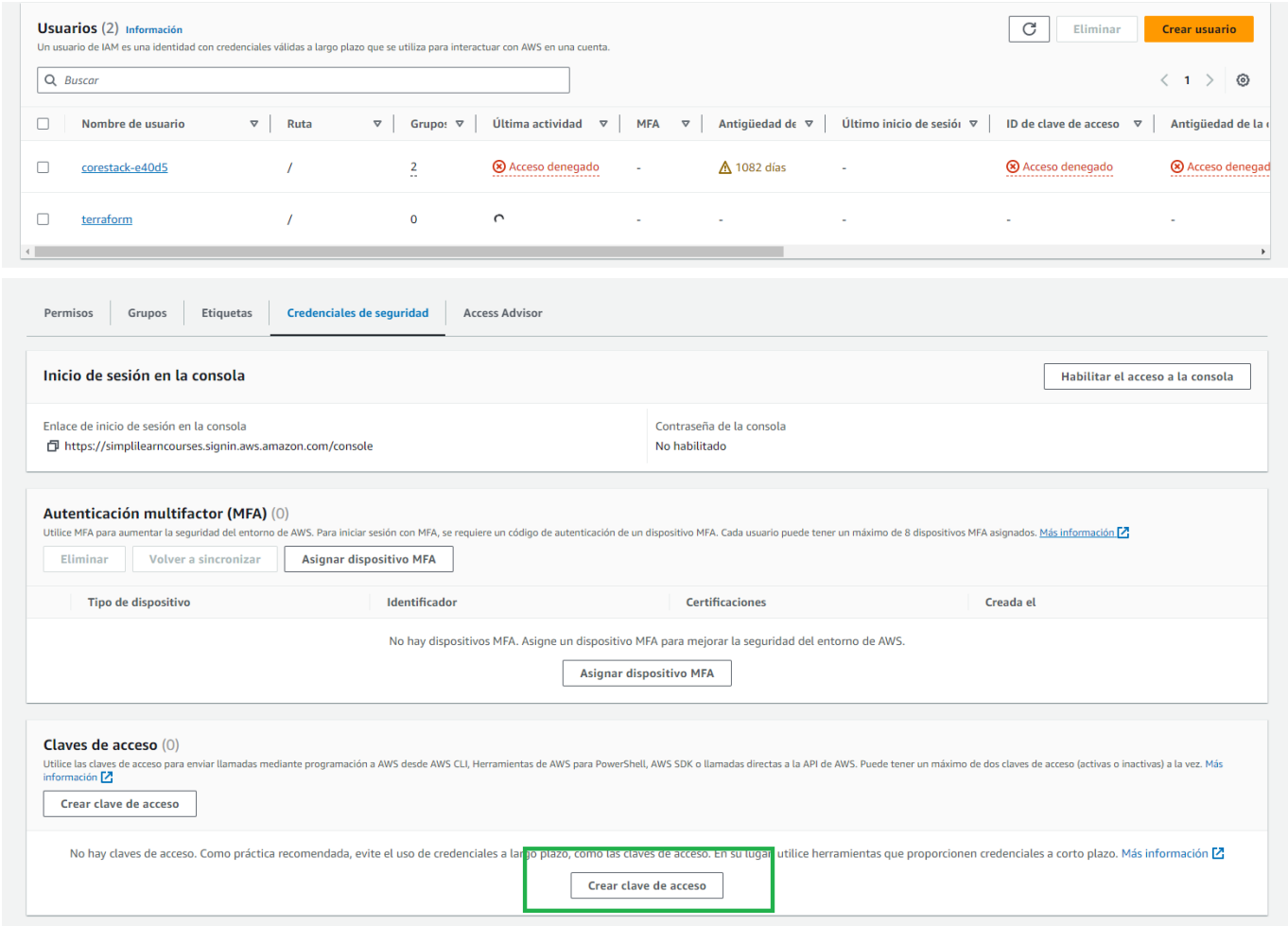


Create the new User with administrator perms

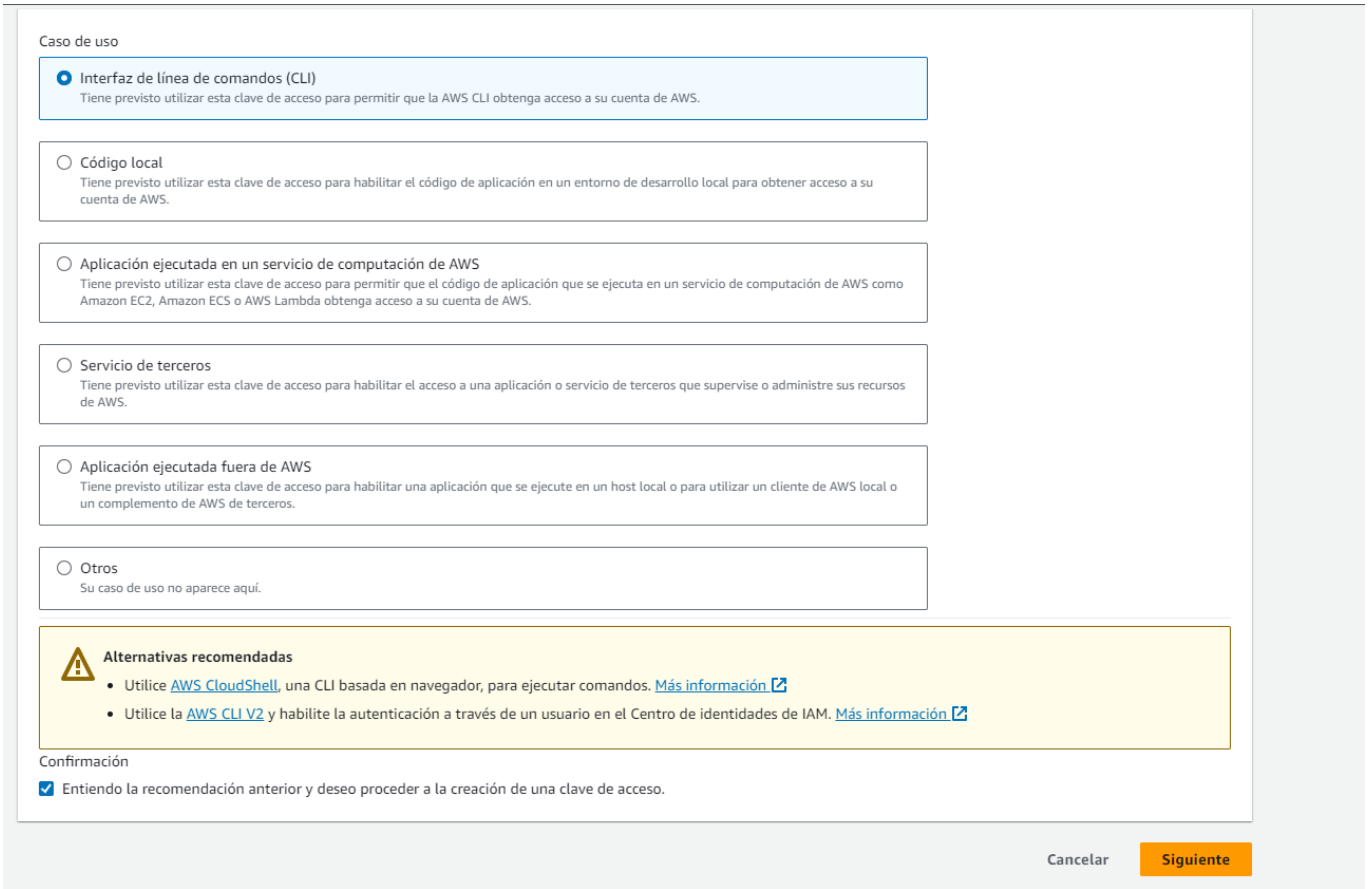




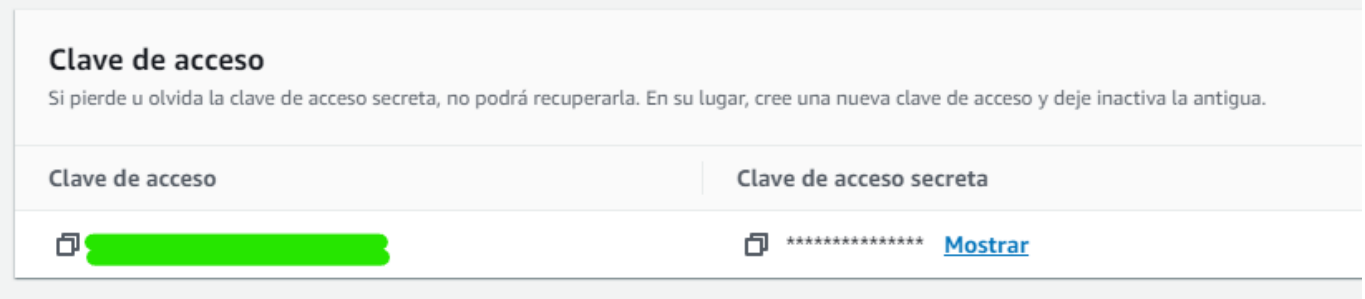




We enter **Secutiry Credentials** and create a new **CLI** set of credentials



> We write down the **secret key** and **access key** provided



2. Install `Terraform`/ Update `Terraform`

Since we count this scenario as a new set up, we should not have `terraform` installed.

We check terraform instalation status.

```
fpedrazav02gmai@ip-172-31-27-37:~/Desktop$ terraform --version
Terraform v1.1.6
on linux_amd64

Your version of Terraform is out of date! The latest version
is 1.5.6. You can update by downloading from https://www.terraform.io/downloads.
html
```

We can use a simple bash script as root to install and update our terraform instalation provided the user has the correct perms in the sudoers file.

```
wget https://releases.hashicorp.com/terraform/1.5.6/terraform_1.5.6_linux_386.zip

unzip terraform_1.5.6_linux_386.zip

mv terraform /usr/local/bin
```

Check installation.

```
Terraform v1.5.6
on linux_386
```

3. Gather `AWS` needed data.

We connect to AWS and gather the next data.

> **Ami ID:** ami-051f7e7f6c2f40dc1
> **Region:** N.Virginia
> **Instance type:** t2.micro
> **Private Key:** -
> **Access Key:** -

4. Write `Terraform`

Since we have all the data availiable, we can now start coding our `terraform` code.

1) Initialize Terraform and download cloud providers dependencies

Prior to crafting a whole script we can add a simple *providers block* to download any cloud provider module needed.

```
provider "aws" {
  region     = "us-east-1"
  access_key = "my-access-key"
  secret_key = "my-secret-key"
}
```

Then we run `terraform init` to start and donwload this said modules.

```
fpedrazav02gmai@ip-172-31-27-37:~/Desktop/terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.15.0...
- Installed hashicorp/aws v5.15.0 (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.
```

We can then write our security rules as well as inbound and outbound traffic in our code.

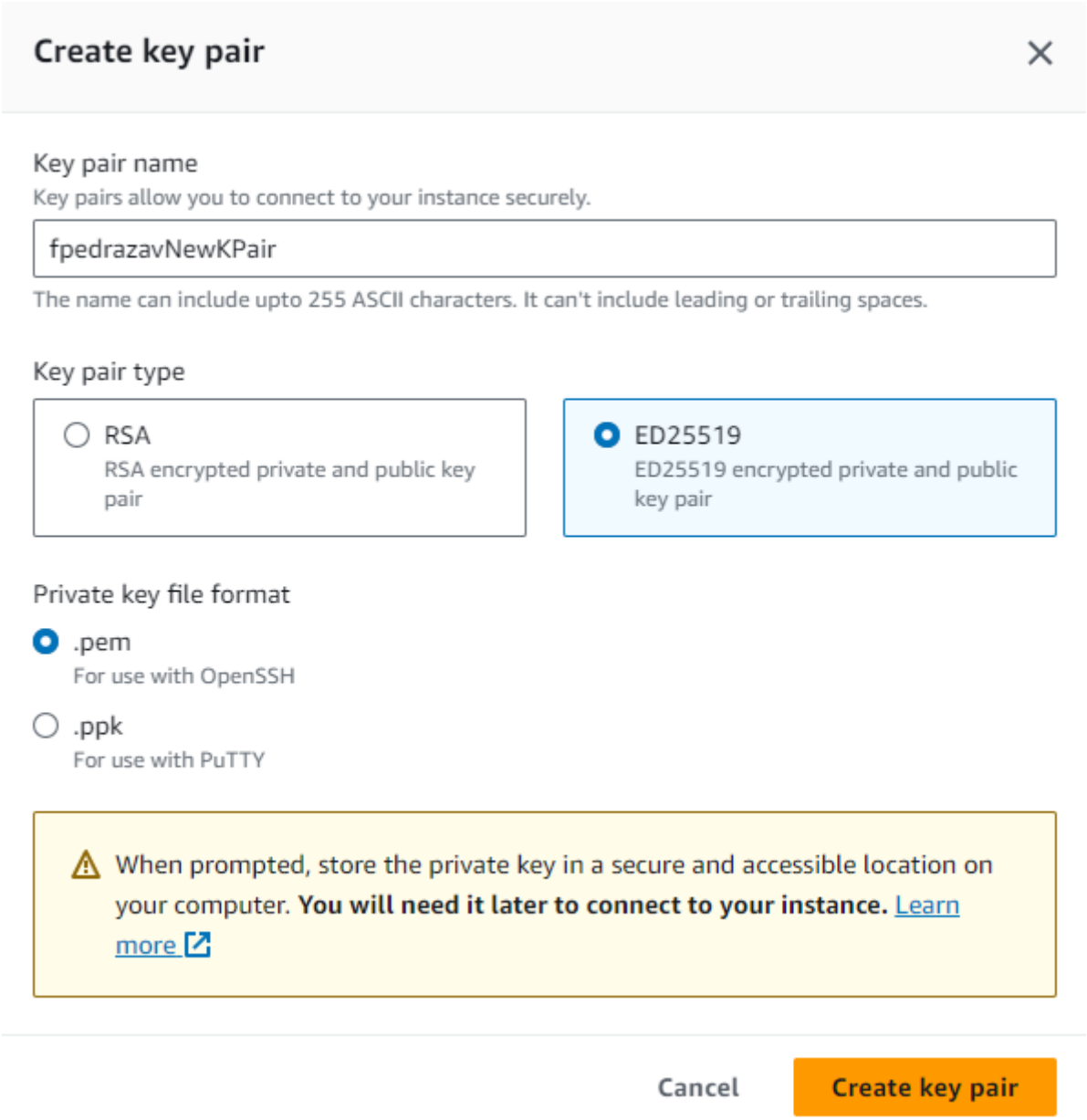
```
resource "aws_security_group" "SecurityFwRules" {

    name = "allow_tls"

    description = "Allow TLS inbound traffic"
}

ingress {
    description = "SSH connection from VPC"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}
```

```
  ingress {
      description = "Http connection from VPC"
      from_port   = 8080
      to_port     = 8080
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
      from_port   = 0
      to_port     = 0
      protocol    = "-1"
      cidr_blocks = ["0.0.0.0/0"]
      ipv6_cidr_blocks = ["::/0"]
  }
```

With this simple code we are allowing both *ssh* on port 22 to connect to the VM (ingress) and the traffic to the port 8080 via *http*. Which we will use to host Jenkins web portal.

Also, we allow outbound traffic to any IP via any protocol.

After this we can generate a new key pair. In the portal, if launch the `key pairs` section, and generate one with the next settings.



Next, we add the `aws_instance` resource block in order to connect the key and what type of intance we want to create.

```
resource "aws_instance" "AWS-TERRA-LAUNCH" {
  ami           = "ami-051f7e7f6c2f40dc1"
  instance_type = "t2.micro"
  tags = {
    Name = "Project1"
  }
   key_name = "fpedrazavNewKPair"
}
```

In this block we can add the tag `user_data` which lets us excute commands on the created machine. We will use this to insert a bash script to install the required dependencies.

```
    user_data = <<-EOF
```

```
        #!/bin/bash

          sudo yum install git -y

          sudo amazon-linux-extras install java-openjdk11 -y

          sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-
  stable/jenkins.repo

          sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

          sudo yum install jenkins -y

          sudo systemctl start jenkins

          sudo apt install python3

      EOF
```

Finally, with the resource `aws_network_intergace_sg_attachment` we match our security group with our instance.

```
resource "aws_network_interface_sg_attachment" "sg_attachment1" {
    security_group_id = aws_security_group.SecurityFwRules.id
    network_interface_id = aws_instance.AWS-TERRA-
LAUNCH.primary_network_interface_id
}
```

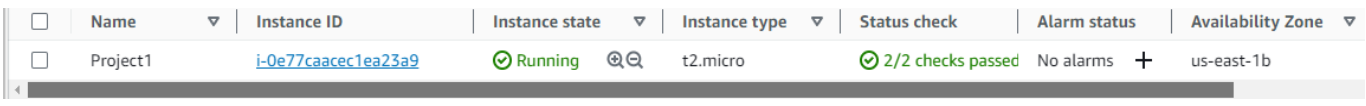
Once we finish our `terraform` main file we can apply it via CLI with `terraform apply`.

```
Plan: 3 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.AWS-TERRA-LAUNCH: Creating...
aws_security_group.SecurityFwRules: Creating...
aws_security_group.SecurityFwRules: Creation complete after 2s [id=sg-00305623a8bab3faf]
aws_instance.AWS-TERRA-LAUNCH: Still creating... [10s elapsed]
aws_instance.AWS-TERRA-LAUNCH: Still creating... [20s elapsed]
aws_instance.AWS-TERRA-LAUNCH: Still creating... [30s elapsed]
aws_instance.AWS-TERRA-LAUNCH: Creation complete after 32s [id=i-0e77caacec1ea23a9]
aws_network_interface_sg_attachment.sg_attachment1: Creating...
aws_network_interface_sg_attachment.sg_attachment1: Creation complete after 1s [id=sg-00305623a8bab3faf_eni-066039a3ebcc024f1]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

We see how our 3 resources are created as well as the interfaces attached.

If we check on AWS.

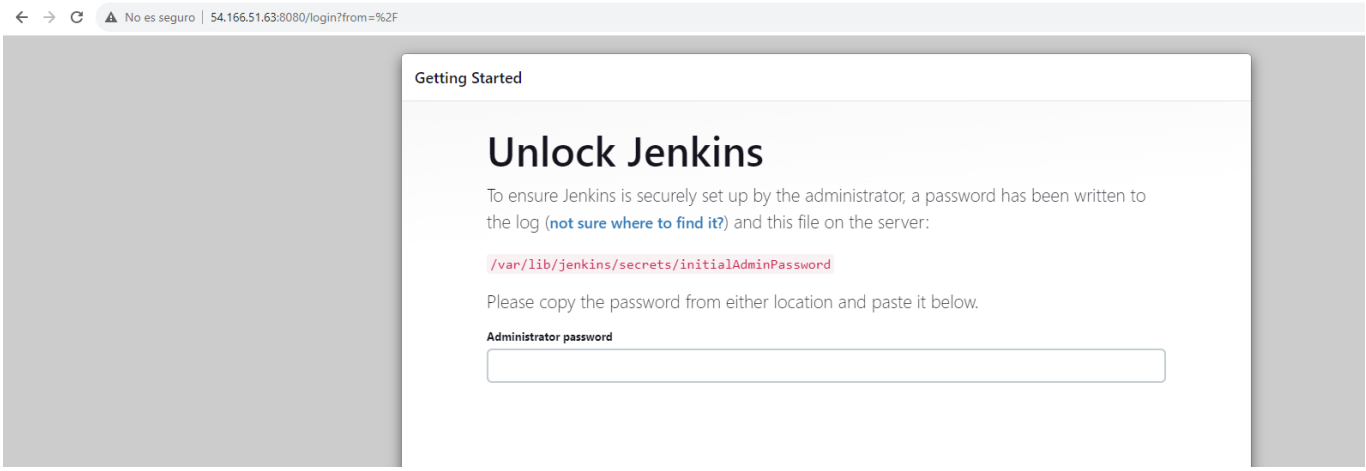
We can see Project1 running.



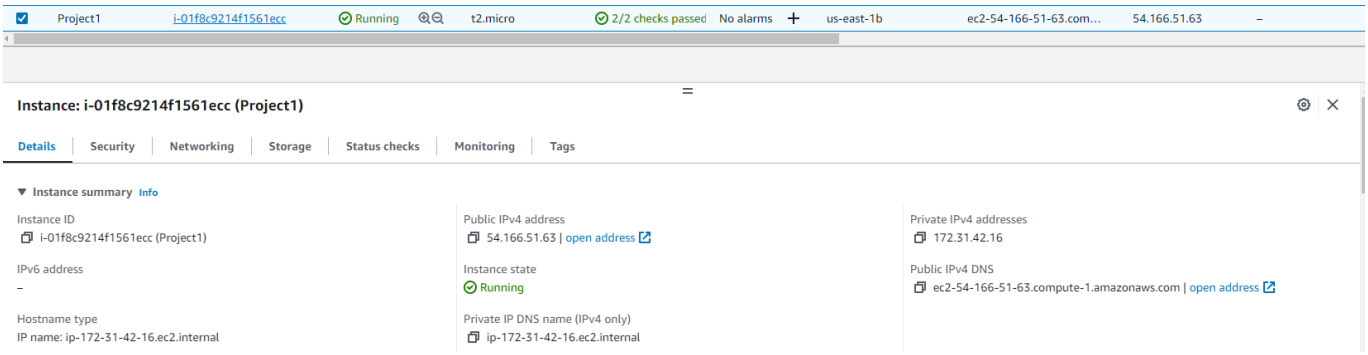
If we take a peek inside to check if everything is installed we can see it has all dependencies

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-42-16 ~]$ git --version
git version 2.40.1
[ec2-user@ip-172-31-42-16 ~]$ python --version
Python 2.7.18
[ec2-user@ip-172-31-42-16 ~]$ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; disabled; vendor preset: disabled)
   Active: activating (start) since Fri 2023-09-01 22:26:32 UTC; 48s ago
 Main PID: 4241 (java)
   CGroup: /system.slice/jenkins.service
           └─4241 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: *************************************************************
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: *************************************************************
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: *************************************************************
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: Jenkins initial setup is required. An admin user has been created and a password generated.
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: Please use the following password to proceed to installation:
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: 2acac2d96bd044df9c6ba5e66e80bece
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: *************************************************************
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: *************************************************************
Sep 01 22:26:49 ip-172-31-42-16.ec2.internal jenkins[4241]: *************************************************************
[ec2-user@ip-172-31-42-16 ~]$ java --version
openjdk 11.0.20 2023-07-18 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.20.0.8-1.amzn2.0.1) (build 11.0.20+8-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.20.0.8-1.amzn2.0.1) (build 11.0.20+8-LTS, mixed mode, sharing)
[ec2-user@ip-172-31-42-16 ~]$
```

We can also see Jenkins web portal is installed and running in port 8080.



Finally the Instance IP settings are:



Check on GitHub