Project submitter : Vaibhav Kulkarni

Project objective : Automating Infrastructure using Terraform

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

Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It enables users to define and manage their cloud infrastructure, such as virtual machines, networks, storage, and more, using a declarative configuration language. Terraform allows you to specify your desired infrastructure state in code, which it then uses to create, modify, and manage resources across various cloud providers like AWS, Azure, Google Cloud, and more. This approach enhances automation, consistency, and scalability in managing complex infrastructure setups.

First step is to setup Terraform on the lab. To setup the terraform on simiplilen lab we can use following commands :

1. Set up terraform on the Simplilearn lab:

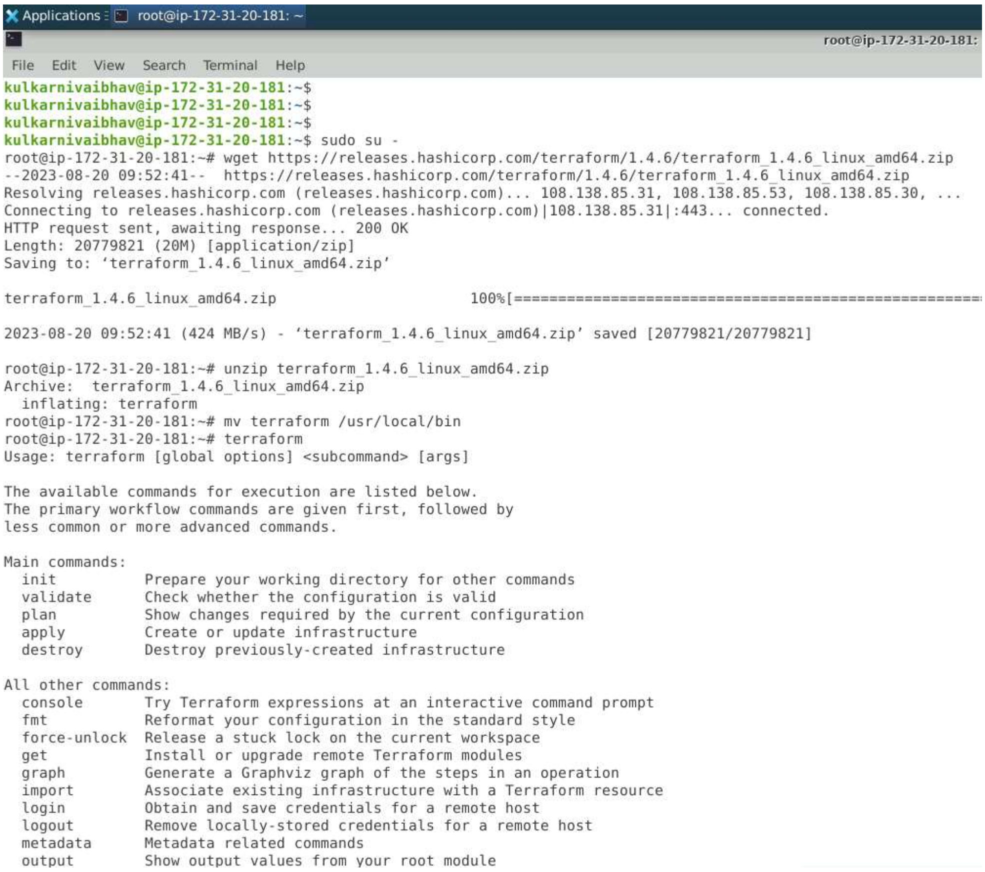
# sudo su -

wget https://releases.hashicorp.com/terraform/1.4.6/terraform_1.4.6_linux_amd64.zip

unzip terraform_1.4.6_linux_amd64.zip

mv terraform /usr/local/bin

terraform

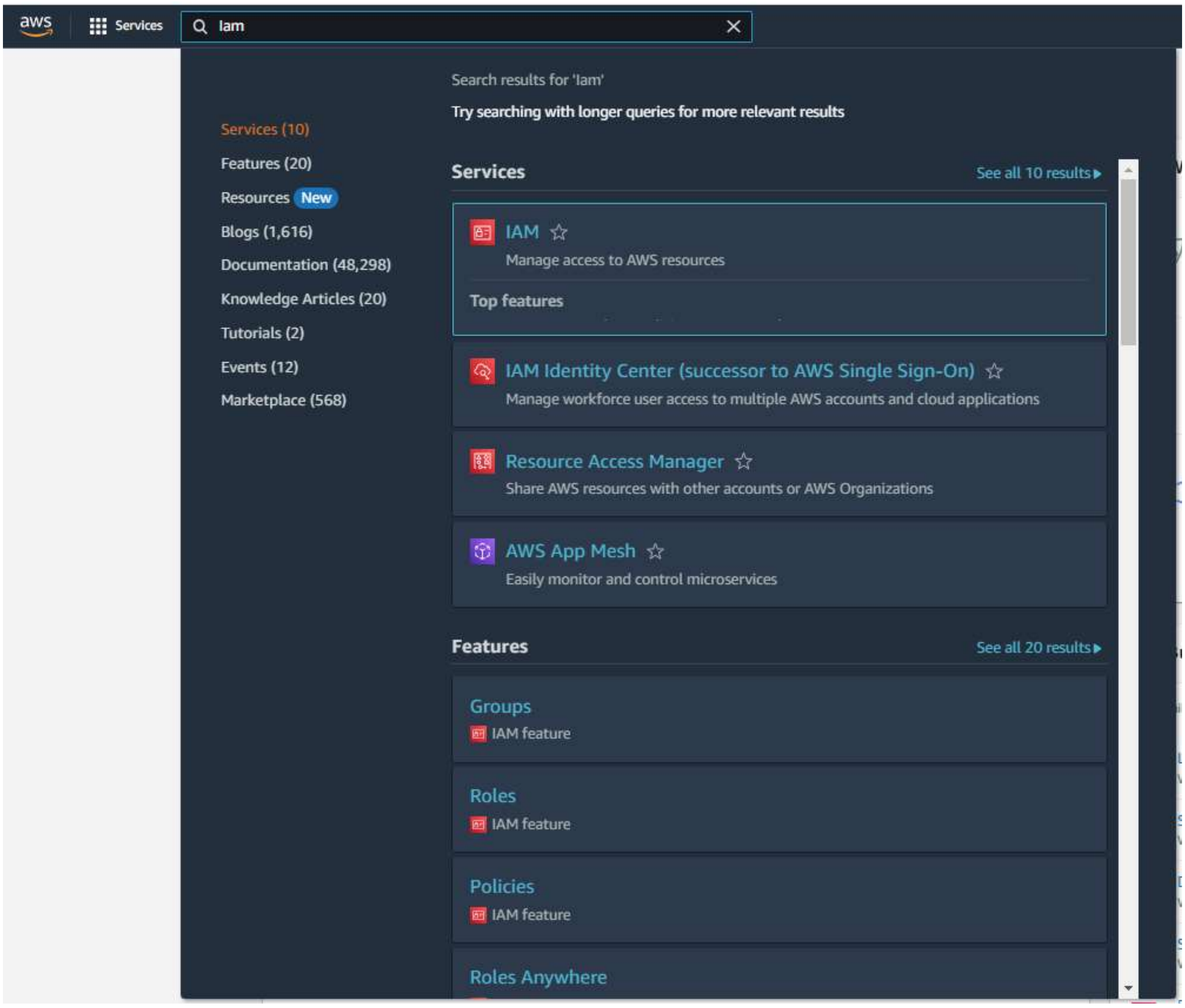


```
kulkarnivaibhav@ip-172-31-20-181:~$
kulkarnivaibhav@ip-172-31-20-181:~$
kulkarnivaibhav@ip-172-31-20-181:~$
kulkarnivaibhav@ip-172-31-20-181:~$ sudo su -
root@ip-172-31-20-181:~# wget https://releases.hashicorp.com/terraform/1.4.6/terraform_1.4.6_linux_amd64.zip
--2023-08-20 09:52:41--  https://releases.hashicorp.com/terraform/1.4.6/terraform_1.4.6_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 108.138.85.31, 108.138.85.53, 108.138.85.30, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|108.138.85.31|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20779821 (20M) [application/zip]
Saving to: ‘terraform_1.4.6_linux_amd64.zip’

terraform_1.4.6_linux_amd64.zip                      100%[=====================================================

2023-08-20 09:52:41 (424 MB/s) - ‘terraform_1.4.6_linux_amd64.zip’ saved [20779821/20779821]

root@ip-172-31-20-181:~# unzip terraform_1.4.6_linux_amd64.zip
Archive:  terraform_1.4.6_linux_amd64.zip
  inflating: terraform
root@ip-172-31-20-181:~# mv terraform /usr/local/bin
root@ip-172-31-20-181:~# terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
```

Second step is to setup AWS account : For that we will have to go to AWS or using AWS LAB provided by Simplilearn

To Set up AWS User and its security credentials

Create an IAM user and create accesskey and secret key >>   In search box give IAM --> select IAM service



Search results for 'Iam'

Try searching with longer queries for more relevant results

Services See all 10 results

IAM ☆
Manage access to AWS resources

Top features

IAM Identity Center (successor to AWS Single Sign-On) ☆
Manage workforce user access to multiple AWS accounts and cloud applications

Resource Access Manager ☆
Share AWS resources with other accounts or AWS Organizations

AWS App Mesh ☆
Easily monitor and control microservices

Features See all 20 results

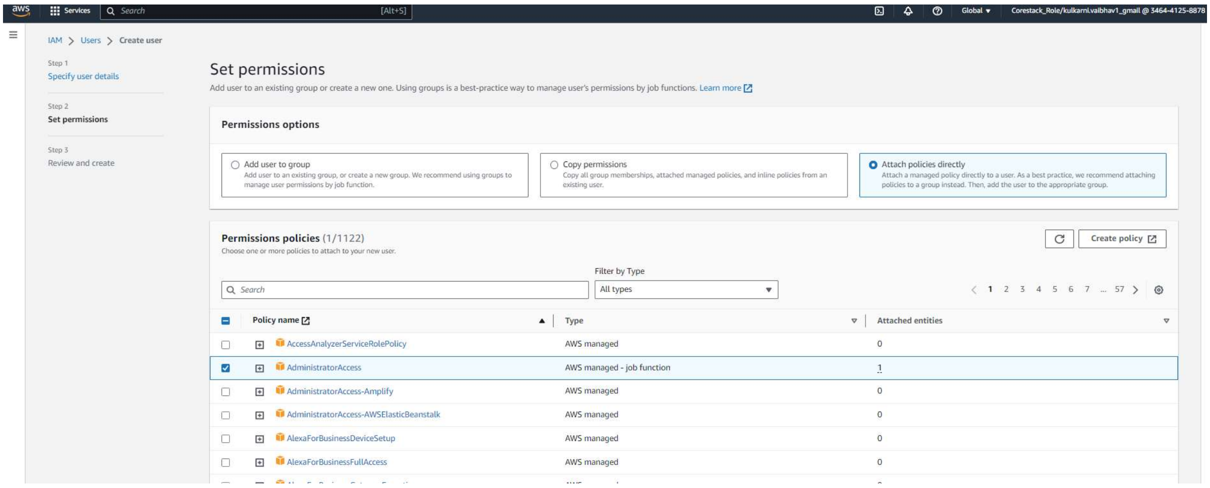
Groups
IAM feature

Roles
IAM feature

Policies
IAM feature

Roles Anywhere

On left side click on USers and then click on ADD users >>> Give user name = terraform --> press next >>>> Select Add exsiiting permission --> select Administrator accesss--> press next --> click on Create User



Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Permissions options

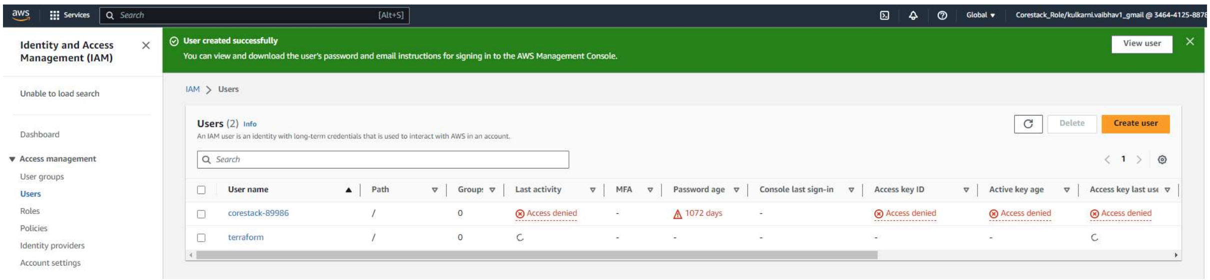
○ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

● Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1122)
Choose one or more policies to attach to your new user.

	Policy name	Type	Attached entities
□	AccessAnalyzerServiceRolePolicy	AWS managed	0
☑	AdministratorAccess	AWS managed - job function	1
□	AdministratorAccess-Amplify	AWS managed	0
□	AdministratorAccess-AWSElasticBeanstalk	AWS managed	0
□	AlexaForBusinessDeviceSetup	AWS managed	0
□	AlexaForBusinessFullAccess	AWS managed	0



User created successfully
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

IAM > Users

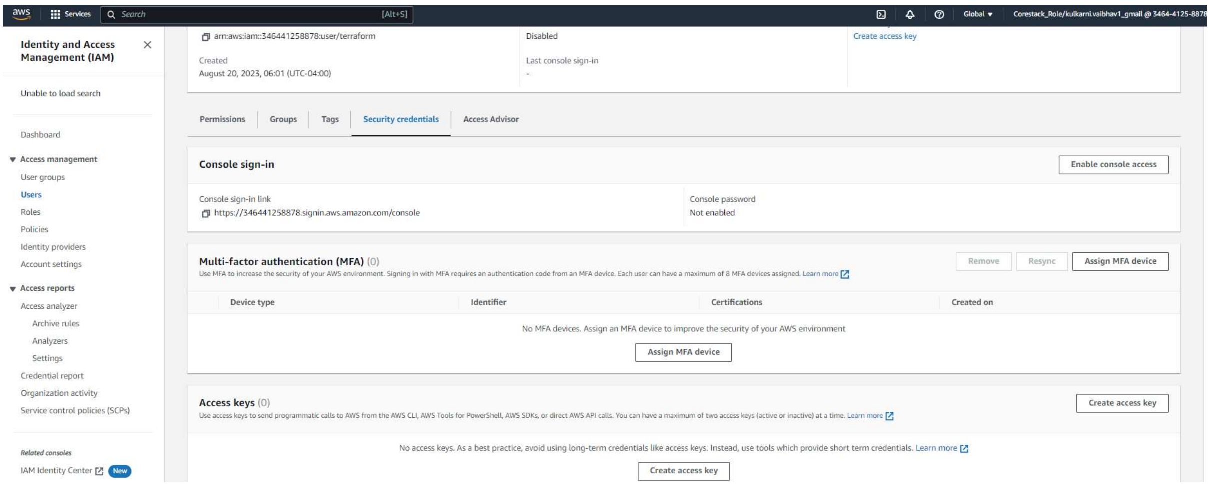
Users (2)
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

□	User name ▲	Path	Group:	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access key last use
□	corestack-89986	/	0	Access denied	-	1072 days	-	Access denied	Access denied	Access denied
□	terraform	/	0		-	-	-	-	-	

Add credentials to the USER in AWS

==

Click on the user name terraform --> click on Security credentials --> scroll down to Access Key --> click on Create Access key >>>> Select Command Line Interface (CLI) --> scroll down and check the box for I understand the above recommendation and want to proceed to create an access key. >>> Press next and click on create access key >>> Copy the access key and secret key



arn:aws:iam::346441258878:user/terraform
Disabled
Create access key

Created
August 20, 2023, 06:01 (UTC-04:00)

Last console sign-in
-

Console sign-in

Console sign-in link
https://346441258878.signin.aws.amazon.com/console

Console password
Not enabled

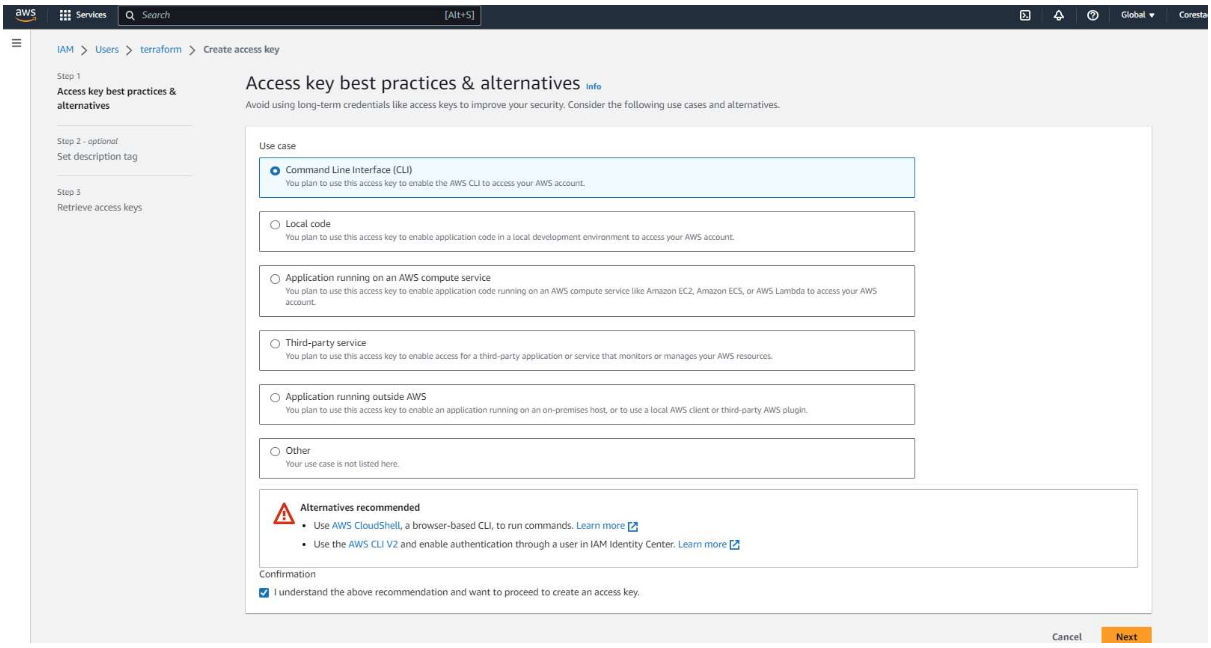
Multi-factor authentication (MFA) (0)
Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. Learn more

Device type	Identifier	Certifications	Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Access keys (0)
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more



IAM > Users > terraform > Create access key

Step 1
Access key best practices & alternatives

Step 2 - *optional*
Set description tag

Step 3
Retrieve access keys

Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

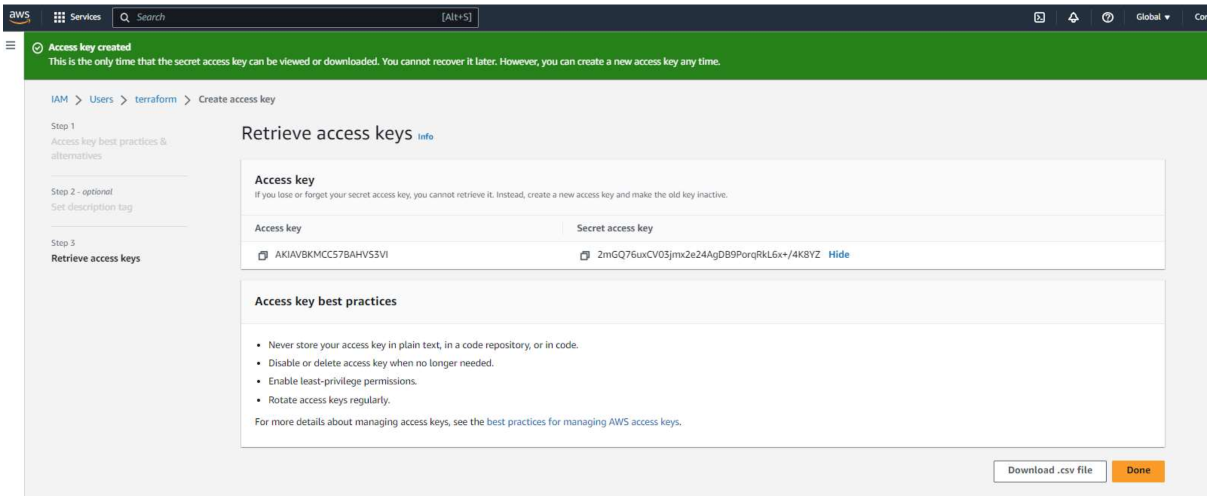
○ Application running outside AWS
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

○ Other
Your use case is not listed here.

Alternatives recommended
- Use AWS CloudShell, a browser-based CLI, to run commands. Learn more
- Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center. Learn more

Confirmation
☑ I understand the above recommendation and want to proceed to create an access key.

Access key	Secret access key
AKIAVBKMCC57BAHVS3VI	2mGQ76uxCV03jmx2e24AgDB9PorqRkL6x+/4K8YZ



Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

IAM > Users > terraform > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys

Retrieve access keys Info

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAVBKMCC57BAHVS3VI	2mGQ76uxCV03jmx2e24AgDB9PorqRkL6x+/4K8YZ Hide

Access key best practices

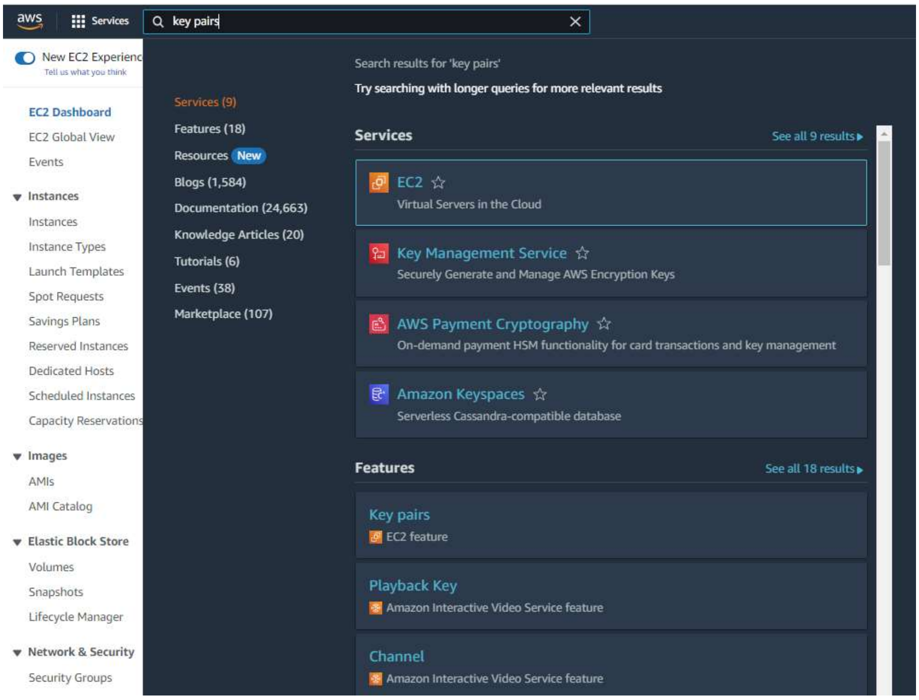
- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

3. Create KeyPiars in AWS

===

In search box --> give key pairs --> click on keypairs under features >> Click on create key pair --> give name as project1 >>> Key pair type ---> ED25519 >>> Private key file format --> .pem >> Click on Create key pair



Search results for 'key pairs'

Try searching with longer queries for more relevant results

Services — See all 9 results ▸

- EC2 ☆ — Virtual Servers in the Cloud
- Key Management Service ☆ — Securely Generate and Manage AWS Encryption Keys
- AWS Payment Cryptography ☆ — On-demand payment HSM functionality for card transactions and key management
- Amazon Keyspaces ☆ — Serverless Cassandra-compatible database

Features — See all 18 results ▸

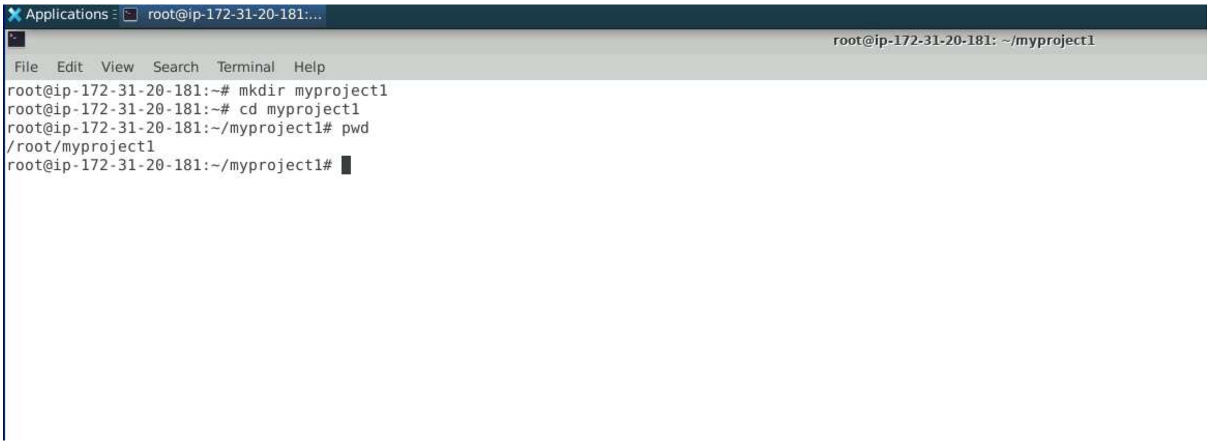
- Key pairs — EC2 feature
- Playback Key — Amazon Interactive Video Service feature
- Channel — Amazon Interactive Video Service feature

Expected Deliverables:

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

On simplilearn lab follow the steps

```
# mkdir myproject1
# cd myproject1
```


Applications
root@ip-172-31-20-181:...
root@ip-172-31-20-181: ~/myproject1
File Edit View Search Terminal Help

```
root@ip-172-31-20-181:~# mkdir myproject1
root@ip-172-31-20-181:~# cd myproject1
root@ip-172-31-20-181:~/myproject1# pwd
/root/myproject1
root@ip-172-31-20-181:~/myproject1# 
```

Step now is to create terraform file which will have all detailed activity

```
# vim terraformproject.tf
```

```
provider "aws" {
 region     = "us-east-1"
 access_key = "AKIAVBKMCC57BAHVS3VI"
 secret_key = "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"

}
```

save the file

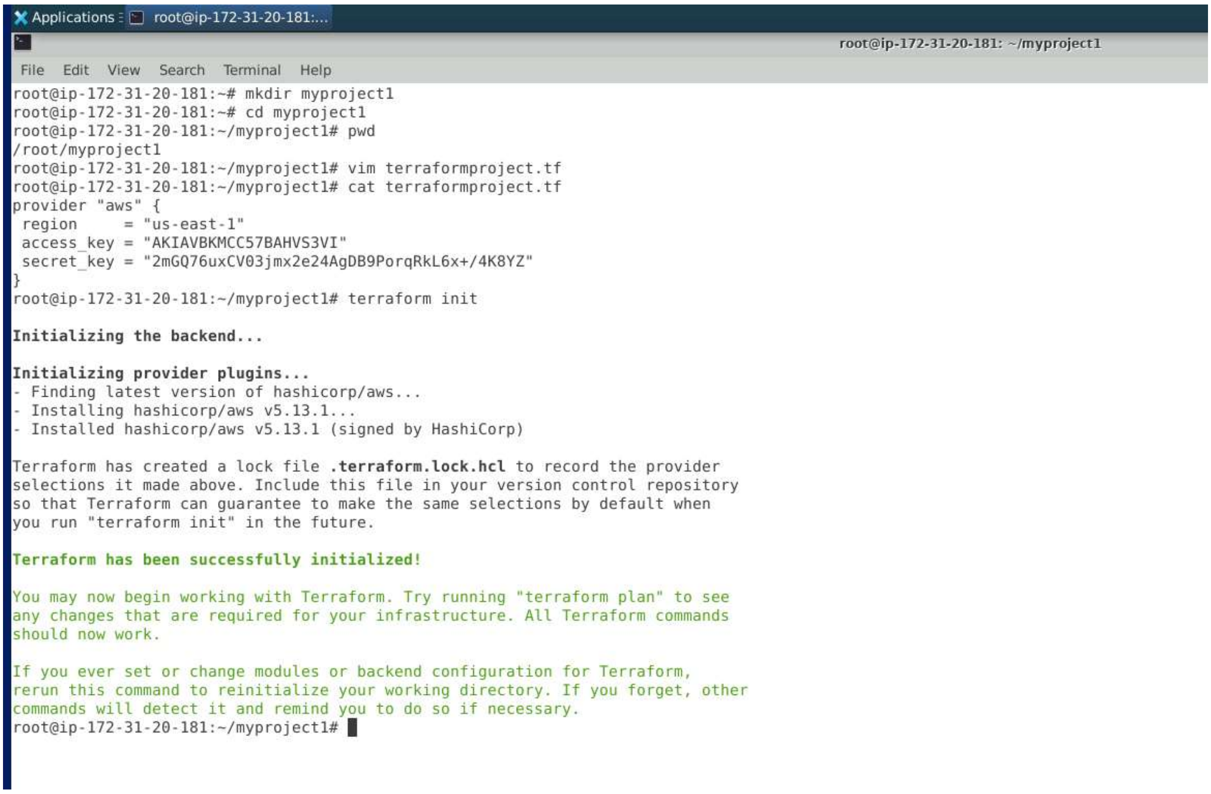
==

```
root@ip-172-31-20-181:~/myproject1# cat terraformproject.tf
provider "aws" {
 region     = "us-east-1"
 access_key = "AKIAVBKMCC57BAHVS3VI"
 secret_key = "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"
}
root@ip-172-31-20-181:~/myproject1#
```

==

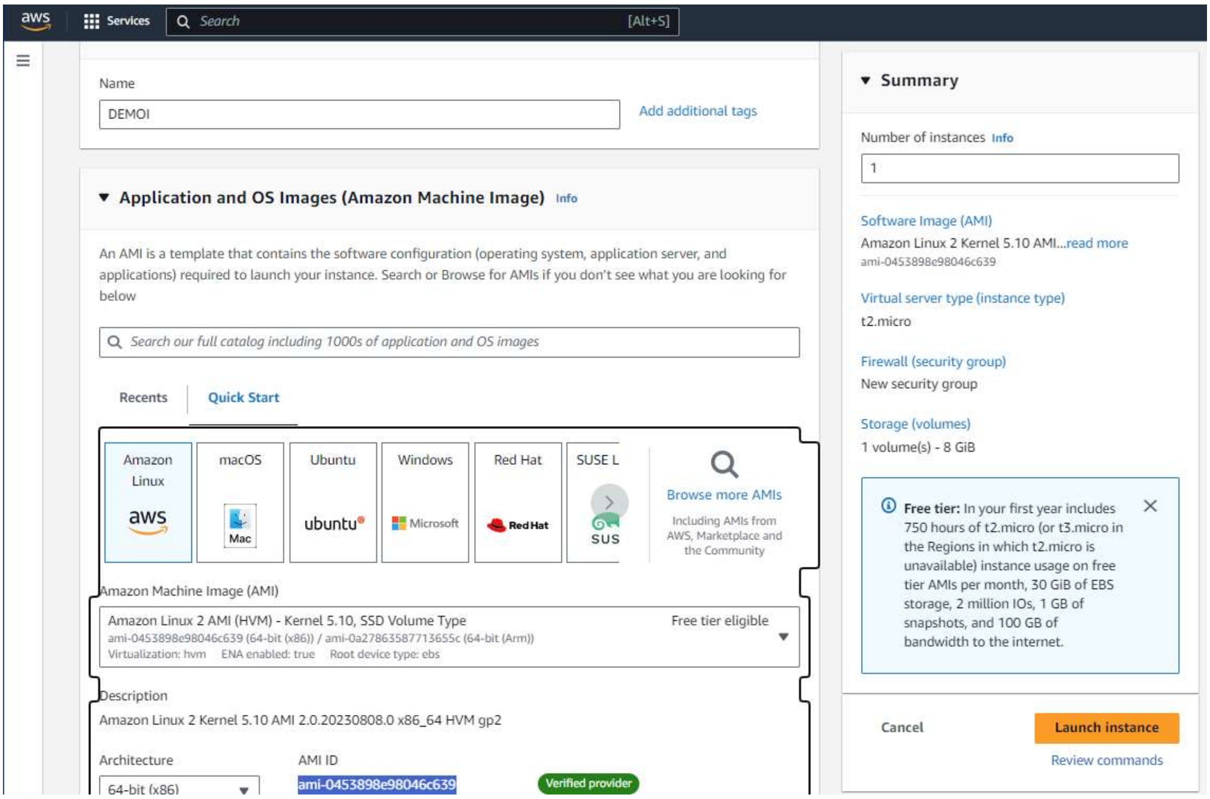
Execute the command.

# terraform init


Applications
root@ip-172-31-20-181:...
root@ip-172-31-20-181: ~/myproject1
File Edit View Search Terminal Help

```
root@ip-172-31-20-181:~# mkdir myproject1
root@ip-172-31-20-181:~# cd myproject1
root@ip-172-31-20-181:~/myproject1# pwd
/root/myproject1
root@ip-172-31-20-181:~/myproject1# vim terraformproject.tf
root@ip-172-31-20-181:~/myproject1# cat terraformproject.tf
provider "aws" {
 region     = "us-east-1"
 access_key = "AKIAVBKMCC57BAHVS3VI"
 secret_key = "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"
}
root@ip-172-31-20-181:~/myproject1# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.1...
- Installed hashicorp/aws v5.13.1 (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.
root@ip-172-31-20-181:~/myproject1#
```

For ami id get the following



Name: DEMOI

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-0453898e98046c639 (64-bit (x86)) / ami-0a27863587713655c (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description: Amazon Linux 2 Kernel 5.10 AMI 2.0.20230808.0 x86_64 HVM gp2

Architecture: 64-bit (x86)

AMI ID: ami-0453898e98046c639

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2 Kernel 5.10 AMI...read more
ami-0453898e98046c639

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

5. Launch an EC2 instance using Terraform **terraformproject.tf**

```
==============================================================
resource "aws_security_group" "Project1SG" {
  name        = "Proj1"
  description = "Allow inbound SSH"

  ingress {
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
    ipv6_cidr_blocks = ["::/0"]
  }
ingress {
      description = "HTTP"
      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"]
    }
}


resource "aws_instance" "Project-web1" {
  ami           = "ami-0453898e98046c639"
  instance_type = "t2.micro"

  tags = {
    Name = "Project1-web1"
  }
   key_name = "project1"
   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 yum -y install gcc openssl-devel bzip2-devel libffi-devel zlib-devel
            sudo wget  https://www.python.org/ftp/python/3.11.4/Python-3.11.4.tgz
```

```
          sudo  tar xvf Python-3.11.4.tgz
       sudo  cd /home/ec2-user/Python-3.11.4/
          sudo /home/ec2-user/Python-3.11.4/configure --enable-optimizations
          sudo /home/ec2-user/Python-3.11.4/make altinstall
  EOF
}

resource "aws_network_interface_sg_attachment" "sg_attachment1"
{
security_group_id = aws_security_group.Project1SG.id
network_interface_id = aws_instance.Project-web1.primary_network_interface_id
}
```

```
=================================================================
root@ip-172-31-20-181:~/myproject1# terraform plan


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

Execute command

```
# terraform apply
```

Enter yes

After successful process of terraform

```
======================================================
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_security_group.Project1SG: Creating...
aws_instance.Project-web1: Creating...
```

aws_security_group.Project1SG: Creation complete after 2s [id=sg-065b9af60e62705c1]

aws_instance.Project-web1: Still creating... [10s elapsed]

aws_instance.Project-web1: Still creating... [20s elapsed]

aws_instance.Project-web1: Still creating... [30s elapsed]

aws_instance.Project-web1: Still creating... [40s elapsed]

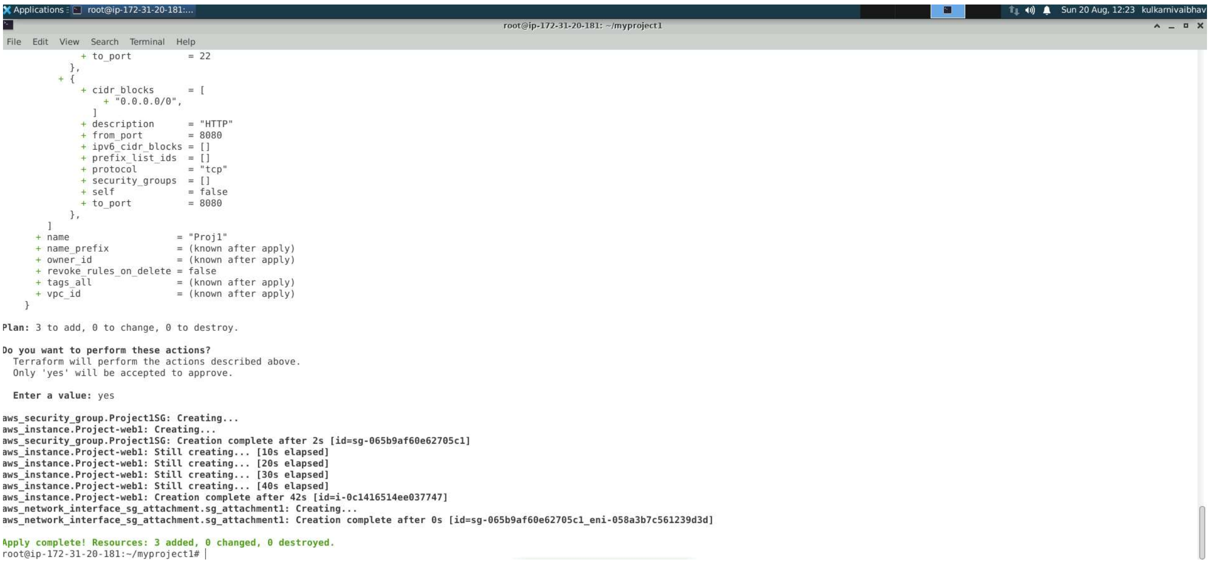
aws_instance.Project-web1: Creation complete after 42s [id=i-0c1416514ee037747]

aws_network_interface_sg_attachment.sg_attachment1: Creating...

aws_network_interface_sg_attachment.sg_attachment1: Creation complete after 0s [id=sg-065b9af60e62705c1_eni-058a3b7c561239d3d]

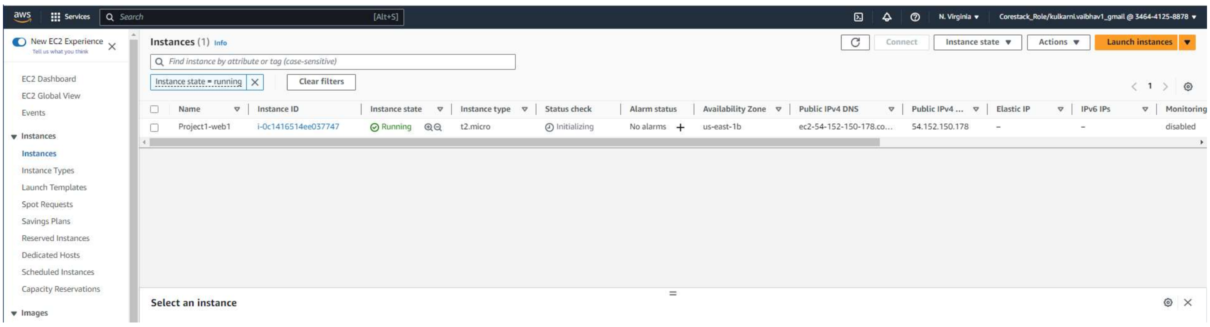
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

===



root@ip-172-31-20-181:...
Sun 20 Aug, 12:23 kulkarnivaibhav
root@ip-172-31-20-181: ~/myproject1
File Edit View Search Terminal Help

```
          + to_port          = 22
        },
      + {
          + cidr_blocks      = [
              + "0.0.0.0/0",
            ]
          + description      = "HTTP"
          + from_port        = 8080
          + ipv6_cidr_blocks = []
          + prefix_list_ids  = []
          + protocol         = "tcp"
          + security_groups  = []
          + self             = false
          + to_port          = 8080
        },
    ]
  + name                   = "Proj1"
  + name_prefix            = (known after apply)
  + owner_id               = (known after apply)
  + revoke_rules_on_delete = false
  + tags_all               = (known after apply)
  + vpc_id                 = (known after 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_security_group.Project1SG: Creating...
aws_instance.Project-web1: Creating...
aws_security_group.Project1SG: Creation complete after 2s [id=sg-065b9af60e62705c1]
aws_instance.Project-web1: Still creating... [10s elapsed]
aws_instance.Project-web1: Still creating... [20s elapsed]
aws_instance.Project-web1: Still creating... [30s elapsed]
aws_instance.Project-web1: Still creating... [40s elapsed]
aws_instance.Project-web1: Creation complete after 42s [id=i-0c1416514ee037747]
aws_network_interface_sg_attachment.sg_attachment1: Creating...
aws_network_interface_sg_attachment.sg_attachment1: Creation complete after 0s [id=sg-065b9af60e62705c1_eni-058a3b7c561239d3d]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
root@ip-172-31-20-181:~/myproject1#
```

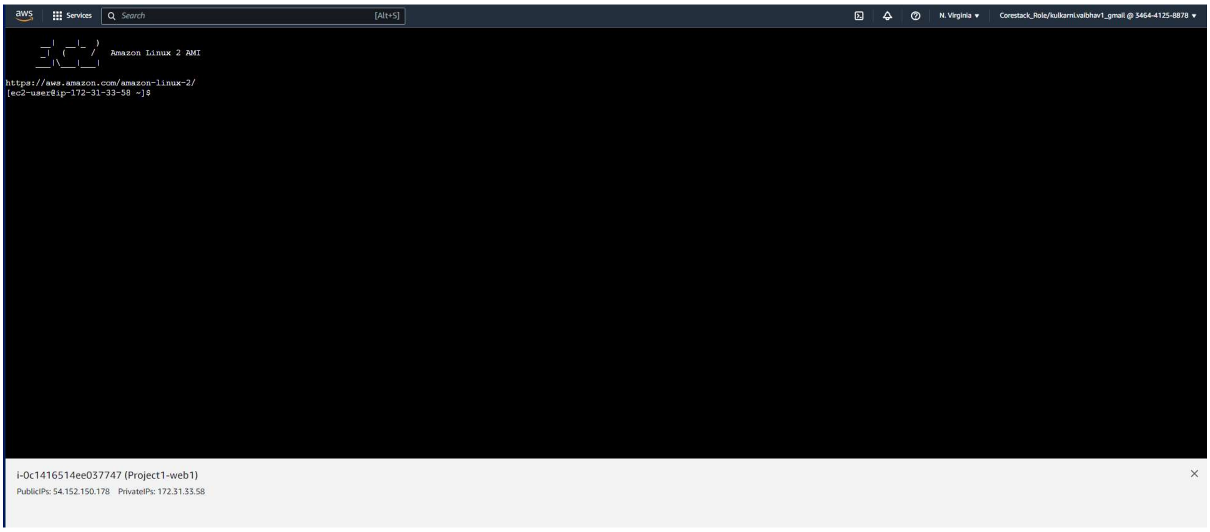


Instances (1) Info

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring
Project1-web1	i-0c1416514ee037747	Running	t2.micro	Initializing	No alarms	us-east-1b	ec2-54-152-150-178.co...	54.152.150.178	–	–	disabled

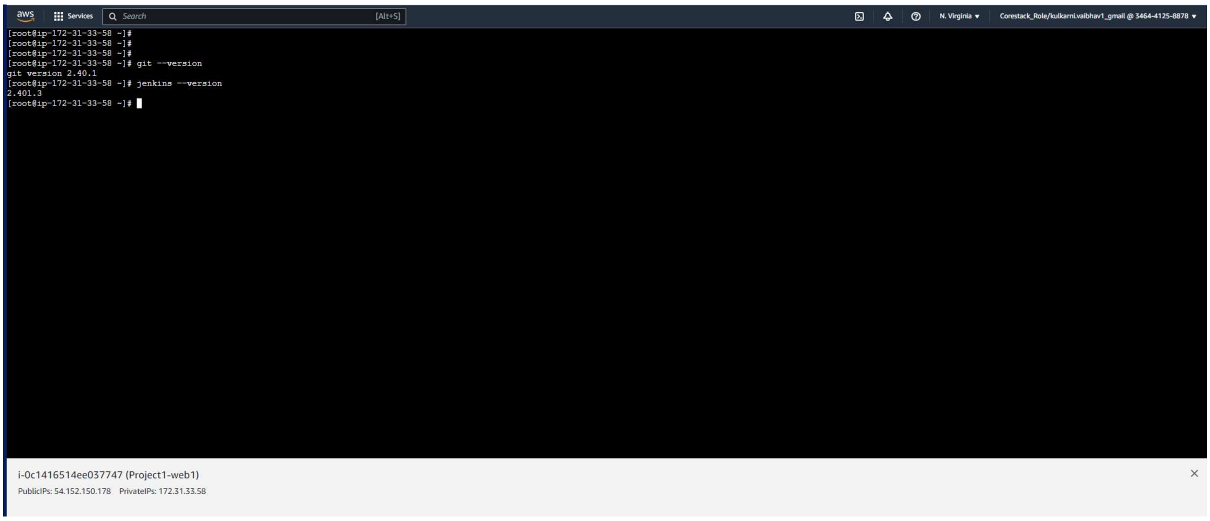
Select an instance



```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-33-58 ~]$
```

i-0c1416514ee037747 (Project1-web1)

PublicIPs: 54.152.150.178 PrivateIPs: 172.31.33.58



```
[root@ip-172-31-33-58 ~]#
[root@ip-172-31-33-58 ~]#
[root@ip-172-31-33-58 ~]#
[root@ip-172-31-33-58 ~]# git --version
git version 2.40.1
[root@ip-172-31-33-58 ~]# jenkins --version
2.401.3
[root@ip-172-31-33-58 ~]#
```

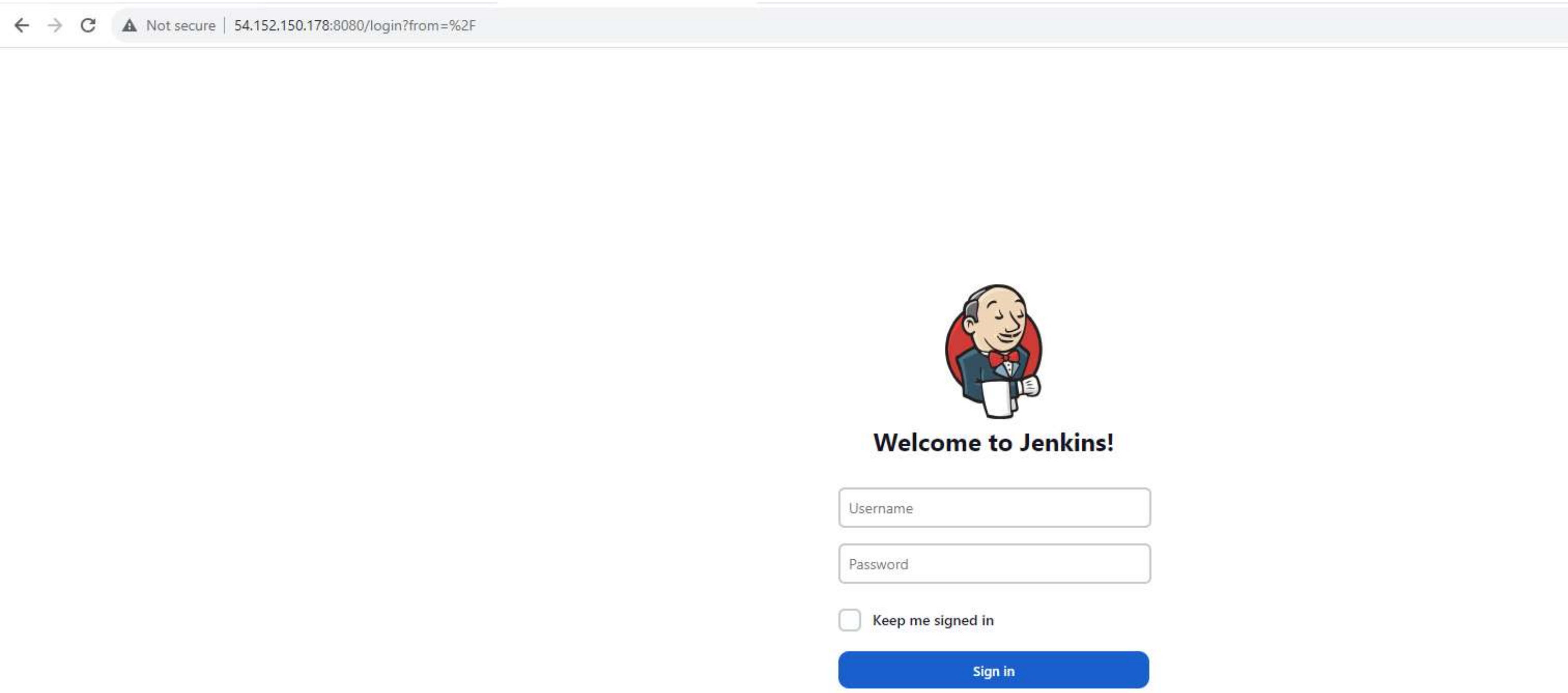
i-0c1416514ee037747 (Project1-web1)

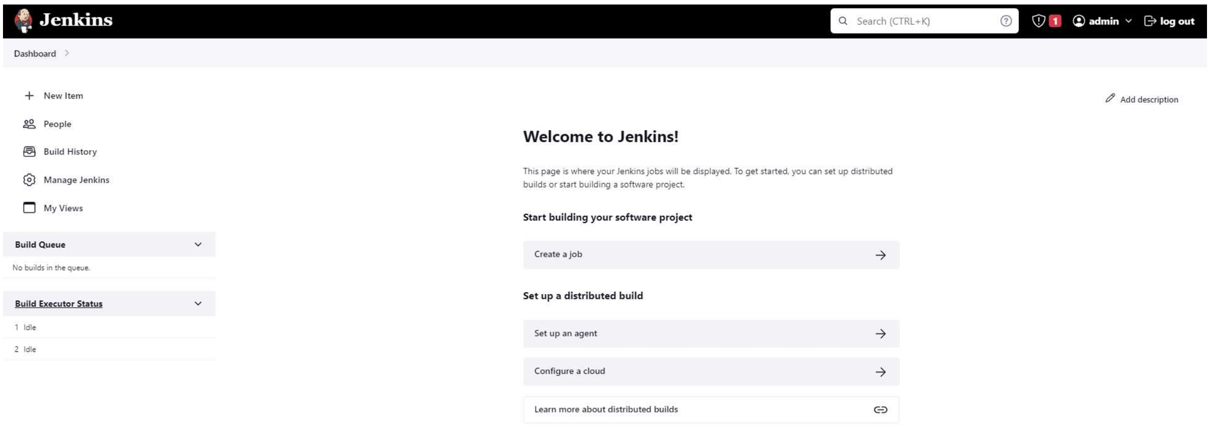
PublicIPs: 54.152.150.178 PrivateIPs: 172.31.33.58

```
[root@ip-172-31-33-58 ~]#

[root@ip-172-31-33-58 ~]# git --version

git version 2.40.1

[root@ip-172-31-33-58 ~]# jenkins --version

2.401.3

[root@ip-172-31-33-58 ~]#

[root@ip-172-31-33-58 /]# /bin/python --version

Python 2.7.18

[root@ip-172-31-33-58 /]# nano hello.py

[root@ip-172-31-33-58 /]# python3 hello.py

Hello, World!
```

```
[root@ip-172-31-33-58 /]#
```

Jenkins login URL : http://54.152.150.178:8080/





No builds in the queue.

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build