PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)

.

Academic Year: 2025-26
Semester: V
Class / Branch/ Div: TEIT C
Subject: ADL Lab

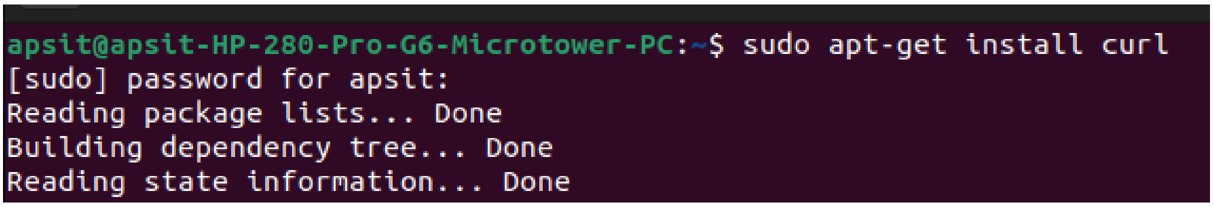
Name of Student:Shreya Sonawale
Student ID:23104134
Roll No. 43
Date of Submission:12/09/25

Experiment No: 6

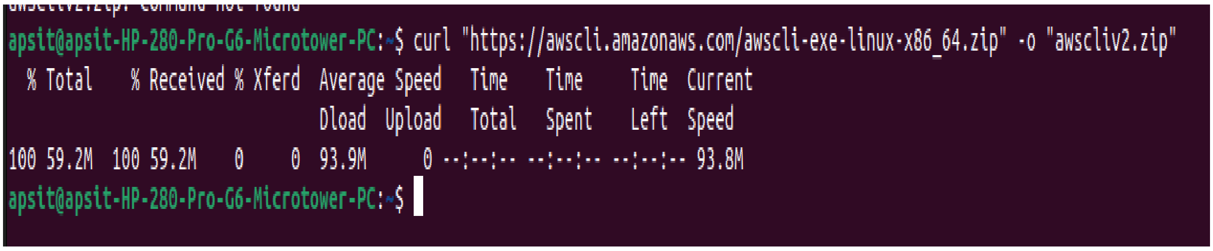
Aim: To Build, change, and destroy AWS infrastructure Using Terraform.

Software Used: Ubuntu **ami-02d26659fd82cf299**

Code & Output:

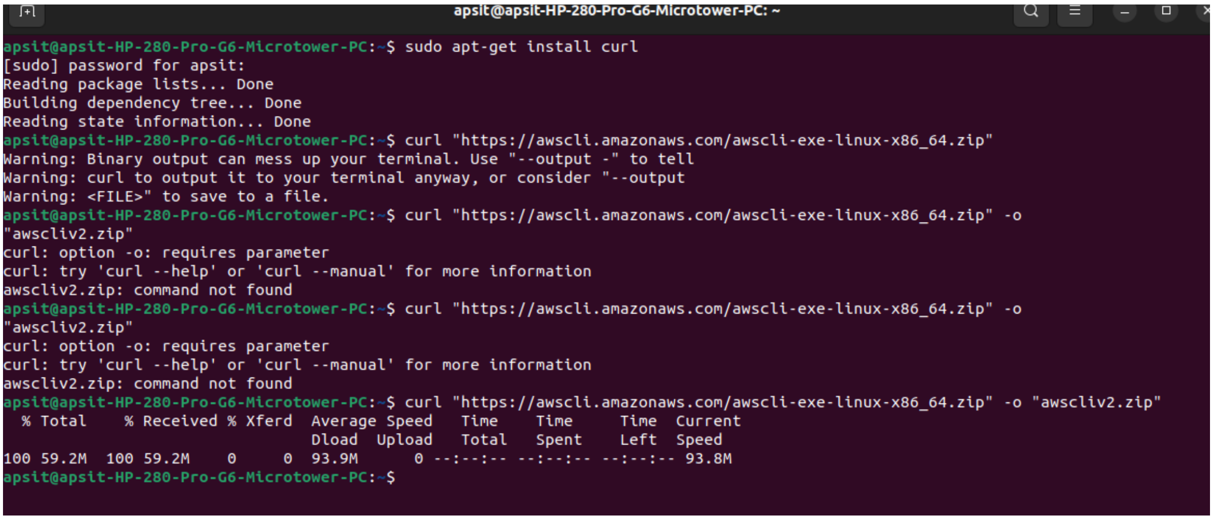


```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ sudo apt-get install curl
[sudo] password for apsit:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

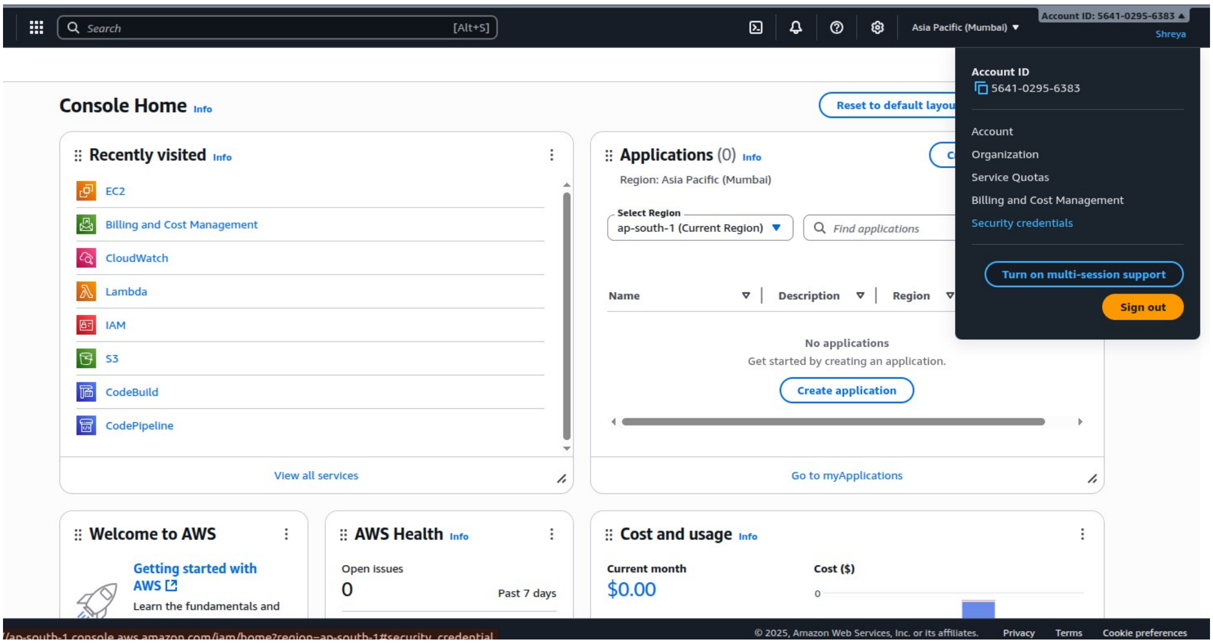


```
awscliv2.zip: command not found
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 59.2M  100 59.2M    0     0  93.9M      0 --:--:-- --:--:-- --:--:-- 93.8M
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$
```

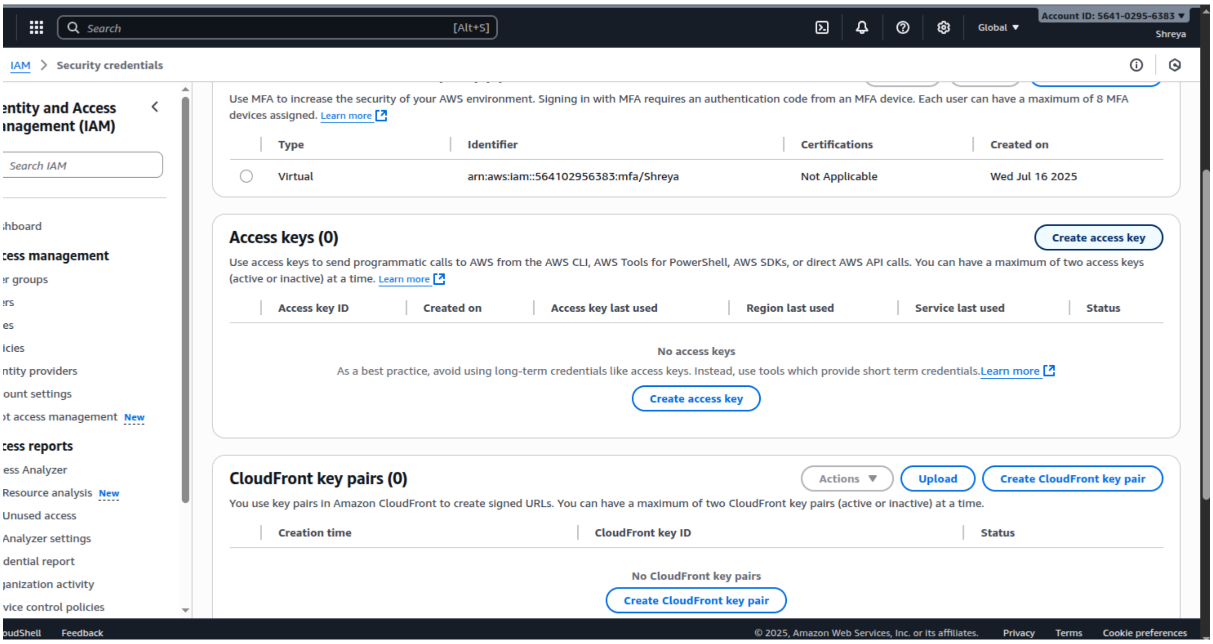
.



```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ sudo apt-get install curl
[sudo] password for apsit:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip"
Warning: Binary output can mess up your terminal. Use "--output -" to tell
Warning: curl to output it to your terminal anyway, or consider "--output
Warning: <FILE>" to save to a file.
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
curl: option -o: requires parameter
curl: try 'curl --help' or 'curl --manual' for more information
awscliv2.zip: command not found
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
curl: option -o: requires parameter
curl: try 'curl --help' or 'curl --manual' for more information
awscliv2.zip: command not found
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 59.2M  100 59.2M    0     0  93.9M      0 --:--:-- --:--:-- --:--:-- 93.8M
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$
```



.



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

	Type	Identifier	Certifications	Created on
○	Virtual	arn:aws:iam::564102956383:mfa/Shreya	Not Applicable	Wed Jul 16 2025

Access keys (0) Create access key

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

Access key ID	Created on	Access key last used	Region last used	Service last used	Status

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

Create access key

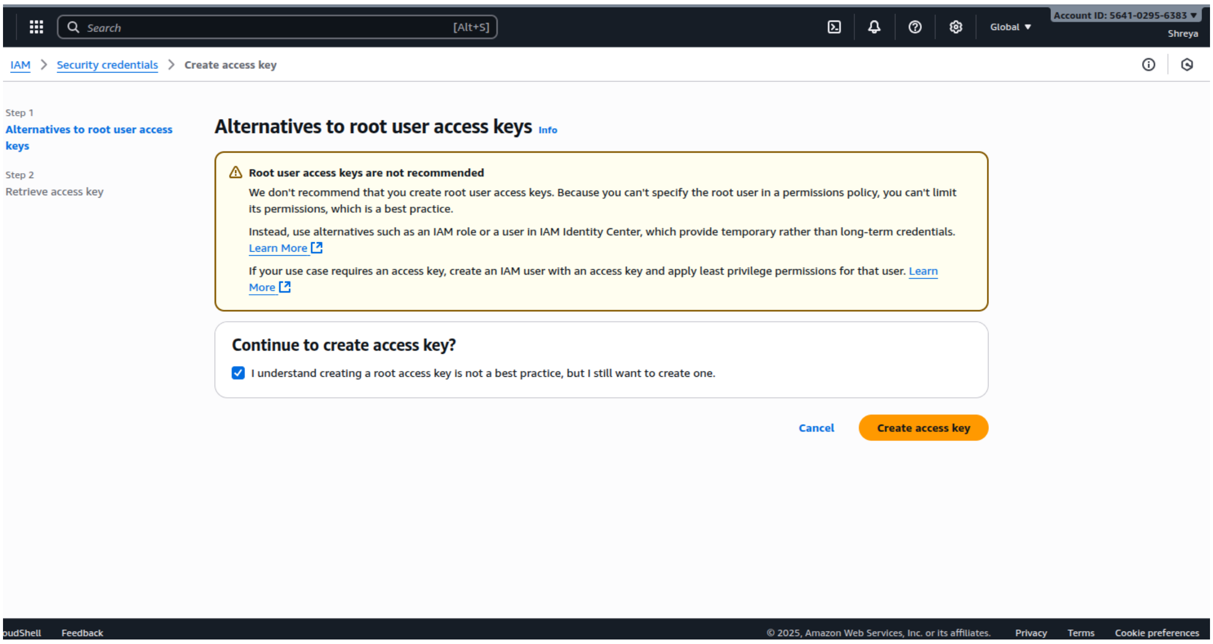
CloudFront key pairs (0) Actions ▾ Upload Create CloudFront key pair

You use key pairs in Amazon CloudFront to create signed URLs. You can have a maximum of two CloudFront key pairs (active or inactive) at a time.

Creation time	CloudFront key ID	Status

No CloudFront key pairs

Create CloudFront key pair


Search [Alt+S]
Global ▾
Account ID: 5641-0295-6383 ▾
Shreya
IAM > Security credentials > Create access key
oudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Alternatives to root user access keys Info

⚠ **Root user access keys are not recommended**

We don't recommend that you create root user access keys. Because you can't specify the root user in a permissions policy, you can't limit its permissions, which is a best practice.

Instead, use alternatives such as an IAM role or a user in IAM Identity Center, which provide temporary rather than long-term credentials. Learn More

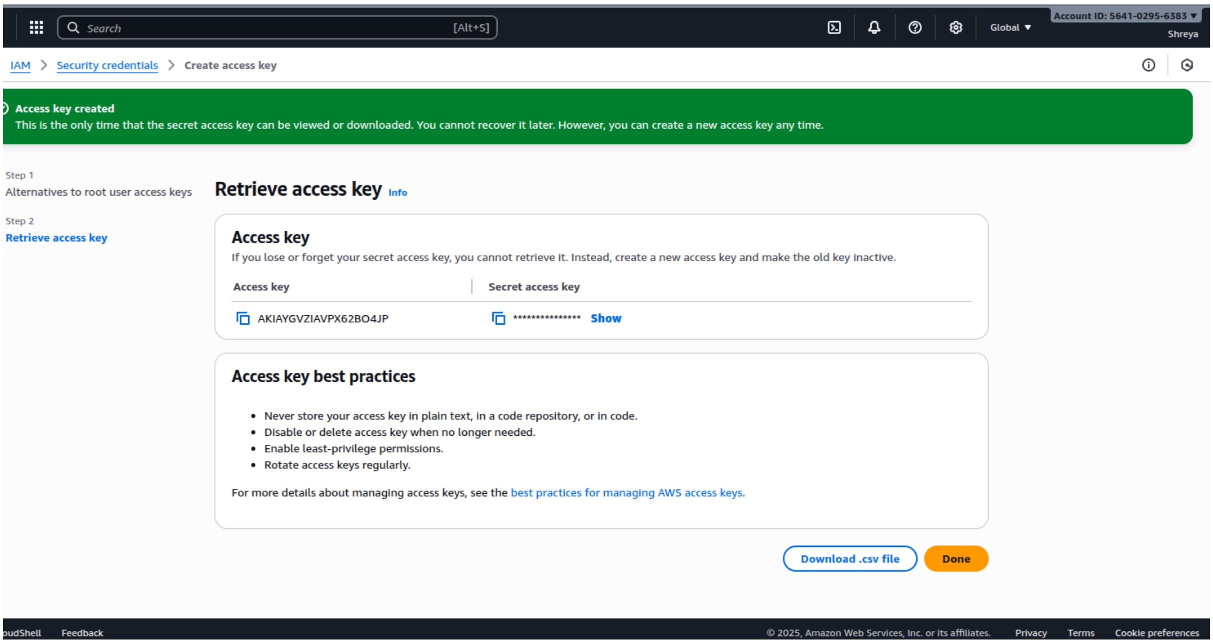
If your use case requires an access key, create an IAM user with an access key and apply least privilege permissions for that user. Learn More

Continue to create access key?

☑ I understand creating a root access key is not a best practice, but I still want to create one.

Cancel Create access key

.



Search [Alt+S]
Global ▾
Account ID: 5641-0295-6383 ▾
Shreya
IAM > Security credentials > Create access key

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.

Step 1
Alternatives to root user access keys
Step 2
Retrieve access key

Retrieve access key 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.

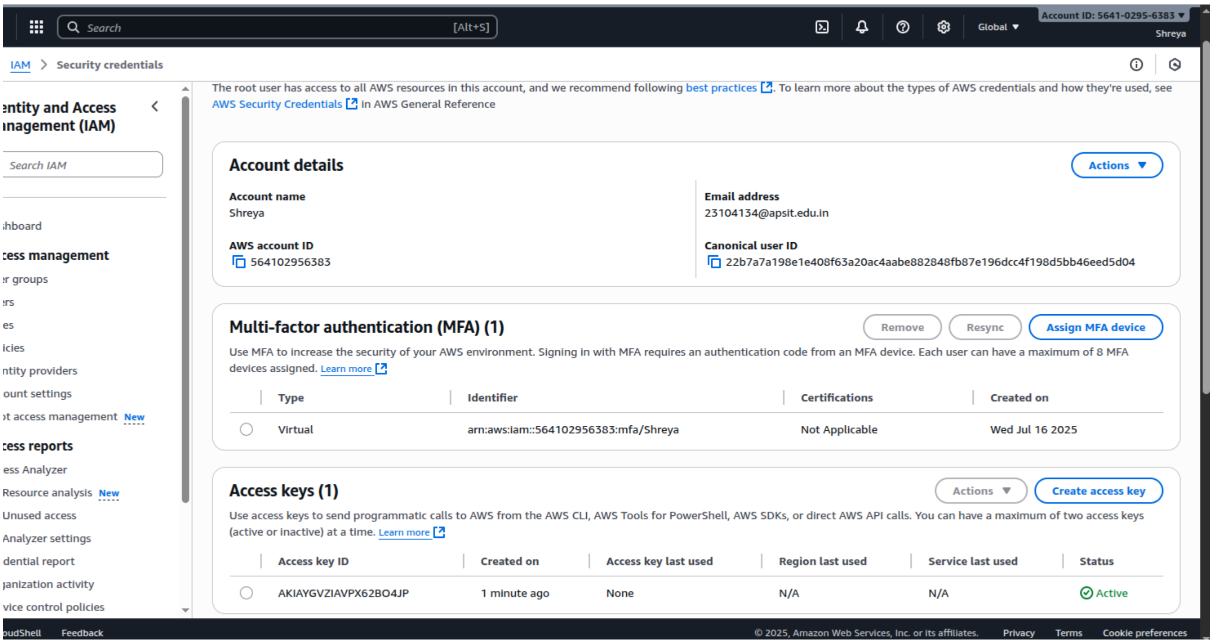
Access key	Secret access key
AKIAYGVZIAVPX62BO4JP	*************** Show

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.

Download .csv file
Done
oudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



The root user has access to all AWS resources in this account, and we recommend following best practices. To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference

Account details

Account name
Shreya

Email address
23104134@apsit.edu.in

AWS account ID
564102956383

Canonical user ID
22b7a7a198e1e408f63a20ac4aabe882848fb87e196dcc4f198d5bb46eed5d04

Multi-factor authentication (MFA) (1)

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

	Type	Identifier	Certifications	Created on
○	Virtual	arn:aws:iam::564102956383:mfa/Shreya	Not Applicable	Wed Jul 16 2025

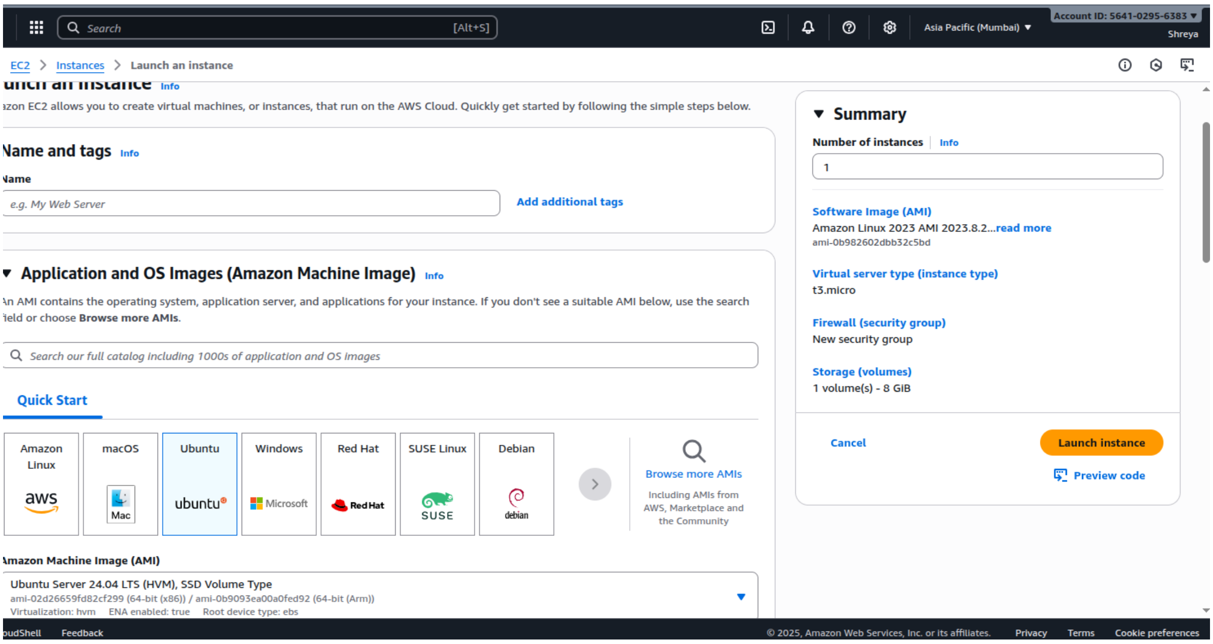
Access keys (1)

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

	Access key ID	Created on	Access key last used	Region last used	Service last used	Status
○	AKIAYGVZIAVPX62BO4JP	1 minute ago	None	N/A	N/A	✓ Active

oudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

.



EC2 > Instances > Launch an instance

unch an instance Info

azon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

e.g. My Web Server

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

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

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-02d26659fd82cf299 (64-bit (x86)) / ami-0b9093ea00a0fed92 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

▼ Summary

Number of instances | Info

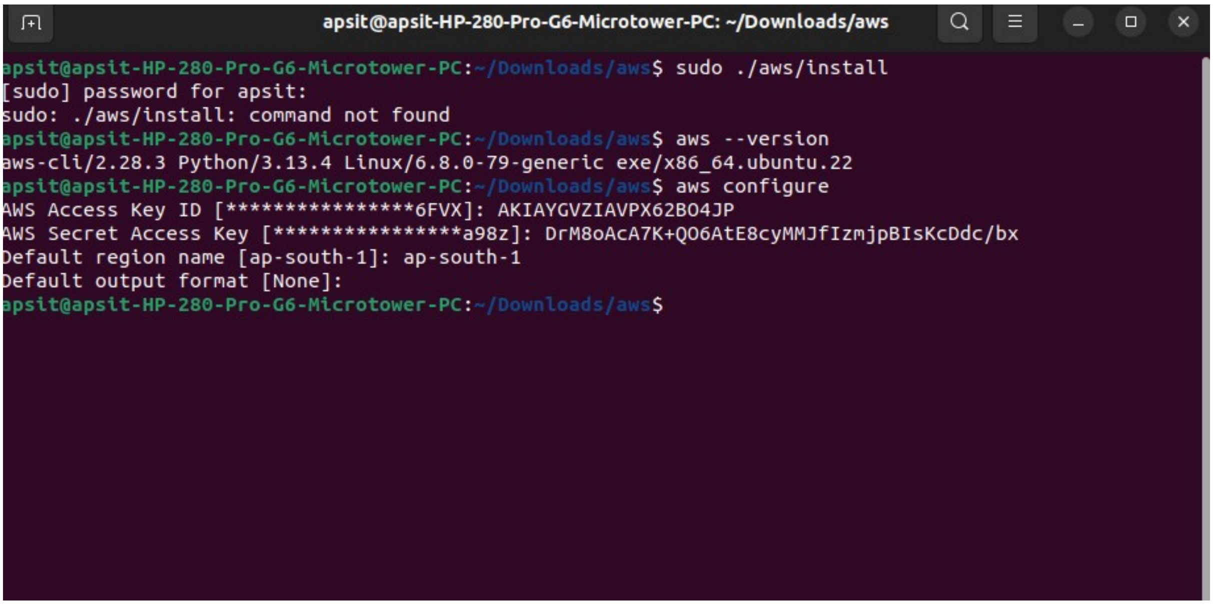
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.8.2...read more
ami-0b982602dbb32c5bd

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

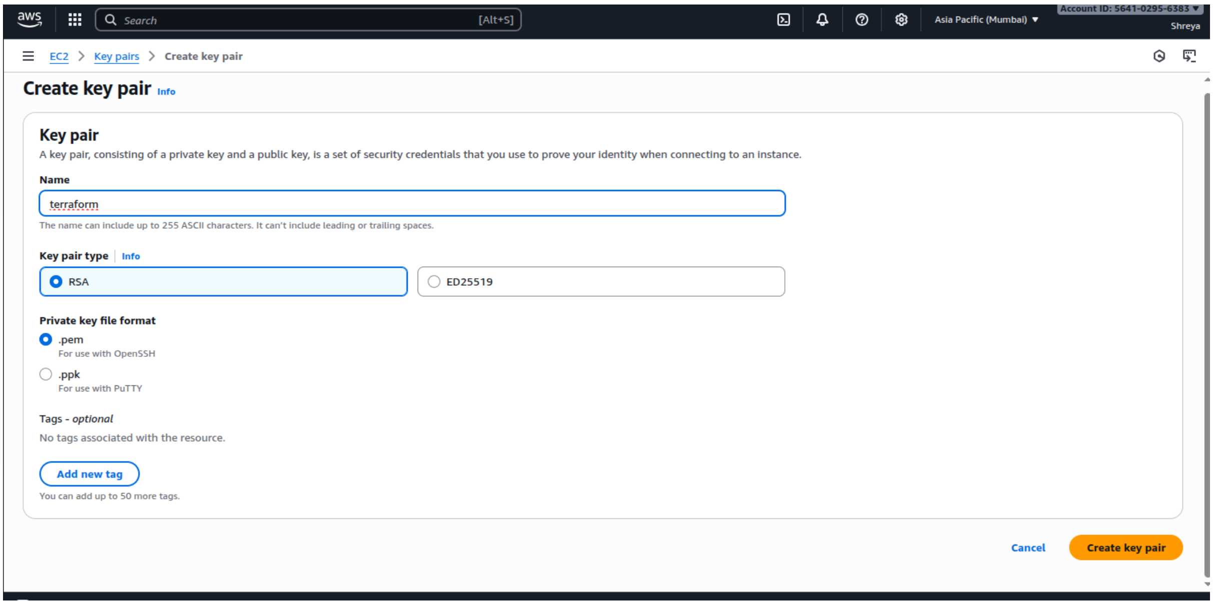
Storage (volumes)
1 volume(s) - 8 GiB



apsit@apsit-HP-280-Pro-G6-Microtower-PC: ~/Downloads/aws

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads/aws$ sudo ./aws/install
[sudo] password for apsit:
sudo: ./aws/install: command not found
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads/aws$ aws --version
aws-cli/2.28.3 Python/3.13.4 Linux/6.8.0-79-generic exe/x86_64.ubuntu.22
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads/aws$ aws configure
AWS Access Key ID [****************6FVX]: AKIAYGVZIAVPX62BO4JP
AWS Secret Access Key [****************a98z]: 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
Default region name [ap-south-1]: ap-south-1
Default output format [None]:
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Downloads/aws$
```

.



aws
Search [Alt+S]
Asia Pacific (Mumbai) ▾
Account ID: 5641-0295-6383 ▾
Shreya
EC2 > Key pairs > Create key pair

Create key pair Info

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

terraform

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type Info

● RSA ○ ED25519

Private key file format

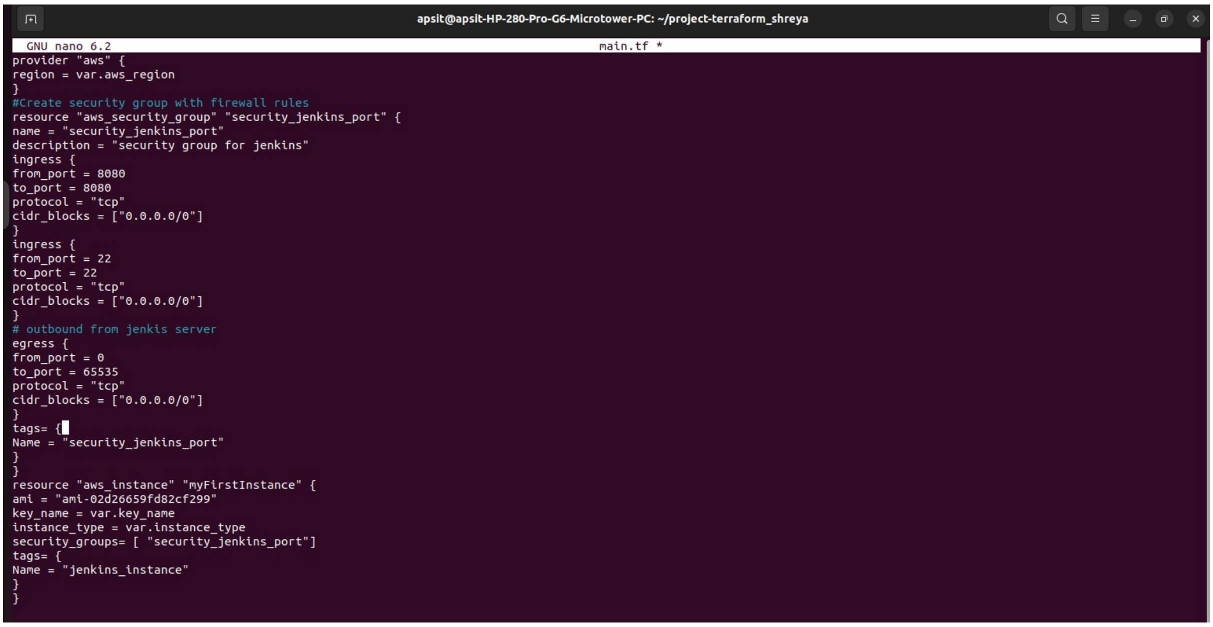
● .pem
For use with OpenSSH

○ .ppk
For use with PuTTY

Tags - *optional*

No tags associated with the resource.

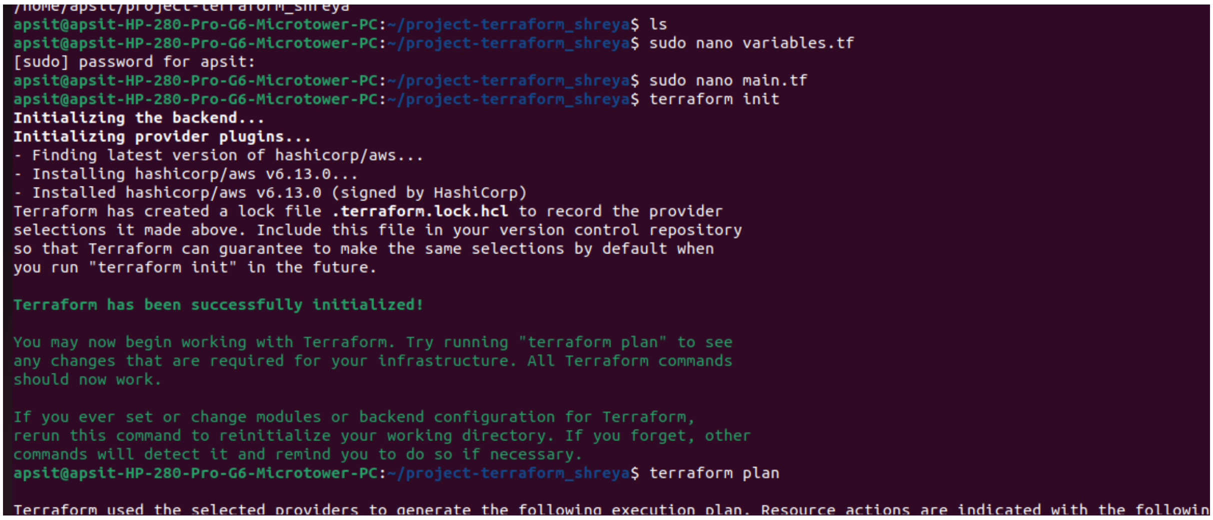
You can add up to 50 more tags.



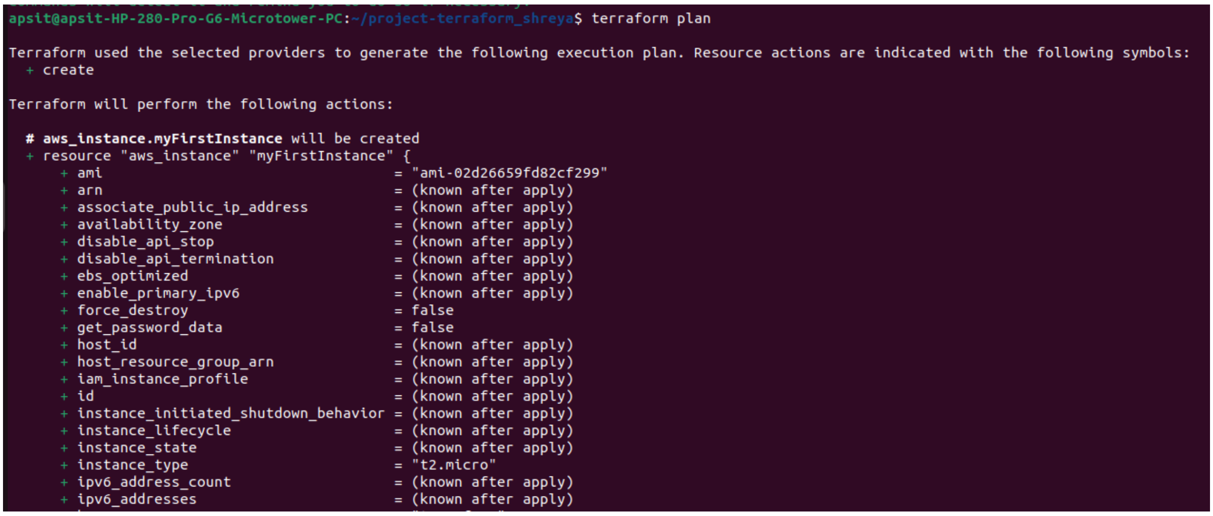
apsit@apsit-HP-280-Pro-G6-Microtower-PC: ~/project-terraform_shreya

```
  GNU nano 6.2                                          main.tf *
provider "aws" {
region = var.aws_region
}
#Create security group with firewall rules
resource "aws_security_group" "security_jenkins_port" {
name = "security_jenkins_port"
description = "security group for jenkins"
ingress {
from_port = 8080
to_port = 8080
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
# outbound from jenkis server
egress {
from_port = 0
to_port = 65535
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
tags= {
Name = "security_jenkins_port"
}
}
resource "aws_instance" "myFirstInstance" {
ami = "ami-02d26659fd82cf299"
key_name = var.key_name
instance_type = var.instance_type
security_groups= [ "security_jenkins_port"]
tags= {
Name = "jenkins_instance"
}
}
```

.

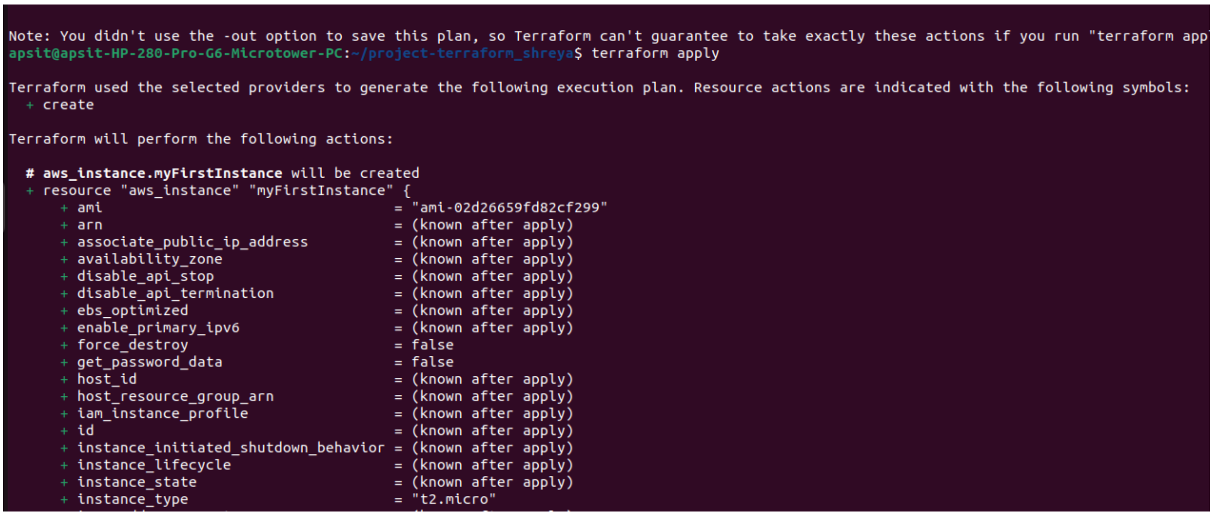


```
/home/apsit/project-terraform_shreya
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ ls
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ sudo nano variables.tf
[sudo] password for apsit:
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ sudo nano main.tf
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.13.0...
- Installed hashicorp/aws v6.13.0 (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.
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ terraform plan

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

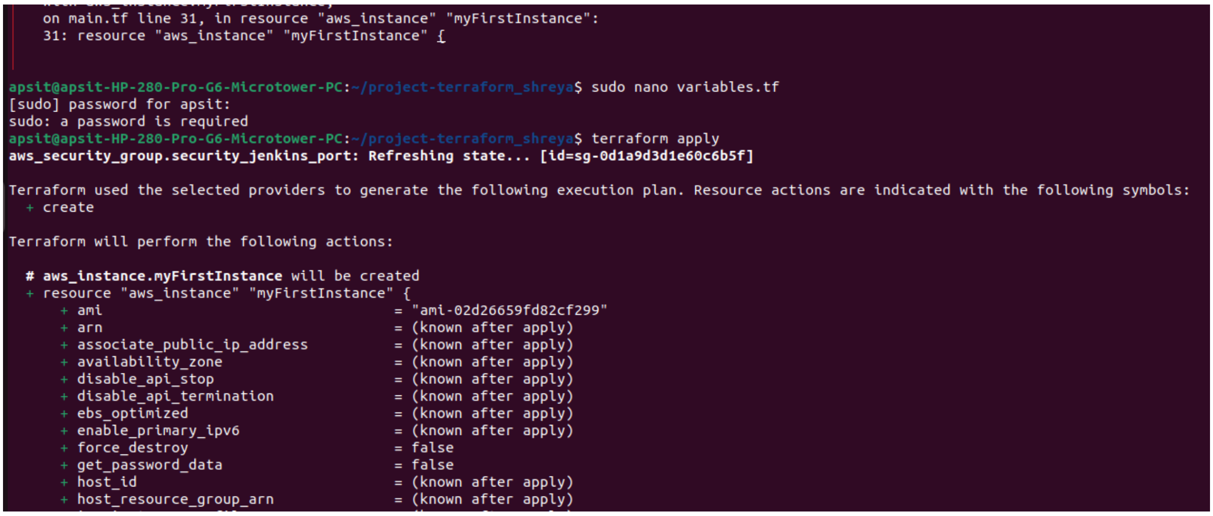


```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ terraform plan

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.myFirstInstance will be created
  + resource "aws_instance" "myFirstInstance" {
      + ami                                  = "ami-02d26659fd82cf299"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
```

.

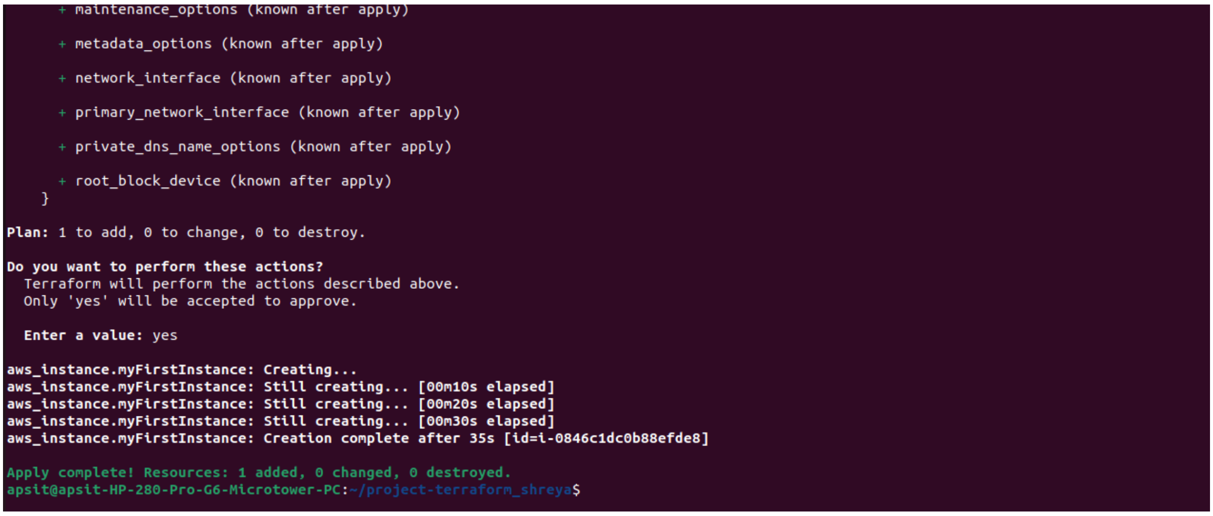


```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform app
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ terraform apply

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.myFirstInstance will be created
  + resource "aws_instance" "myFirstInstance" {
      + ami                                  = "ami-02d26659fd82cf299"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
```



```
    on main.tf line 31, in resource "aws_instance" "myFirstInstance":
    31: resource "aws_instance" "myFirstInstance" {


apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ sudo nano variables.tf
[sudo] password for apsit:
sudo: a password is required
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$ terraform apply
aws_security_group.security_jenkins_port: Refreshing state... [id=sg-0d1a9d3d1e60c6b5f]

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.myFirstInstance will be created
  + resource "aws_instance" "myFirstInstance" {
      + ami                                  = "ami-02d26659fd82cf299"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
```

.



```
      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + primary_network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

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.myFirstInstance: Creating...
aws_instance.myFirstInstance: Still creating... [00m10s elapsed]
aws_instance.myFirstInstance: Still creating... [00m20s elapsed]
aws_instance.myFirstInstance: Still creating... [00m30s elapsed]
aws_instance.myFirstInstance: Creation complete after 35s [id=i-0846c1dc0b88efde8]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform_shreya$
```

Conclusion: In this experiment, we build a AWS infrastructure , changeed it and destroyed it.