



PARSHVANATH CHARITABLE TRUST'S

**A. P. SHAH INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**(NBA Accredited)**



**Semester: V**  
**Academic Year: 2024-25**  
**Class / Branch: TE IT**  
**Subject: Advanced Devops Lab (ADL)**  
**Name of Instructor: Prof. Manjusha K.**

**Name of Student: Chirag Jayesh Malde**  
**Student ID: 22104186**

## EXPERIMENT NO. 06

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

### Pre-requisites:

#### 1. Install the AWS CLI version 2 on Linux

Follow these steps from the command line to install the AWS CLI on Linux.

**Install curl on linux**

```
root@apsit:~# sudo apt-get install curl
```

```
vishal@apsit:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
curl is already the newest version (7.58.0-2ubuntu3.24).
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit# curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 57.9M  100 57.9M    0     0  45.1M      0  0:00:01  0:00:01 --:--:--  45.1M
```

```
vishal@apsit:~$ sudo apt install unzip
```

```
root@apsit:~# sudo apt install unzip
```

```
vishal@apsit:~$ sudo unzip awscliv2.zip
```

```
root@apsit:~# sudo unzip awscliv2.zip
```

```
vishal@apsit:~$ sudo ./aws/install
```

```
vishal@apsit:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

Department of Information Technology | APSIT



PARSHVANATH CHARITABLE TRUST'S

**A. P. SHAH INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**(NBA Accredited)**



**vishal@apsit:~\$ aws --version**

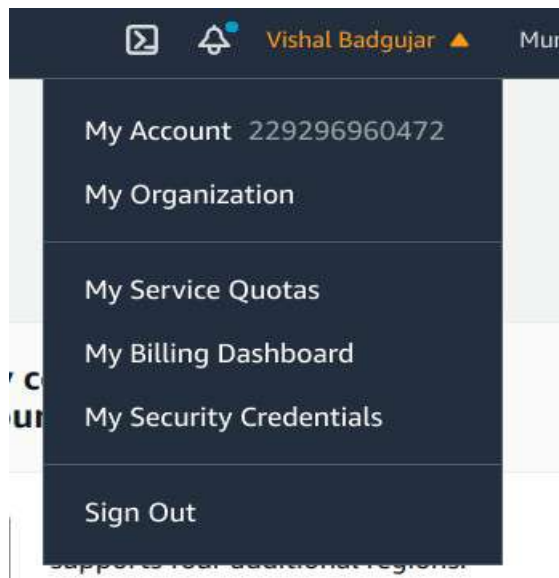
it should display the below output.

**aws-cli/2.1.29 Python/3.8.8 Linux/5.4.0-1038-aws exe/x86\_64.ubuntu.18 prompt/off**

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ aws --version
aws-cli/2.13.11 Python/3.11.4 Linux/5.4.0-150-generic exe/x86_64.ubuntu.18 prompt/off
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$
```

**2. Create a new access key if you don't have one. Make sure you download the keys in your local machine.**

Login to AWS console, click on username and go to My security credentials.





## Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#)

▲ Password

▲ Multi-factor authentication (MFA)

▼ Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS : time.

For your protection, you should never share your secret keys with anyone. As a best practice, we rec  
**If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key a**

Created	Access Key ID	Last Used
---------	---------------	-----------

Continue on security credentials, click on access keys

## Perform below commands in Linux where you have installed Terraform

First setup your access keys, secret keys and region code locally.

**vishal@apsit:~\$aws configure**

Created	Access Key ID
Jun 4th 2021	AKIATKYZ...
Aug 1st 2021	AKIATKYZ...

You can check region as

Vishal Badgujar ▼ Mumbai ▲

US East (N. Virginia) us-east-1

US East (Ohio) us-east-2

US West (N. California) us-west-1

US West (Oregon) us-west-2

Africa (Cape Town) af-south-1

Asia Pacific (Hong Kong) ap-east-1

**Asia Pacific (Mumbai) ap-south-1**

Asia Pacific (Osaka) ap-northeast-3

Asia Pacific (Seoul) ap-northeast-2

Asia Pacific (Singapore) ap-southeast-1

Asia Pacific (Sydney) ap-southeast-2

Asia Pacific (Tokyo) ap-northeast-1

Canada (Central) ca-central-1

Europe (Frankfurt) eu-central-1

Europe (Ireland) eu-west-1

Last Used Region	Last Used Service	Status
us-east-1	sts	Active
N/A	N/A	Active

shown in below image :



```
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit# aws configure
AWS Access Key ID [*****VK24]: AKIA52MEPFJLF24JHEWI
AWS Secret Access Key [*****2kOW]: bqELCwLPinYgmCwL9WBFlnh3kqfud7DxkxqWvI/
Default region name [us-east-1]: us-east-1
Default output format [None]:
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit# |
```

Create one Directory for Terraform project in which all files of terraform we can save

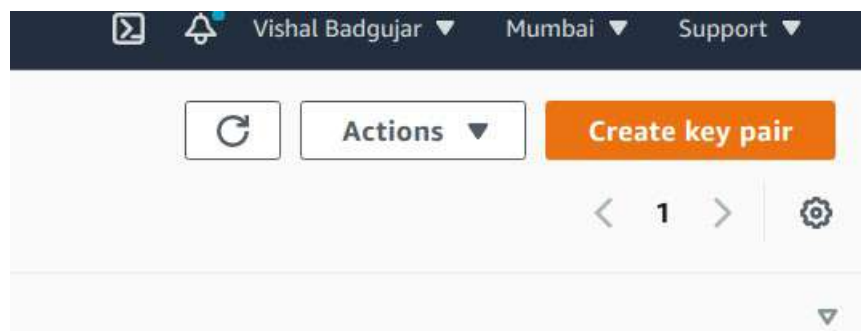
```
vishal@apsit:~$ cd ~
vishal@apsit:~$ mkdir project-terraform
vishal@apsit:~$ cd project-terraform
```

```
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit# mkdir project-terraform-exp6
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit# cd project-terraform-exp6
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit/project-terraform-exp6# |
```

### Create Terraform Files

```
vishal@apsit:~$ sudo nano variables.tf
```

In order to provide key name in variables first create key pair as shown:





Give name to key pair file as **terraform**

## Create key pair

### 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

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

Private key file format

☒ .pem  
For use with OpenSSH

☐ .ppk  
For use with PuTTY

Tags (Optional)

No tags associated with the resource.

Add tag

You can add 50 more tags.

Cancel

Create key pair

Key pair is generated

<input type="checkbox"/>	terraform	d4:aa:d4:24:a8:f5:a2:2a:28:59:e6:38:d...	key-080872ef28d76fe24
--------------------------	-----------	--	-----------------------

Use your Region and Key name in variable.tf as shown and provide instance type which you want to create.





```
GNU nano 2.9.3 variables.tf

variable "aws_region" {
  description = "The AWS region to create things in."
  default = "ap-south-1"
}

variable "key_name" {
  description = "SSH keys to connect to ec2 instance"
  default = "terraform"
}

variable "instance_type" {
  description = "instance type for ec2"
  default = "t2.micro"
}
```

After creating variable terraform file note down the AMI ID of instance which u want to create which we will use to configure our instance in main.tf file.



**Amazon Linux 2 AMI (HVM), SSD Volume Type - [ami-04db49c0fb2215364](#)** (64-bit x86) / ami

Amazon Linux  
Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance c  
Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Li  
2020 and has been removed from this wizard.

**Now create main.tf file:**

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform-exp6$ sudo nano main.tf
```

```
provider "aws" {
  region = var.aws_region
```



PARSHVANATH CHARITABLE TRUST'S

**A. P. SHAH INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**(NBA Accredited)**



}

#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 jenkins 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-0b9064170e32bde34"
```

```
    key_name = var.key_name
```

```
    instance_type = var.instance_type
```

```
    security_groups= [ "security_jenkins_port"]
```

```
    tags= {
```

```
        Name = "jenkins_instance"
```

```
    }
```

```
}
```

```
# Create Elastic IP address
```

```
resource "aws_eip" "myFirstInstance" {
```

```
    vpc    = true
```

```
    instance = aws_instance.myFirstInstance.id
```

```
tags= {
```

```
    Name = "jenkins_elstic_ip"
```

```
}
```

```
}
```





PARSHVANATH CHARITABLE TRUST'S

**A. P. SHAH INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**(NBA Accredited)**



Put AMI-ID in above highlighted space and Now execute the below command:

```
~/project-terraform$ terraform init
```

you should see like below screenshot.

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/project-terraform-exp6$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.62.0

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-exp6$ |
```

Execute the below command

```
~/project-terraform$ terraform plan
```

the above command will show how many resources will be added.

Plan: 3 to add, 0 to change, 0 to destroy.

```
vishal@apsit:~/project-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_eip.myFirstInstance will be created
+ resource "aws_eip" "myFirstInstance" {
  + allocation_id      = (known after apply)
  + association_id     = (known after apply)
  + carrier_ip         = (known after apply)
```



PARSHVANATH CHARITABLE TRUST'S

**A. P. SHAH INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**(NBA Accredited)**



```
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit/project-terraform-exp6# 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_elp.myFirstInstance will be created
+ resource "aws_elp" "myFirstInstance" {
  + allocation_id      = (known after apply)
  + arn                = (known after apply)
  + association_id     = (known after apply)
  + carrier_ip         = (known after apply)
  + customer_owned_ip  = (known after apply)
  + domain             = (known after apply)
  + id                 = (known after apply)
  + instance           = (known after apply)
  + network_border_group = (known after apply)
  + network_interface  = (known after apply)
  + private_dns        = (known after apply)
  + private_ip         = (known after apply)
  + ptr_record         = (known after apply)
  + public_dns         = (known after apply)
  + public_ip          = (known after apply)
  + public_ipv4_pool    = (known after apply)
  + tags               = {
    + "Name" = "jenkins_elstic_ip"
  }
  + tags_all           = {
    + "Name" = "jenkins_elstic_ip"
  }
  + vpc                 = true
}
```

Execute the below command

```
:~/project-terraform$ terraform apply
```

Provide the value as Yes for applying terraform

```
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
```

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



PARSHVANATH CHARITABLE TRUST'S  
**A. P. SHAH INSTITUTE OF TECHNOLOGY**  
Department of Information Technology  
(NBA Accredited)



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

```
File Edit View Search Terminal Help
+ network_interface (known after apply)
+ private_dns_name_options (known after apply)
+ root_block_device (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

Warning: Argument is deprecated
  with aws_elb.myFirstInstance1,
  on main.tf line 43, in resource "aws_elb" "myFirstInstance1":
  43: vpc = true

use domain attribute instead
(and one more similar warning elsewhere)

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.myFirstInstance1: Creating...
aws_instance.myFirstInstance1: Still creating... [10s elapsed]
aws_instance.myFirstInstance1: Still creating... [20s elapsed]
aws_instance.myFirstInstance1: Still creating... [30s elapsed]
aws_instance.myFirstInstance1: Creation complete after 35s [id=i-0be30baa225056deb]
aws_elb.myFirstInstance1: Creating...
aws_elb.myFirstInstance1: Creation complete after 3s [id=elballoc-0dfad66b1eb3b48c9]

Warning: Argument is deprecated
  with aws_elb.myFirstInstance1,
  on main.tf line 43, in resource "aws_elb" "myFirstInstance1":
  43: vpc = true

use domain attribute instead

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
root@apslt-HP-280-Pro-G6-Microtower-PC:/home/apslt/project-terraform-exp6#
```

Now login to EC2 console, to see the new instances up and running, you can see Jenkins\_instance is up and running which we deploy from terraform.

Instances (2) Info

Filter instances

Instance state: running X Clear filters

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	BitnamiMoodleCFDb01Ec2Instance	i-07ca078b9bcb1598b	Running	t3a.medium	2/2 checks passed	No alarms +	ap-south-1a
<input type="checkbox"/>	jenkins_instance	i-0a4a0fb7e55252d0f	Running	t2.micro	2/2 checks passed	No alarms +	ap-south-1a

EC2 > Security Groups > sg-0f04dc9c71cdf3dd - security\_jenkins\_port

sg-0f04dc9c71cdf3dd - security\_jenkins\_port

Details

Security group name	Security group ID	Description	VPC ID
security_jenkins_port	sg-0f04dc9c71cdf3dd	security group for jenkins	vpc-18be7c73
Owner	Inbound rules count	Outbound rules count	
229296960472	2 Permission entries	1 Permission entry	

Inbound rules

Outbound rules

Tags

You can now check network connectivity with Reachability Analyzer

Run Reachability Analyzer

Inbound rules (2)

	Name	Security group rule...	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-072ea72c21e715fa8	IPv4	SSH	TCP	22	0.0.0.0/0
<input type="checkbox"/>	-	sgr-022c2f4b64a5b9934	IPv4	Custom TCP	TCP	8080	0.0.0.0/0





PARSHVANATH CHARITABLE TRUST'S

# A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



You can also check the security group resource details which you created from terraform :

## Terraform destroy

you can also destroy or delete your instance by using terraform destroy command :

```
~/project-terraform$ terraform destroy
```

```
Plan: 0 to add, 0 to change, 3 to destroy.
```

### Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.  
There is no undo. Only 'yes' will be accepted to confirm.

```
Enter a value: yes
```

```
File Edit View Search Terminal Help
- revoke_rules_on_delete = false -> null
- tags
  - Name = "security_jenkins_port"
  - Name = "security_jenkins_port"
- vpc_id = "vpc-07a3531d3ad732038" -> null
# (1 unchanged attribute hidden)
}

Plan: 0 to add, 0 to change, 3 to destroy.

Warning: Argument is deprecated
  with aws_eip.myFirstInstance1,
  on main.tf line 43, in resource "aws_eip" "myFirstInstance1":
   43: vpc = true

use domain attribute instead

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_eip.myFirstInstance1: Destroying... [id=elalloc-0dfad66b1eb3b48c9]
aws_security_group.security_jenkins_port: Destroying... [id=sg-0eff1d7e4f94d8c92]
aws_eip.myFirstInstance1: Destruction complete after 6s
aws_instance.myFirstInstance1: Destroying... [id=i-0be30baa225056deb]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0eff1d7e4f94d8c92, 10s elapsed]
aws_instance.myFirstInstance1: Still destroying... [id=i-0be30baa225056deb, 10s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0eff1d7e4f94d8c92, 20s elapsed]
aws_instance.myFirstInstance1: Still destroying... [id=i-0be30baa225056deb, 20s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0eff1d7e4f94d8c92, 30s elapsed]
aws_instance.myFirstInstance1: Still destroying... [id=i-0be30baa225056deb, 30s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0eff1d7e4f94d8c92, 40s elapsed]
aws_instance.myFirstInstance1: Still destroying... [id=i-0be30baa225056deb, 40s elapsed]
aws_instance.myFirstInstance1: Destruction complete after 46s
aws_instance.myFirstInstance1: Destruction complete after 46s

Destroy complete! Resources: 3 destroyed.
root@apsit-HP-280-Pro-G6-Microtower-PC:/home/apsit/project-terraform-exp6#
```

Now you can see instance which you created by using terraform is deleted successfully from aws console also you can check it will removed successfully:

Instances (1) Info							Connect	Instance state	Actions	Launch instances
Filter instances							< 1 >			
Instance state: running X Clear filters										
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone			
<input type="checkbox"/>	BitnamiMoodleCfDb01Ec2Instance	i-07ca078b9bcb1598b	Running	t3a.medium	2/2 checks passed	No alarms	ap-south-1a			



PARSHVANATH CHARITABLE TRUST'S

**A. P. SHAH INSTITUTE OF TECHNOLOGY**

**Department of Information Technology**

**(NBA Accredited)**



All the Resources including Security groups, EC2 instances using terraform will be deleted. In this way we can automate infrastructure set up using terraform in aws cloud.

**Conclusion:** The experiment demonstrated how Terraform simplifies AWS infrastructure management by allowing easy creation, modification, and deletion of resources through configuration files. This automation streamlines the deployment process and ensures consistent, repeatable infrastructure setups.