



**Academic Year: 2022-23**

**Semester: V**

**Class / Branch: TE IT**

**Subject: Advanced Devops Lab (ADL)**

**Subject Lab Incharge: Prof. Manjusha K.**

---

### EXPERIMENT NO. 06

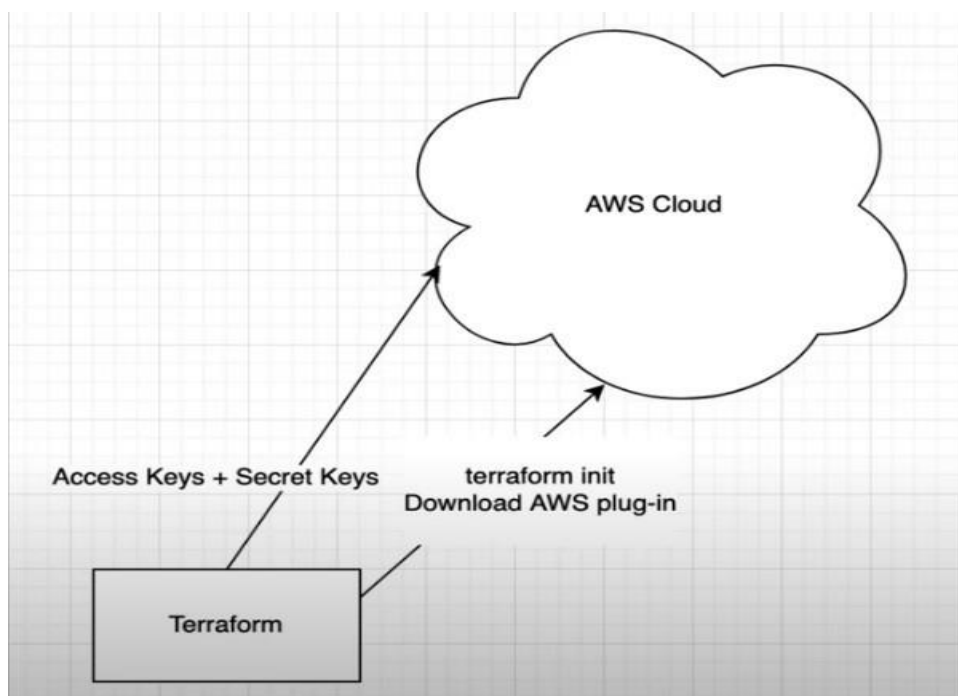
**Aim: To Build, change, and destroy AWS infrastructure Using Terraform.**

#### Theory:

Hashicorp's Terraform is an open-source tool for provisioning and managing cloud infrastructure. Terraform can provision resources on any cloud platform.

Terraform allows you to create infrastructure in configuration files(**tf files**) that describe the topology of cloud resources. These resources include virtual machines, storage accounts, and networking interfaces.

We will see how you can use Terraform to provision EC2 instance. Please do the below steps for provisioning EC2 instances on AWS:





## 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

```
vishal@apsit:~$ sudo apt-get install curl
```

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

```
vishal@apsit:~$ 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 41.8M  100 41.8M    0     0 2529k      0  0:00:16  0:00:16 --:--:-- 2555k
```

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

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

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

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

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

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

```
vishal@apsit:~$ aws --version
```

it should display the below outout.

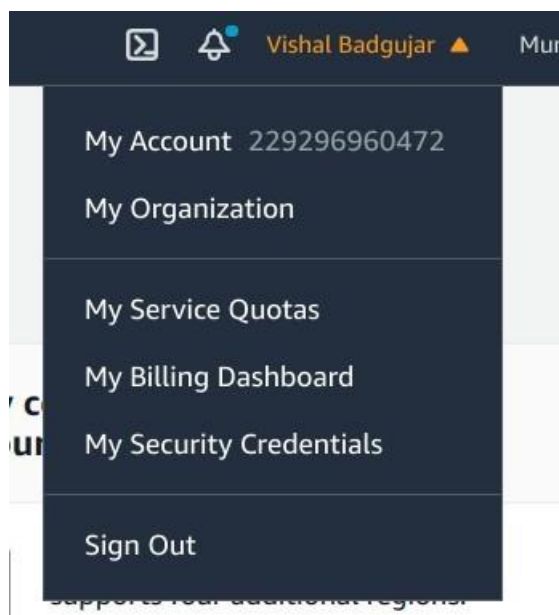
```
aws-cli/2.1.29 Python/3.8.8 Linux/5.4.0-1038-aws exe/x86_64.ubuntu.18 prompt/off
```



```
vishal@apsit:~$ aws --version  
aws-cli/2.2.25 Python/3.8.8 Linux/5.4.0-80-generic exe/x86_64.ubuntu.18 prompt/off
```

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.



Continue on security credentials, click on access keys

## 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](#) in the

▲ 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 SDKs, and other tools.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend that you rotate your access keys regularly. **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
---------	---------------	-----------



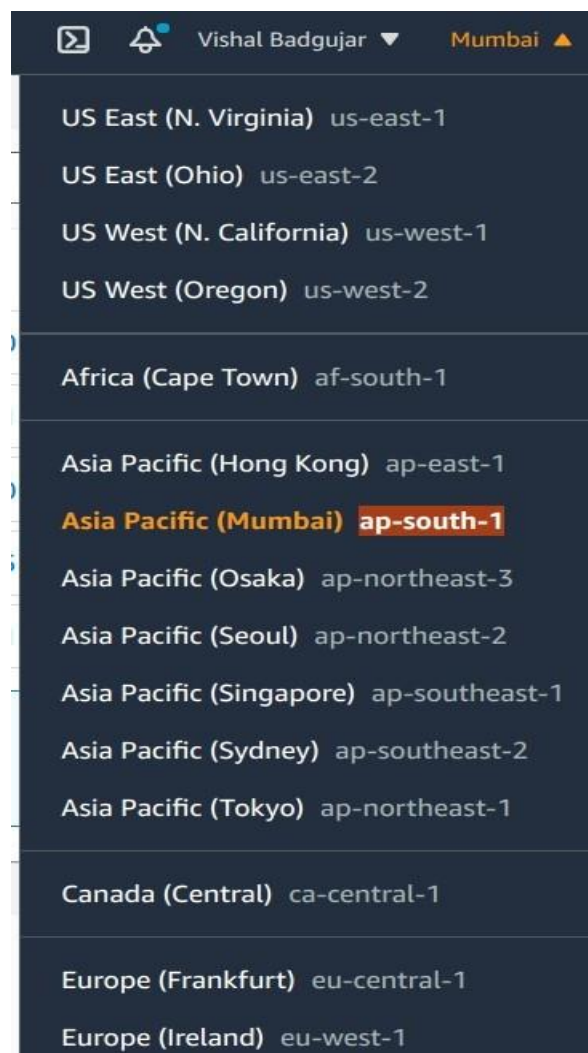
## Perform below commands in Linux where you have installed Terraform

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

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

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status
Jun 4th 2021	AKIATKYZJ6PMCN2VF436	2021-07-04 21:26 UTC+0530	us-east-1	sts	Active
Aug 1st 2021	AKIATKYZJ6PMFLTCGGPV	N/A	N/A	N/A	Active

You can check region as shown in below image :





```
vishal@apsit:~$ aws configure
AWS Access Key ID [None]: AKIATKYZJ6PMFLTCGGPV
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]:
vishal@apsit:~$
```

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

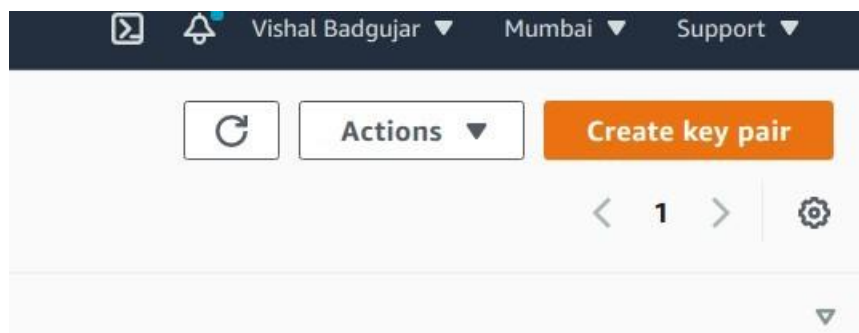
```
vishal@apsit:~$ cd ~
vishal@apsit:~$ mkdir project-terraform
vishal@apsit:~$ cd project-terraform
```

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

### 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.

CancelCreate 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.

```
File Edit View Search Terminal Help
GNU nano 2.9.3 variables.tf Modified

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"
}

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell   ^_ Go To Line  M-E Redo
```

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  
Free tier eligible

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

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.



PARSHVANATH CHARITABLE TRUST'S

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

**Department of Information Technology**

**(NBA Accredited)**



**Now create main.tf file:**

```
vishal@apsit:~/project-terraform$ sudo nano 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 jenkins server
```





PARSHVANATH CHARITABLE TRUST'S

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

**Department of Information Technology**

**(NBA Accredited)**



```
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
```



PARSHVANATH CHARITABLE TRUST'S

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

**Department of Information Technology**

**(NBA Accredited)**



```
tags= {  
    Name = "jenkins_elstic_ip"  
}  
}
```

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

```
vishal@apsit:~/project-terraform$ terraform init
```

you should see like below screenshot.

```
vishal@apsit:~/project-terraform$ terraform init  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding latest version of hashicorp/aws...  
- Installing hashicorp/aws v3.52.0...  
- Installed hashicorp/aws v3.52.0 (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.
```



Execute the below command

```
vishal@apsit:~/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)
```

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

Execute the below command

```
vishal@apsit:~/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

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

```
Plan: 2 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... [10s elapsed]
```

```
aws_instance.myFirstInstance: Still creating... [20s elapsed]
```

```
aws_instance.myFirstInstance: Still creating... [30s elapsed]
```

```
aws_instance.myFirstInstance: Creation complete after 32s [id=i-0a4a0fb7e55252d0f]
```

```
aws_eip.myFirstInstance: Creating...
```

```
aws_eip.myFirstInstance: Creation complete after 1s [id=eipalloc-0fd8f60524b10fc93]
```

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

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.

	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



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

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

### sg-0f04dc9c71cdf3dd - security\_jenkins\_port

Details

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

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

#### Inbound rules (2)

Filter security group rules

	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	-

## Terraform destroy

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

```
vishal@apsit:~/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



PARSHVANATH CHARITABLE TRUST'S

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

**Department of Information Technology**

**(NBA Accredited)**



Enter a value: yes

```
aws_eip.myFirstInstance: Destroying... [id=eipalloc-0fd8f60524b10fc93]
aws_security_group.security_jenkins_port: Destroying... [id=sg-0f04dc9c71cdf3dd]
aws_eip.myFirstInstance: Destruction complete after 2s
aws_instance.myFirstInstance: Destroying... [id=i-0a4a0fb7e55252d0f]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdf3dd, 10s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 10s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdf3dd, 20s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 20s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdf3dd, 30s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 30s elapsed]
aws_security_group.security_jenkins_port: Destruction complete after 38s
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 40s elapsed]
aws_instance.myFirstInstance: Destruction complete after 40s

Destroy complete! Resources: 3 destroyed.
```

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:

	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

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: Write your own findings.**