

ame: Hrishikesh Kumbhar

Div: D15A

Roll no: 32

Sub: Advanced DevOps

Experiment No: 6

Date: 31/08/2022

**Aim:** To Build, change, and destroy AWS /GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.

**Steps:**

## **A) Installation and Configuration of Terraform in Windows**

### **Step 1: Download terraform**

**To install Terraform, First Download the Terraform Cli Utility for windows from terraforms official website**

**website:**<https://www.terraform.io/downloads.html>

**Select the Operating System Windows followed by either 32bit or 64 bit based on your OS type.**

## **Download Terraform**

macOS

**Windows**

Linux

FreeBSD

OpenBSD

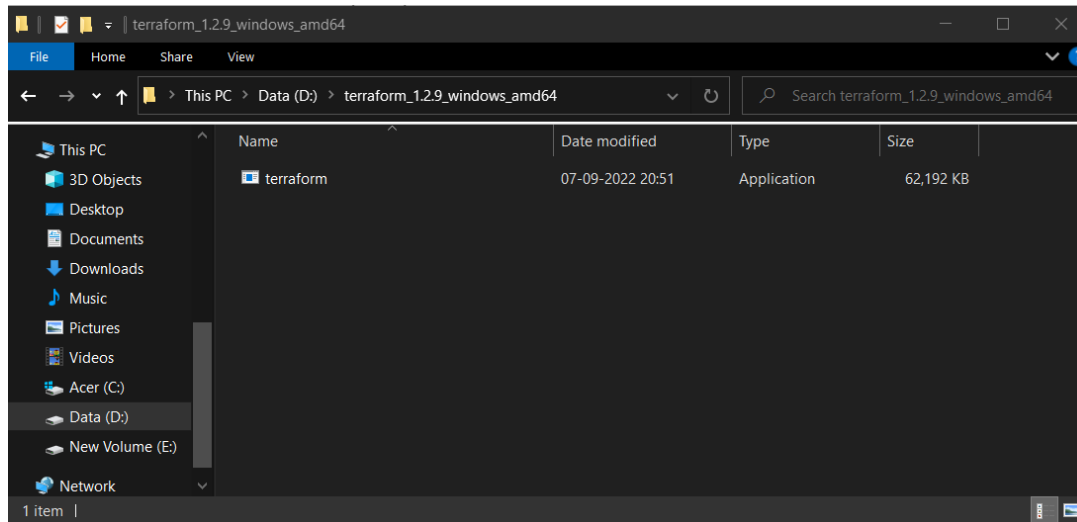
Solaris

WINDOWS BINARY DOWNLOAD

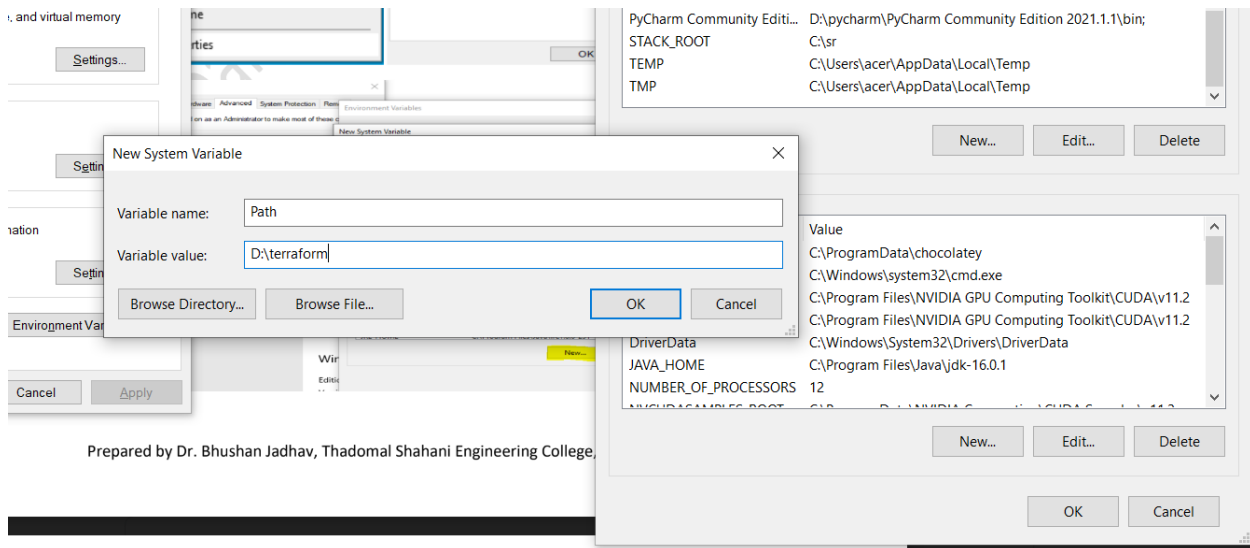
 **Terraform** 1.2.9

[386](#) [Amd64](#)

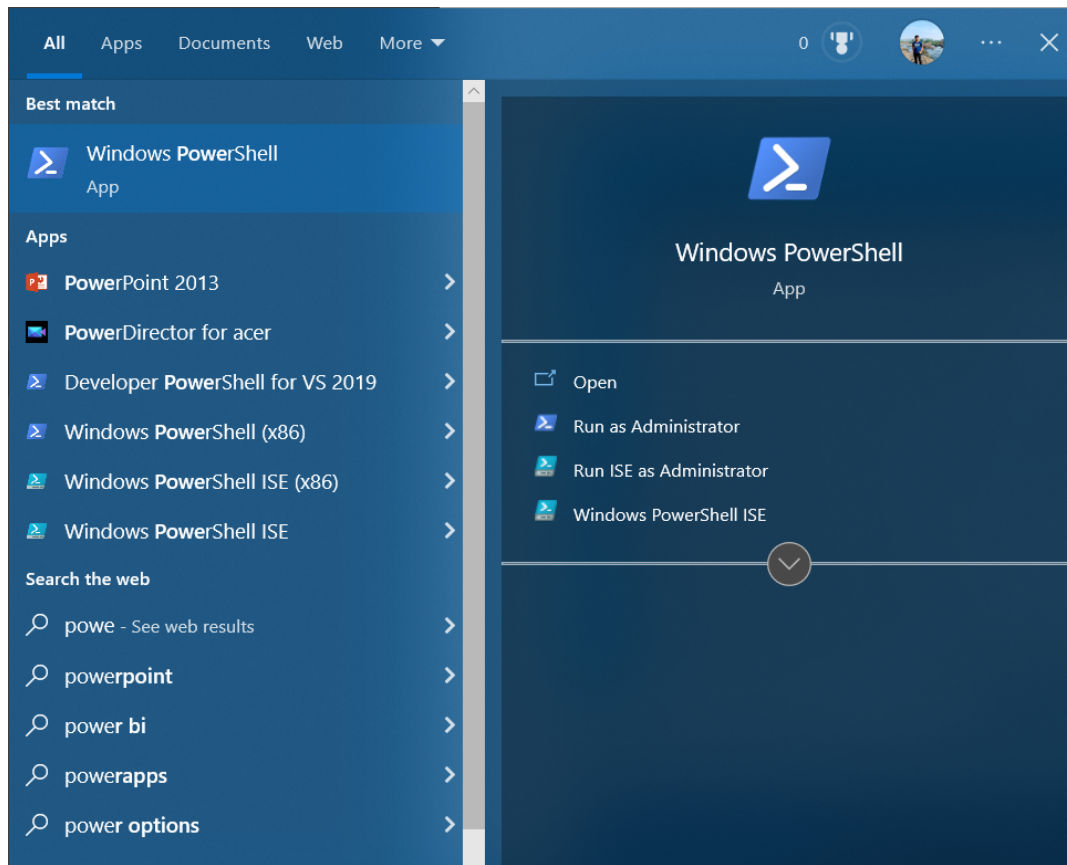
## Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory



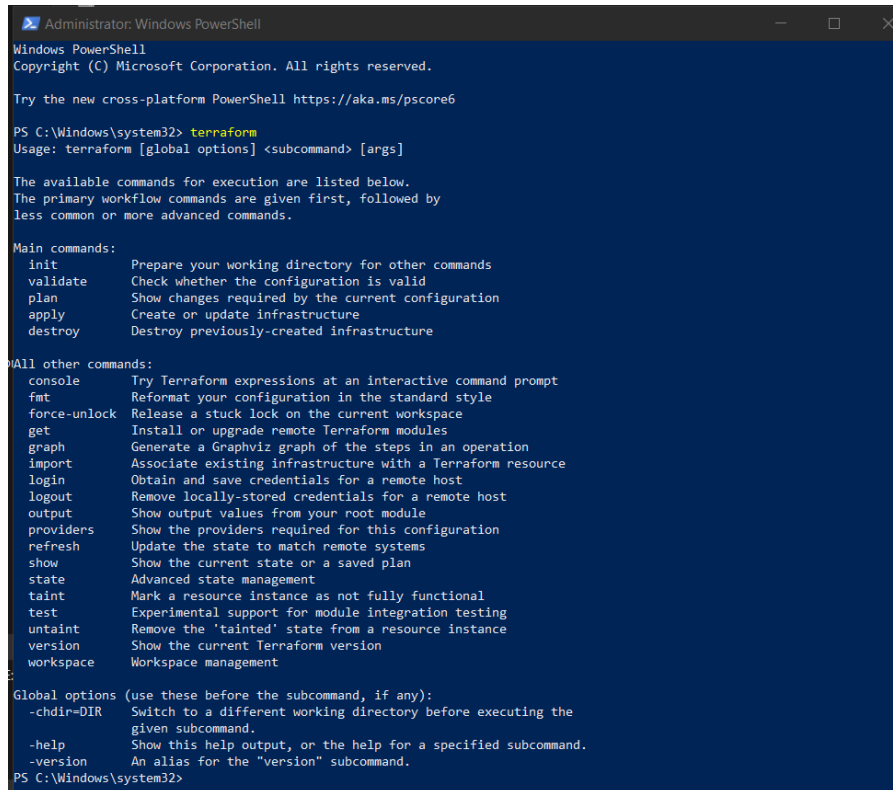
## Step 3: Set the System path for Terraform in Environment Variables.



## Step 4: Open PowerShell with Admin Access



## Step 5 : Open Terraform in PowerShell and check its functionality

A screenshot of a Windows PowerShell terminal window titled "Administrator: Windows PowerShell". The terminal shows the output of the 'terraform' command. It displays the Terraform version (0.12.24), copyright information, and a list of available commands categorized into 'Main commands' and 'All other commands'. The 'Main commands' include init, validate, plan, apply, and destroy. The 'All other commands' include console, fmt, force-unlock, get, graph, import, login, logout, output, providers, refresh, show, state, taint, test, untaint, version, and workspace. Global options like -chdir, -help, and -version are also listed.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> 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
  output     Show output values from your root module
  providers  Show the providers required for this configuration
  refresh    Update the state to match remote systems
  show       Show the current state or a saved plan
  state      Advanced state management
  taint      Mark a resource instance as not fully functional
  test       Experimental support for module integration testing
  untaint    Remove the 'tainted' state from a resource instance
  version    Show the current Terraform version
  workspace  Workspace management

Global options (use these before the subcommand, if any):
  -chdir=DIR  Switch to a different working directory before executing the
              given subcommand.
  -help       Show this help output, or the help for a specified subcommand.
  -version    An alias for the "version" subcommand.

PS C:\Windows\system32>
```

**Note:** If any error comes, then please recheck or set the path of Terraform in Environment variable again.

## B)Creating S3 Bucket using terraform

### Prerequisite:

1) Download and Install Docker Desktop from <https://www.docker.com/products/docker-desktop>

### Step 1: Check the docker functionality

```
Command Prompt
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\acer>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\\Users\\acer\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\\Users\\acer\\.docker\\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\\Users\\acer\\.docker\\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\\Users\\acer\\.docker\\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit
```

```
C:\Users\acer>docker --version
Docker version 20.10.17, build 100c701

C:\Users\acer>
```

## Step 2: Write a terraform script to create a Ubuntu Linux container

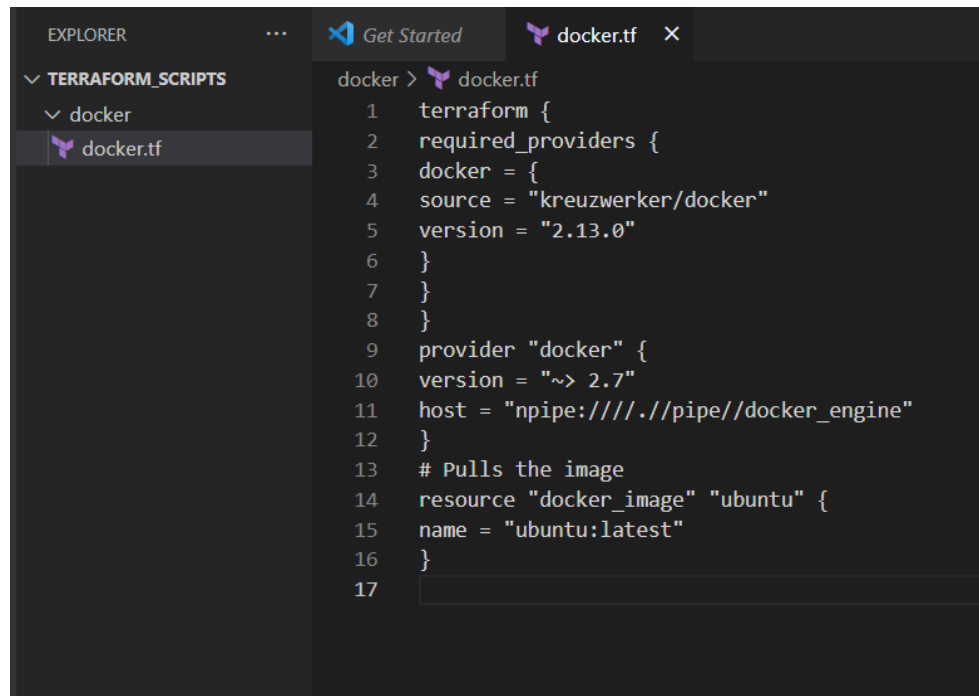
```
Script: docker.tf
*****
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version = "2.13.0"
    }
  }
}

provider "docker" {
  version = "~> 2.7"
  host = "npipe:////./pipe//docker_engine"
}

# Pulls the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

*****
```

Save the file in a new directory called docker where rest of the terraform scripts are stored



**Step 3: Open Command Prompt and go to Terraform\_Scripts\docker directory where our .tf file is stored**

**Step 4: Execute Terraform Init command to initialize the resources**

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\docker> terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding kreuzwerker/docker versions matching "~> 2.7, 2.13.0"...
- Installing kreuzwerker/docker v2.13.0...
- Installed kreuzwerker/docker v2.13.0 (self-signed, key ID 24E54F214569A8A5)

Partner and community providers are signed by their developers.  
If you'd like to know more about provider signing, you can read about it here:  
<https://www.terraform.io/docs/cli/plugins/signing.html>

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.

**Warning:** Version constraints inside provider configuration blocks are deprecated

on docker.tf line 10, in provider "docker":  
10: version = "~> 2.7"

Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is now deprecated and will be removed in a future version of Terraform. To silence this warning, move the provider version constraint into the required\_providers block.

**Terraform has been successfully initialized!**

**Step 5: Execute Terraform plan to see the available resources**

// error

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\docker> terraform plan
```

**Warning:** Version constraints inside provider configuration blocks are deprecated

on docker.tf line 10, in provider "docker":  
10: version = "~> 2.7"

Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is now deprecated and will be removed in a future version of Terraform. To silence this warning, move the provider version constraint into the required\_providers block.

**Error:** Error ping Docker server: error during connect: Get "http://%2F%2F.%2F%2Fpipe%2F%2Fdocker\_engine/\_ping": open //./pipe//docker\_engine: The system cannot find the file specified.

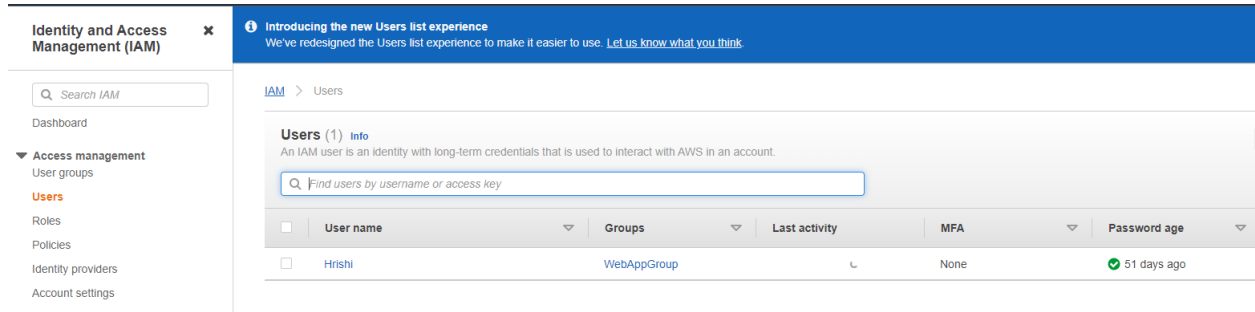
with provider["registry.terraform.io/kreuzwerker/docker"],  
on docker.tf line 9, in provider "docker":  
9: provider "docker" {

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\docker> █
```



# Creating an EC2 instance using Terraform:

## Step1: Create a new IAM user in AWS



## Enter a user name and choose Access Key credential type.

### Add user



### Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*

[+ Add another user](#)

### Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)


- Select AWS credential type\*
- ☒ **Access key - Programmatic access**  
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
  - ☐ **Password - AWS Management Console access**  
Enables a **password** that allows users to sign-in to the AWS Management Console.


## Create a new user group and provide Administrator Access


### Add user

1 2 3 4 5

#### Set permissions

 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

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](#)

#### Add user to group

Create group Refresh

Q Search		Showing 1 result
Group	Attached policies	
<input type="checkbox"/> WebAppGroup	AWSCloud9EnvironmentMember	

#### Set permissions boundary

Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job function.

Group name

Create policy Refresh

Filter policies

Q Search

	Policy name	Type	Used as	
<input checked="" type="checkbox"/>	AdministratorAccess	Job function	None	P
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	None	G
<input type="checkbox"/>	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None	G
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	None	P
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	None	G
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS managed	None	P
<input type="checkbox"/>	AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	None	P
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None	P
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	AWS managed	None	P
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	AWS managed	None	P

Add key and value tag:

Add user

- 1
- 2
- 3
- 4
- 5

Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. [Learn more](#)

Key	Value (optional)	Remove
<input type="text" value="TerraformUserKey"/>	<input type="text" value="TerraformUserValue"/>	✕
<input type="text" value="Add new key"/>	<input type="text"/>	

You can add 49 more tags.

Review changes and create the user.

Add user

- 1
- 2
- 3
- 4
- 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	TerraformUser
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The user shown above will be added to the following groups.

Type	Name
Group	<a href="#">TerraformGroup</a>

Tags

The new user will receive the following tag

Key	Value
TerraformUserKey	TerraformUserValue

Copy the Access Key ID and Secret Key and save them in a Notepad.

Add user

12345

✓ Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://504827858021.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key
▶	✓ TerraformUser	<div></div>	***** <a>Show</a>

**Step2: Create a new folder and create a new .tf file in which we'll have the script**

**First Create an EC2 instance on AWS:**

▼ Application and OS Images (Amazon Machine Image) Info

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

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

RecentsQuick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

S

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type

ami-06489866022a12a14

(64-bit (x86)) / 

ami-0e18b1d379af4e263

 (64-bit (Arm))

Virtualization: hvm    ENA enabled: true    Root device type: ebs

Free tier eligible ▼

Instances (1) Info

Refresh

Connect

Instance state

Actions

Launch instances

Find instance by attribute or tag (case-sensitive)

<

1

>

Refresh

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
<input type="checkbox"/>	-	i-0476140ffa3fcb03	Pending	t2.micro	-	No alarms	ap-south-1b	ec2-3-110-104

**Create a new folder and create a new .tf file in which we'll have the script.**

```
mkdir terraform_scripts
```

```
gedit ec2-on-terraform.tf
```

```
provider "aws" {
  access_key = "<Key>"
  secret_key = "<Key>"
  region = "ap-south-1"
}

resource "aws_instance" "terraform-ec2" {
  ami = "<ami-code>"
  instance_type="t2.micro"
}
```

ec2-on-terraform.tf ✕

```
terraform_scripts > ec2-on-terraform.tf
```

```
1 provider "aws" {
2   access_key = "REDACTED"
3   secret_key = "REDACTED"
4   region = "ap-south-1"
5 }
6 resource "aws_instance" "terraform-ec2" {
7   ami = "ami-06489866022e12a14"
8   instance_type="t2.micro"
9 }
10
```

### Step3: Perform Terraform Commands to initialize backend and then apply your scripts to start an EC2 instance.

#### terraform init

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\terraform_scripts> terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.30.0...
- Installed hashicorp/aws v4.30.0 (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.

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\terraform_scripts> █
```

#### terraform plan

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\terraform_scripts> 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.terraform-ec2 will be created
+ resource "aws_instance" "terraform-ec2" {
  + ami                    = "ami-06489866022e12a14"
  + arn                   = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone      = (known after apply)
  + cpu_core_count        = (known after apply)
  + cpu_threads_per_core   = (known after apply)
  + disable_api_stop       = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized          = (known after apply)
  + get_password_data      = false
  + host_id                = (known after apply)
  + host_resource_group_arn = (known after apply)
  + id                    = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_state         = (known after apply)
  + instance_type          = "t2.micro"
  + ipv6_address_count     = (known after apply)
  + ipv6_addresses         = (known after apply)
  + key_name               = (known after apply)
  + monitoring              = (known after apply)
```

## terraform apply

Note: You don't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply".  
PS D:\vesit\sem 5\ADV Dev Ops\Terraform\_Scripts\terraform\_scripts> 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.terraform-ec2 will be created
+ resource "aws_instance" "terraform-ec2" {
  + ami              = "ami-06489866022e12a14"
  + arn              = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone = (known after apply)
  + cpu_core_count   = (known after apply)
  + cpu_threads_per_core = (known after apply)
  + disable_api_stop  = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized     = (known after apply)
  + get_password_data = false
  + host_id            = (known after apply)
  + host_resource_group_arn = (known after apply)
  + id                = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_state     = (known after apply)
  + instance_type      = "t2.micro"
  + ipv6_address_count = (known after apply)
  + ipv6_addresses     = (known after apply)
  + key_name           = (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.

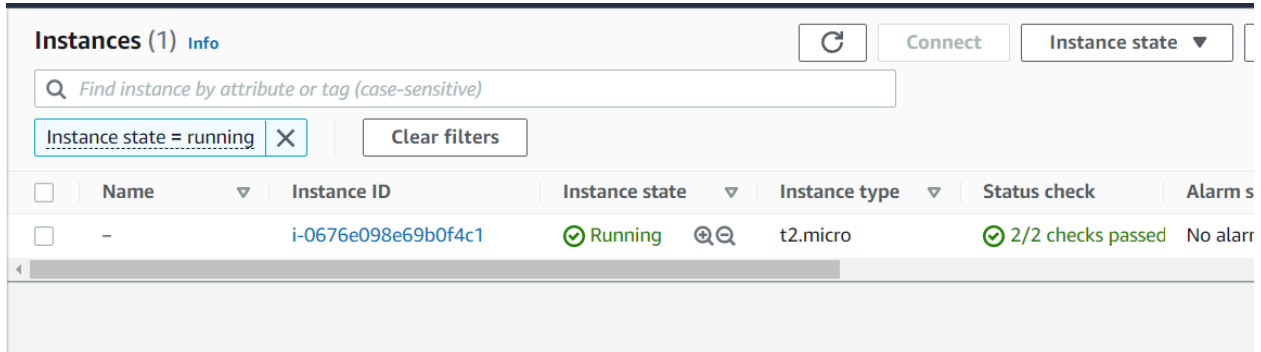
Enter a value: yes

```
aws_instance.terraform-ec2: Creating...
aws_instance.terraform-ec2: Still creating... [10s elapsed]
aws_instance.terraform-ec2: Still creating... [20s elapsed]
aws_instance.terraform-ec2: Still creating... [30s elapsed]
aws_instance.terraform-ec2: Creation complete after 32s [id=i-0676e098e69b0f4c1]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

PS D:\vesit\sem 5\ADV Dev Ops\Terraform\_Scripts\terraform\_scripts>

**Step4: Check if your instance is running in the AWS EC2 console.**



The screenshot shows the AWS Management Console 'Instances' page. At the top, there's a search bar with the text 'Find instance by attribute or tag (case-sensitive)'. Below it, a filter is applied: 'Instance state = running'. The main table lists instances with columns: Name, Instance ID, Instance state, Instance type, Status check, and Alarm s. One instance is listed with ID 'i-0676e098e69b0f4c1', state 'Running', type 't2.micro', and '2/2 checks passed'.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm s
<input type="checkbox"/>	-	i-0676e098e69b0f4c1	Running	t2.micro	2/2 checks passed	No alarm

**Step5: Use the destroy command to terminate the EC2 instance**

```
terraform destroy
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\terraform_scripts> terraform destroy
aws_instance.terraform-ec2: Refreshing state... [id=i-0676e098e69b0f4c1]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_instance.terraform-ec2 will be destroyed
- resource "aws_instance" "terraform-ec2" {
  - ami              = "ami-06489866022e12a14" -> null
  - arn              = "arn:aws:ec2:ap-south-1:504827858021:instance/i-0676e098e69b0f4c1" -> null
  - associate_public_ip_address = true -> null
  - availability_zone = "ap-south-1b" -> null
  - cpu_core_count    = 1 -> null
  - cpu_threads_per_core = 1 -> null
  - disable_api_stop   = false -> null
  - disable_api_termination = false -> null
  - ebs_optimized      = false -> null
  - get_password_data   = false -> null
  - hibernation         = false -> null
  - id                 = "i-0676e098e69b0f4c1" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state     = "running" -> null
  - instance_type      = "t2.micro" -> null
}
```

**Plan:** 0 to add, 0 to change, 1 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



```
aws_instance.terraform-ec2: Destroying... [id=i-0676e098e69b0f4c1]
aws_instance.terraform-ec2: Still destroying... [id=i-0676e098e69b0f4c1, 10s elapsed]
aws_instance.terraform-ec2: Still destroying... [id=i-0676e098e69b0f4c1, 20s elapsed]
aws_instance.terraform-ec2: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\terraform_scripts> |
```

**Step6: Check updates back in the Ec2 console. The instance should be terminated.**

Instances (2) <a href="#">Info</a>							<a href="#">Refresh</a>	<a href="#">Connect</a>	<a href="#">Instance state</a> ▼	<a href="#">Actions</a>
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/>										
<input type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status				
<input type="checkbox"/>	-	<a href="#">i-0676e098e69b0f4c1</a>	⊖ Terminated 🔍	t2.micro	-	No alarms +				

## Creating an S3 Bucket using Terraform

**Step1:** Create a new directory and two .tf files, provider.tf and s3-on-terraform.tf

provider.tf

```
provider "aws" {  
  access_key = "<Key>"  
  secret_key = "<Key>"  
  region = "ap-south-1"  
}
```

```
1  provider "aws" {  
2    access_key = "  
3    secret_key = "  
4    region = "ap-south-1"  
5  }
```

s3-on-terraform.tf

```
resource "aws_s3_bucket" "hrishi" {  
  bucket = "hk-terraform-test-bucket-0001"  
  acl = "public-read"  
  tags={  
    Name = "Hrishi S3 Bucket"  
    Environment = "Dev"  
  }  
}
```

## Step2: Perform Terraform commands.

```
terraform init
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\S3BucketTerraform> terraform init
```

Initializing the backend...

Initializing provider plugins...

- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.30.0...
- Installed hashicorp/aws v4.30.0 (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.

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\S3BucketTerraform> |
```

```
terraform plan
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\S3BucketTerraform> terraform plan
```

Acquiring state lock. This may take a few moments...

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_s3_bucket.hrishi will be created
+ resource "aws_s3_bucket" "hrishi" {
  + acceleration_status      = (known after apply)
  + acl                      = "public-read"
  + arn                     = (known after apply)
  + bucket                  = "hk-terraform-test-bucket-0001"
  + bucket_domain_name      = (known after apply)
  + bucket_regional_domain_name = (known after apply)
  + force_destroy           = false
  + hosted_zone_id          = (known after apply)
  + id                      = (known after apply)
  + object_lock_enabled     = (known after apply)
  + policy                  = (known after apply)
  + region                  = (known after apply)
  + request_payer           = (known after apply)
  + tags                    = {
    + "Environment" = "Dev"
    + "Name"       = "Hrishi S3 Bucket"
  }
  + tags_all              = {
    + "Environment" = "Dev"
    + "Name"       = "Hrishi S3 Bucket"
  }
}
```

## terraform apply

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\S3BucketTerraform> 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_s3_bucket.hrishi will be created
+ resource "aws_s3_bucket" "hrishi" {
  + acceleration_status = (known after apply)
  + acl                 = "public-read"
  + arn                 = (known after apply)
  + bucket              = "hk-terraform-test-bucket-0001"
  + bucket_domain_name = (known after apply)
  + bucket_regional_domain_name = (known after apply)
  + force_destroy       = false
  + hosted_zone_id      = (known after apply)
  + id                  = (known after apply)
  + object_lock_enabled = (known after apply)
  + policy              = (known after apply)
  + region              = (known after apply)
  + request_payer       = (known after apply)
  + tags                = {
    + "Environment" = "Dev"
    + "Name"        = "Hrishi S3 Bucket"
  }
  + tags_all            = {
    + "Environment" = "Dev"
    + "Name"        = "Hrishi S3 Bucket"
  }
}
```

**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\_s3\_bucket.hrishi: Creating...

aws\_s3\_bucket.hrishi: Creation complete after 2s [id=hk-terraform-test-bucket-0001]

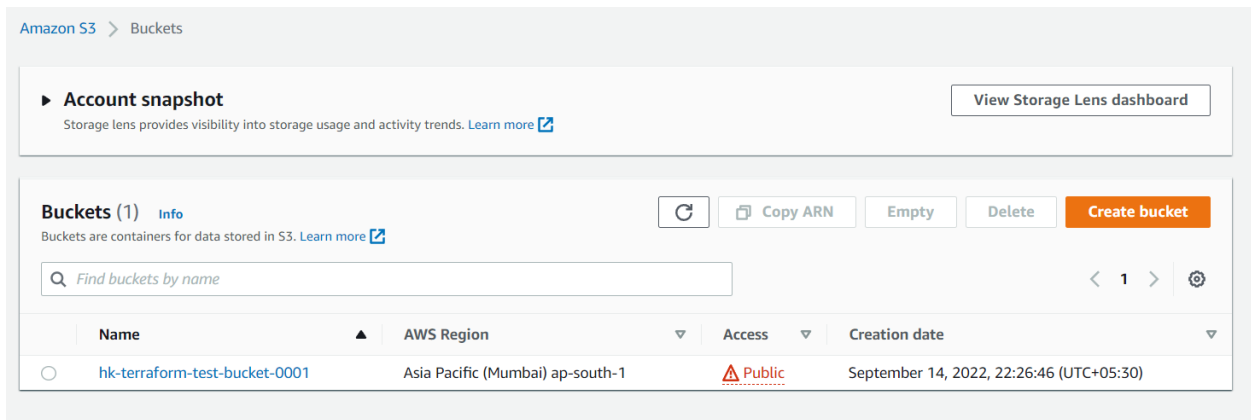
**Warning:** Argument is deprecated

with aws\_s3\_bucket.hrishi,  
on s3-on-terraform.tf line 3, in resource "aws\_s3\_bucket" "hrishi":  
3: acl = "public-read"

Use the aws\_s3\_bucket\_acl resource instead

**Apply complete! Resources: 1 added, 0 changed, 0 destroyed.**

**Step3: After terraform apply, check your S3 Console if you have a new bucket that you just created.**



**Step4: After you've created your bucket and verified it, you can delete this bucket by using terraform destroy.**

```
terraform destroy
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform Scripts\S3BucketTerraform> terraform destroy
aws_s3_bucket.hrishi: Refreshing state... [id=hk-terraform-test-bucket-0001]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_s3_bucket.hrishi will be destroyed
- resource "aws_s3_bucket" "hrishi" {
  - acl                = "public-read" -> null
  - arn                = "arn:aws:s3:::hk-terraform-test-bucket-0001" -> null
  - bucket             = "hk-terraform-test-bucket-0001" -> null
  - bucket_domain_name = "hk-terraform-test-bucket-0001.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "hk-terraform-test-bucket-0001.s3.ap-south-1.amazonaws.com" -> null
  - force_destroy      = false -> null
  - hosted_zone_id     = "Z11RGJOFQWJUP" -> null
  - id                 = "hk-terraform-test-bucket-0001" -> null
  - object_lock_enabled = false -> null
  - region             = "ap-south-1" -> null
  - request_payer       = "BucketOwner" -> null
  - tags               = {
    - "Environment" = "Dev"
    - "Name"        = "Hrishi S3 Bucket"
  } -> null
  - tags_all          = {
    - "Environment" = "Dev"
    - "Name"        = "Hrishi S3 Bucket"
  }
```

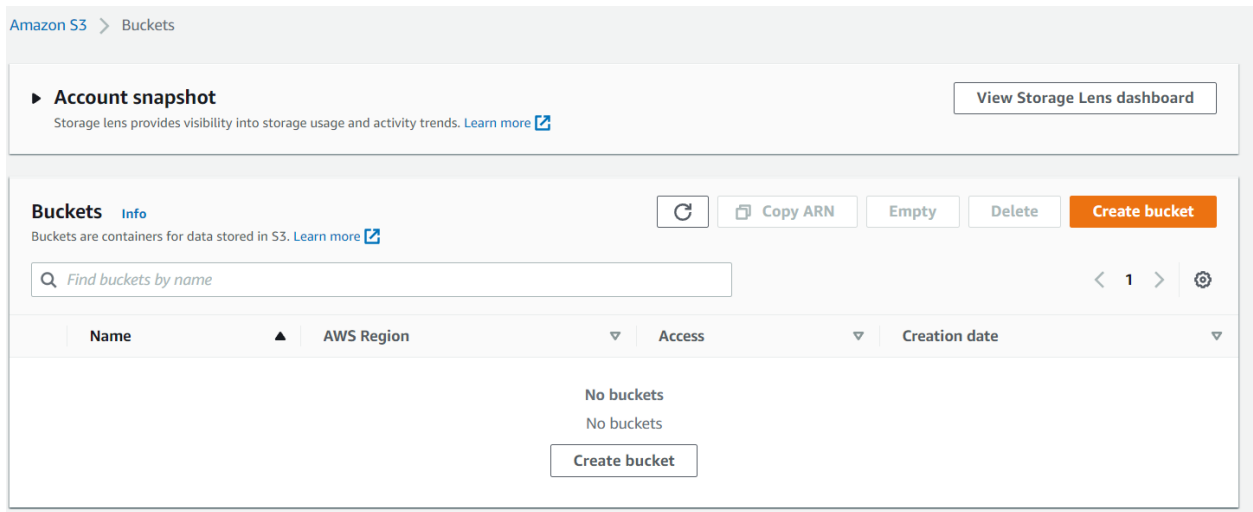
```
Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_s3_bucket.hrishi: Destroying... [id=hk-terraform-test-bucket-0001]
aws_s3_bucket.hrishi: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\S3BucketTerraform> |
```

**Step5: You can look back in the Console if the Bucket is destroyed**



## Creating a docker image using terraform

**Step1:. Verify your docker installation by using**

```
docker -version
```

```
C:\Users\acer>docker --version
Docker version 20.10.17, build 100c701

C:\Users\acer>
```

**Step2: Create a new directory and a create-docker-image.tf file inside.**  
**create-docker-image.tf**

```
terraform {
  required_providers {
    docker = {
      source = "kreuzwerker/docker"
      version = "2.21.0"
    }
  }
}

provider "docker" {
  host = "npipe://///pipe/docker_engine"
}

# Pulls the image
resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name = "foo"
}
```

### Step3: Use terraform commands to create the docker image.

```
terraform init
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "2.15.0"...
- Installing kreuzwerker/docker v2.15.0...
- Installed kreuzwerker/docker v2.15.0 (self-signed, key ID BD080C4571C6104C)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

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.
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> █
```

```
terraform plan
```



```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> terraform plan
```

Terraform used the selected providers to generate the following execution plan. Resource actions  
+ create

Terraform will perform the following actions:

```
# docker_container.foo will be created
+ resource "docker_container" "foo" {
  + attach          = false
  + bridge          = (known after apply)
  + command         = (known after apply)
  + container_logs  = (known after apply)
  + entrypoint      = (known after apply)
  + env            = (known after apply)
  + exit_code       = (known after apply)
  + gateway         = (known after apply)
  + hostname       = (known after apply)
  + id             = (known after apply)
  + image          = (known after apply)
  + init           = (known after apply)
  + ip_address      = (known after apply)
  + ip_prefix_length = (known after apply)
```

```
terraform apply
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> terraform apply
```

Terraform used the selected providers to generate the following execution plan.  
+ create

Terraform will perform the following actions:

```
# docker_container.foo will be created
+ resource "docker_container" "foo" {
  + attach          = false
  + bridge          = (known after apply)
  + command         = (known after apply)
  + container_logs  = (known after apply)
  + entrypoint      = (known after apply)
  + env            = (known after apply)
  + exit code       = (known after apply)
```

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	2dc39ba059dc	4 weeks ago	77.8MB
sonarqube	latest	2cf2f2494695	5 weeks ago	534MB

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> █
```

**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

**docker\_image.ubuntu: Creating...**

**docker\_image.ubuntu: Still creating... [10s elapsed]**

**docker\_image.ubuntu: Still creating... [20s elapsed]**

**docker\_image.ubuntu: Still creating... [30s elapsed]**

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    2dc39ba059dc   4 weeks ago    77.8MB
sonarqube     latest    2cf2f2494695   5 weeks ago    534MB
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> 
```

terraform destroy

```
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:2dc39ba059dcd42ade30aae30147b5692777ba9ff0779a62
```

Terraform used the selected providers to generate the following execution plan  
- destroy

Terraform will perform the following actions:

**# docker\_image.ubuntu will be destroyed**

```
- resource "docker_image" "ubuntu" {
  - id           = "sha256:2dc39ba059dcd42ade30aae30147b5692777ba9ff0779a62"
  - image_id     = "sha256:2dc39ba059dcd42ade30aae30147b5692777ba9ff0779a62"
  - latest       = "sha256:2dc39ba059dcd42ade30aae30147b5692777ba9ff0779a62"
  - name         = "ubuntu:latest" -> null
  - repo_digest  = "ubuntu@sha256:20fa2d7bb4de7723f542be5923b06c4d704370f03"
}
```

**Plan:** 0 to add, 0 to change, 1 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

```
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

docker_image.ubuntu: Destroying... [id=sha256:2dc39ba059dcd42ade30aae30147b5692777ba9ff0779a62ad93a74de02e3e1fubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
sonarqube     latest   2cf2f2494695   5 weeks ago   534MB
PS D:\vesit\sem 5\ADV Dev Ops\Terraform_Scripts\Docker> █
```

## Conclusion:

In this experiment, we used Terraform to create and destroy an AWS EC2 Instance, an Amazon S3 bucket and a Docker image