

School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN, UTTARAKHAND



**System Provisioning and
Configuration Management**

Lab File (2021-2025)
6th Semester

Submitted To:

Dr. Hitesh Kumar

Sharma

Submitted By:

Antriksh Bahri

(500091315)

B Tech CSE

DevOps[6th Semester]

R2142210128

Batch - 1

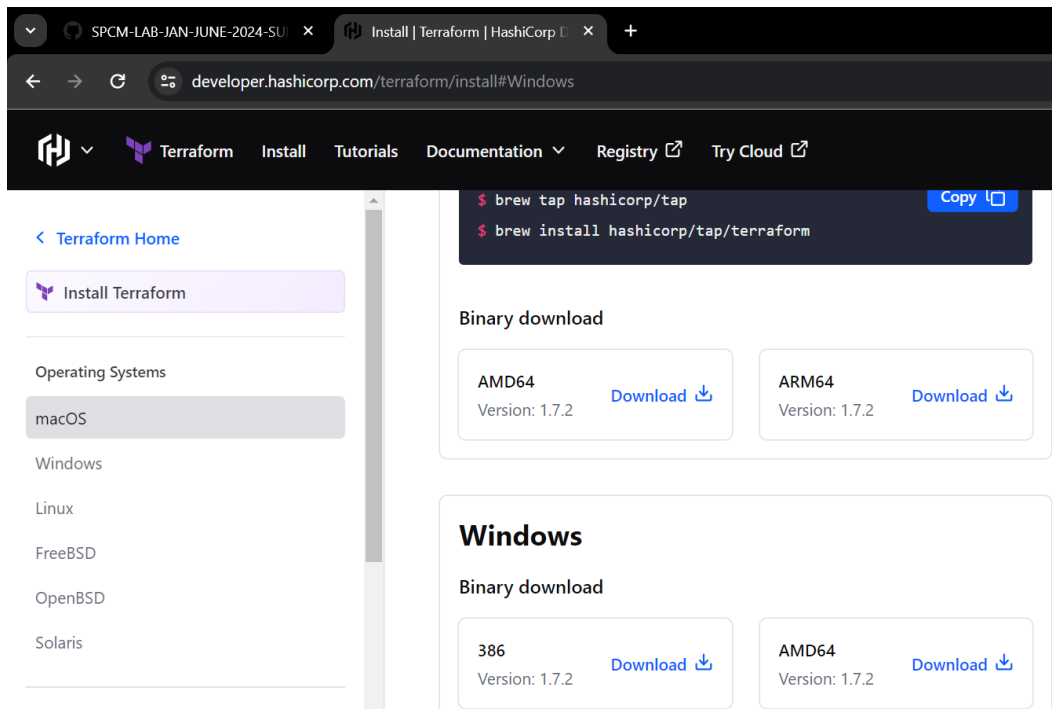
Lab Exercise 1– Install Terraform on Windows

Installing Terraform on Windows requires you to download the correct Terraform package, unpack, and execute it via the CLI. Follow the instructions below to ensure you do not miss any steps.

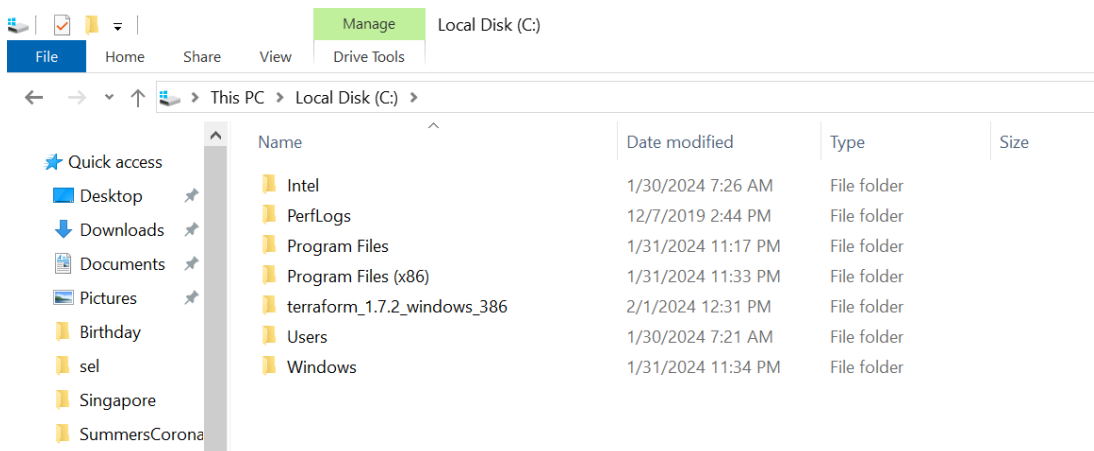
Download Terraform File for Windows

To find the latest version of Terraform for Windows:

1. Browse to the Download Terraform page.
2. Select the Windows tab under the Operating System heading. The latest version is preselected.
3. Choose the binary to download. Select 386 for 32-bit systems or AMD64 for 64-bit systems. Choose the download location for the zip file if the download does not start automatically.

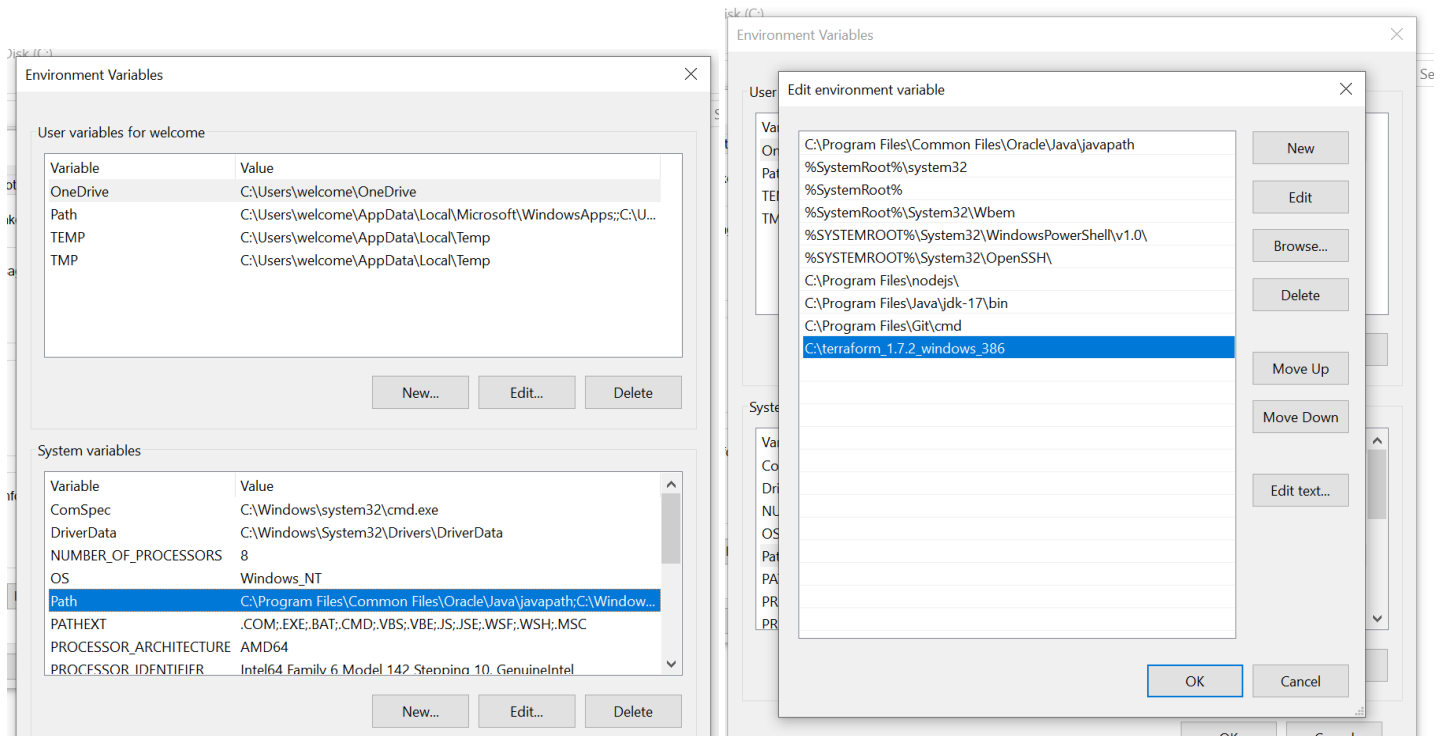


4. Unzip the downloaded file. For example, use the C:\terraform path. Remember this location so you can add the path to the environment variables.



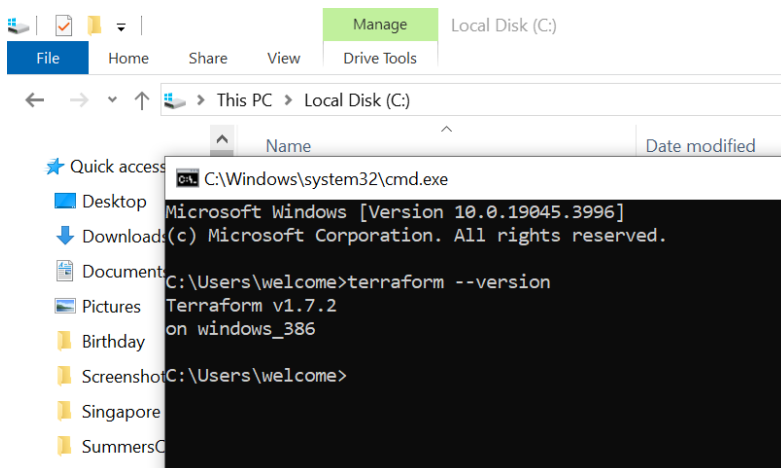
Add Terraform Path to System Environment Variables To add the Terraform executable to the system's global path:

1. Open the start menu, start typing environment and click Edit system environment variables. The System Properties window opens.
2. Click the Environment Variables... button.
3. Select the Path variable in the System variables section to add terraform for all accounts. Alternatively, select Path in the User variables section to add terraform for the currently logged-in user only. Click Edit once you select a Path.
4. Click New in the edit window and enter the location of the Terraform folder.



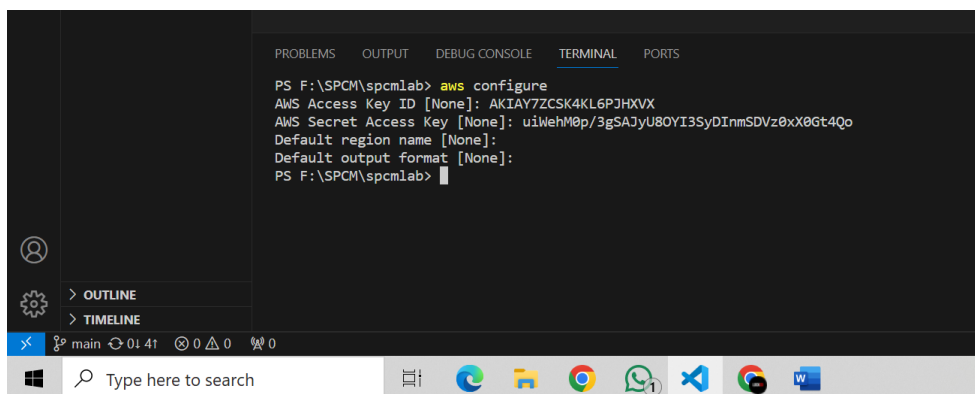
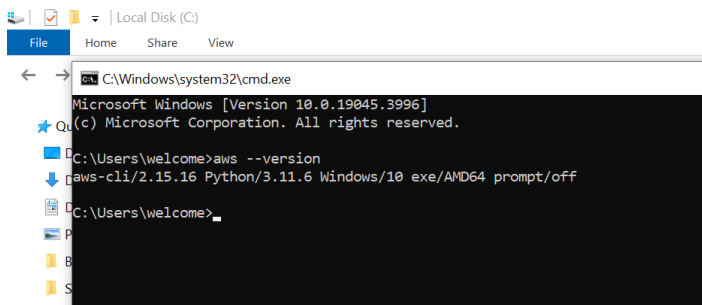
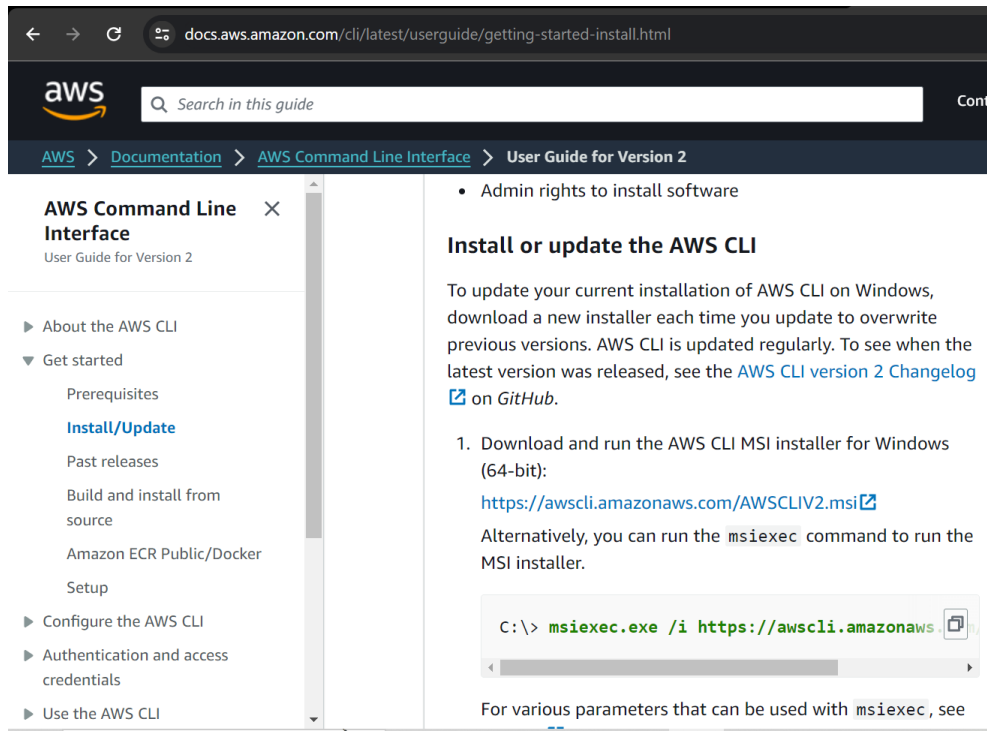
Verify Windows Terraform Installation To check the Terraform global path configuration:

1. Open a new command-prompt window.
2. Enter the command to check the Terraform version: `terraform -version`



Lab Exercise 2– Terraform AWS Provider and IAM User Setting

AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.



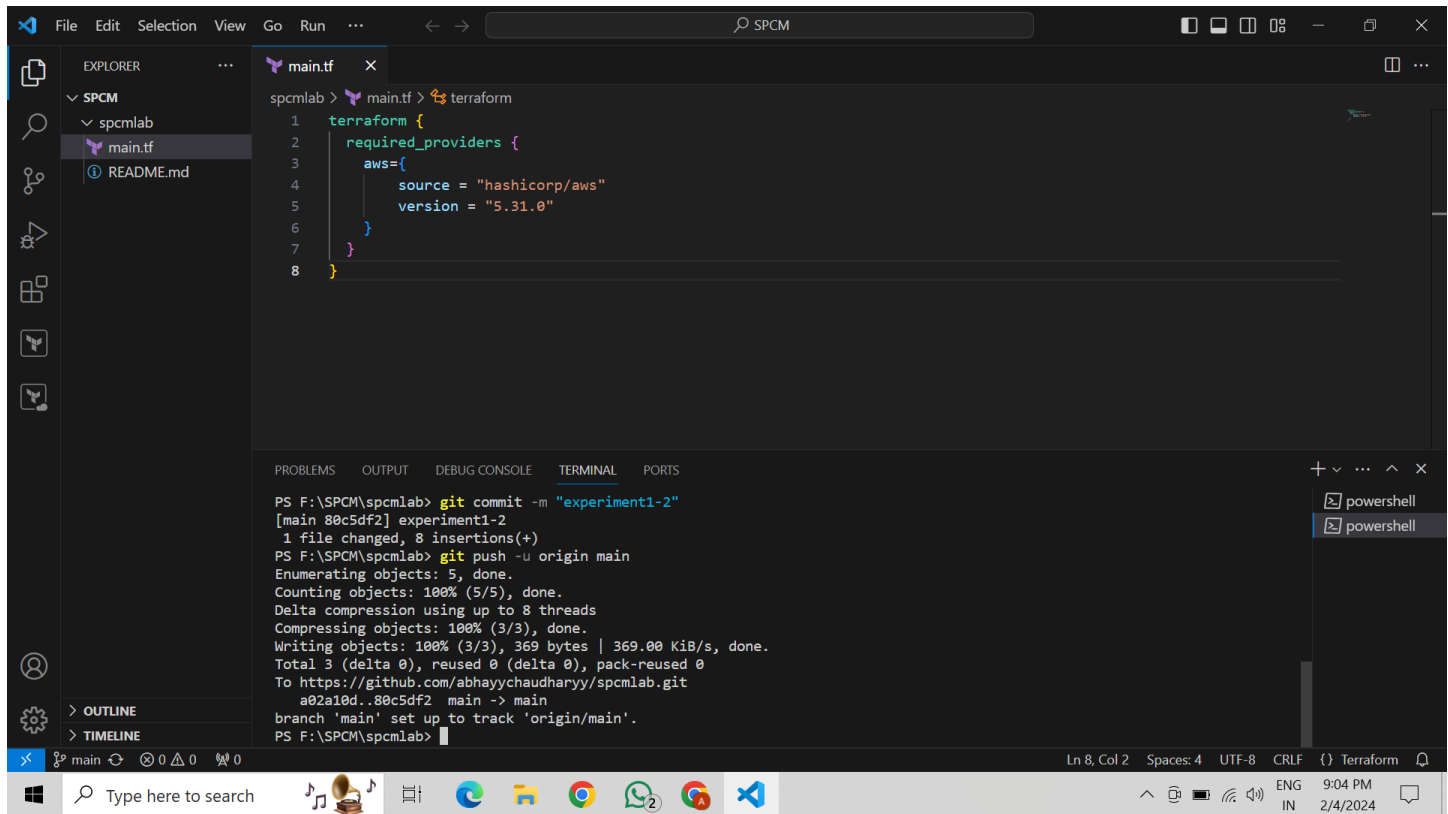
Exercise Steps: Step

1: Create a New Directory: Create a new directory for your Terraform configuration:

2: Create Terraform Configuration File (main.tf): Create a file named main.tf with the following content:

This script defines an AWS provider and provisions an EC2 instance.

3: Initialize Terraform: Run the following command to initialize your Terraform working directory:



```
File Edit Selection View Go Run ... SPCM
```

```
EXPLORER
```

```
SPCM
```

```
main.tf
```

```
main.tf
```

```
README.md
```

```
main.tf
```

```
1 terraform {
```

```
2   required_providers {
```

```
3     aws = {
```

```
4       source = "hashicorp/aws"
```

```
5       version = "5.31.0"
```

```
6     }
```

```
7   }
```

```
8 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS F:\SPCM\spcm\lab> git commit -m "experiment1-2"
```

```
[main 80c5df2] experiment1-2
```

```
1 file changed, 8 insertions(+)
```

```
PS F:\SPCM\spcm\lab> git push -u origin main
```

```
Enumerating objects: 5, done.
```

```
Counting objects: 100% (5/5), done.
```

```
Delta compression using up to 8 threads
```

```
Compressing objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 369 bytes | 369.00 KiB/s, done.
```

```
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

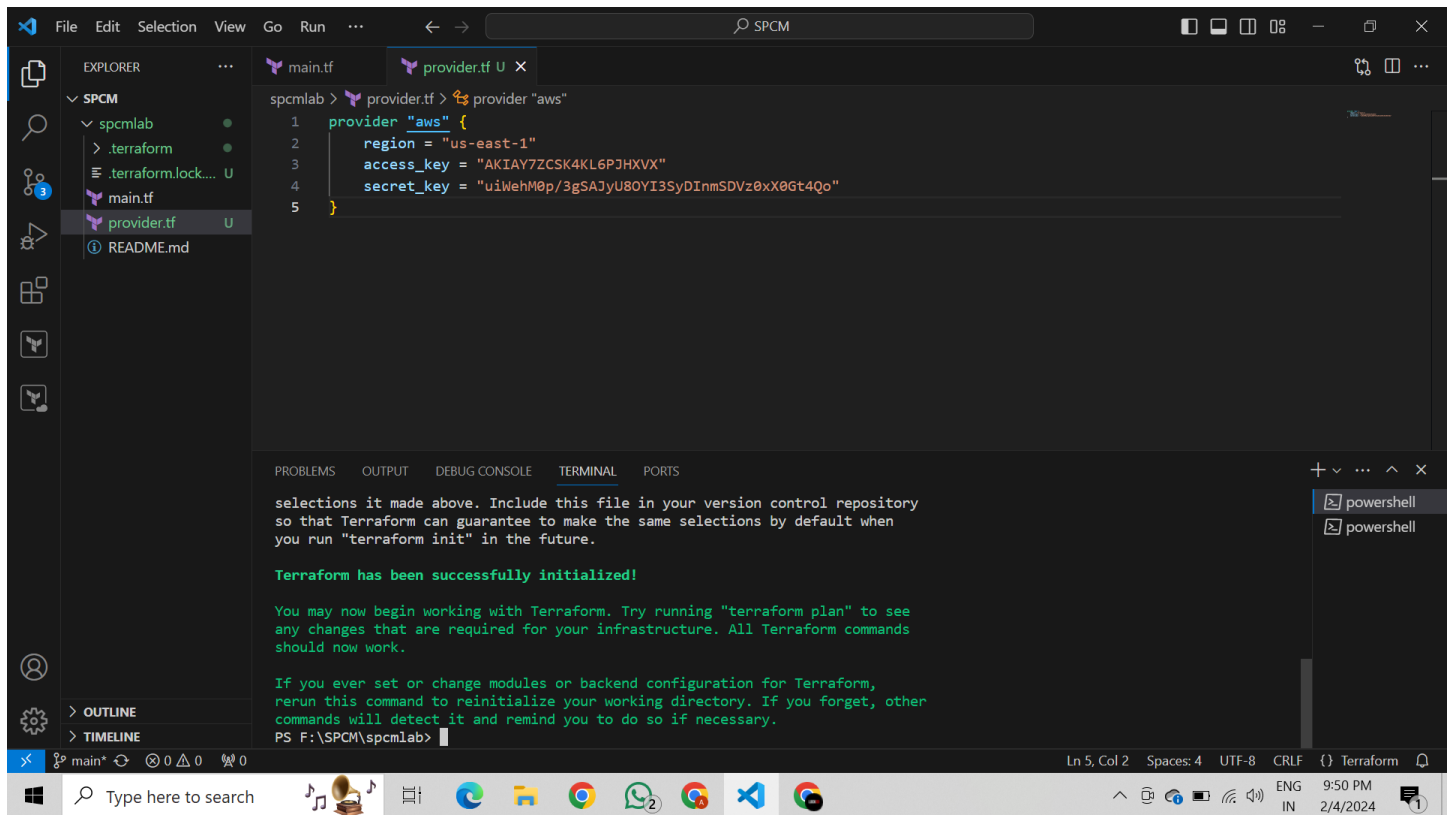
```
To https://github.com/abhayyachaudhary/spcm.git
```

```
a02a10d..80c5df2 main -> main
```

```
branch 'main' set up to track 'origin/main'.
```

```
PS F:\SPCM\spcm\lab>
```

```
Ln 8, Col 2 Spaces: 4 UTF-8 CRLF {} Terraform
```



```
File Edit Selection View Go Run ... SPCM
```

```
EXPLORER
```

```
SPCM
```

```
.terraform
```

```
.terraform.lock.hcl
```

```
main.tf
```

```
provider.tf
```

```
README.md
```

```
provider.tf
```

```
1 provider "aws" {
```

```
2   region = "us-east-1"
```

```
3   access_key = "AKIAIY7ZCSK4KL6PJHXVX"
```

```
4   secret_key = "uiWehM0p/3gSAJyU8OYI3SyDInmSDVz0xX0Gt4Qo"
```

```
5 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
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 F:\SPCM\spcm\lab>
```

```
Ln 5, Col 2 Spaces: 4 UTF-8 CRLF {} Terraform
```

Lab Exercise 3—Provisioning an EC2 Instance on AWS

Exercise Steps:

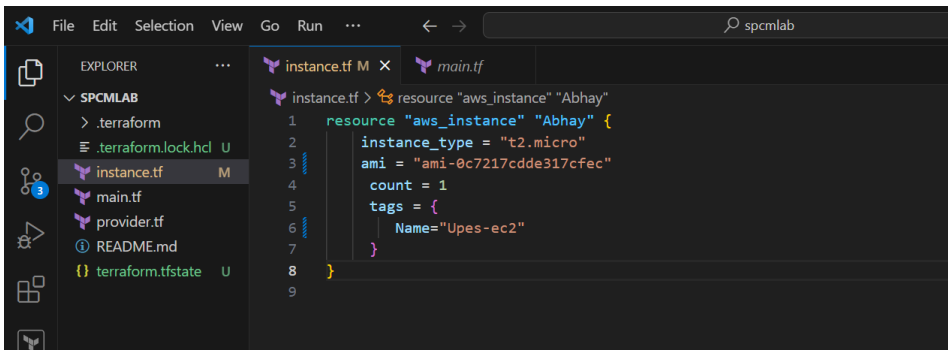
1: Create a New Directory: Create a new directory for your Terraform configuration:

2: Create Terraform Configuration File (main.tf): Create a file named main.tf with the following content:

This script defines an AWS provider and provisions an EC2 instance.

3: Initialize Terraform: Run the following command to initialize your Terraform working directory:

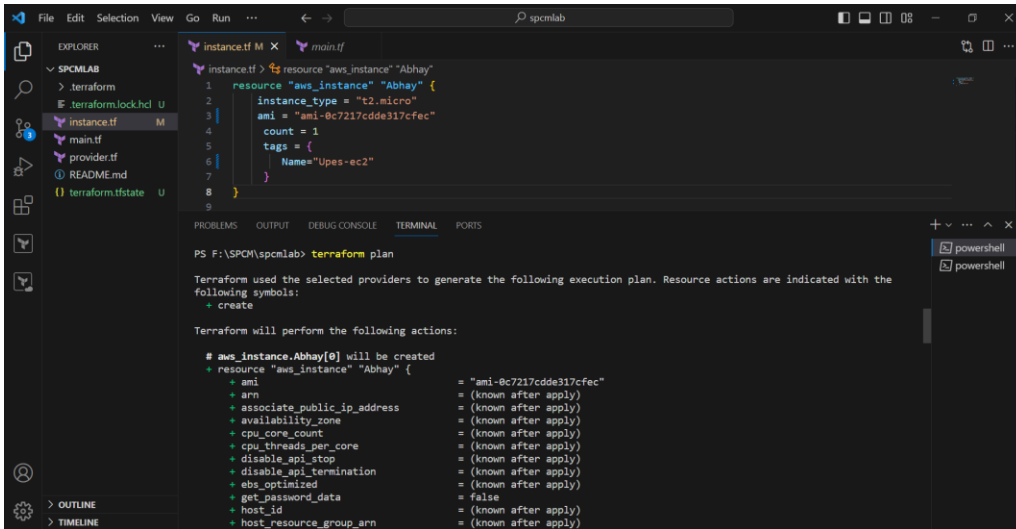
4: Create Terraform Configuration File for EC2 instance (instance.tf): Create a file named instnace.tf with the following content:



```
1 resource "aws_instance" "Abhay" {
2   instance_type = "t2.micro"
3   ami = "ami-0c7217cddde317cfec"
4   count = 1
5   tags = {
6     Name = "Upes-ec2"
7   }
8 }
```

Step 5: Review Plan: Run the following command to see what Terraform will do:

Review the plan to ensure it aligns with your expectations.

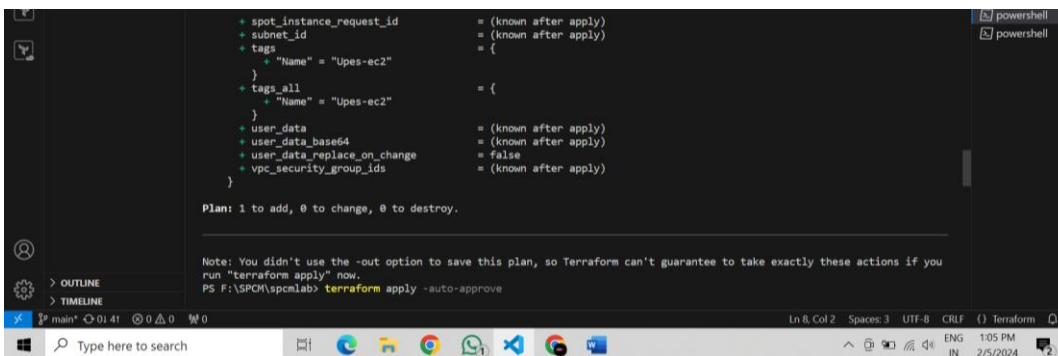


```
PS F:\SPCM\spcm\lab> terraform plan

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.Abhay[0] will be created
+ resource "aws_instance" "Abhay" {
  + ami                    = "ami-0c7217cddde317cfec"
  + 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)
```



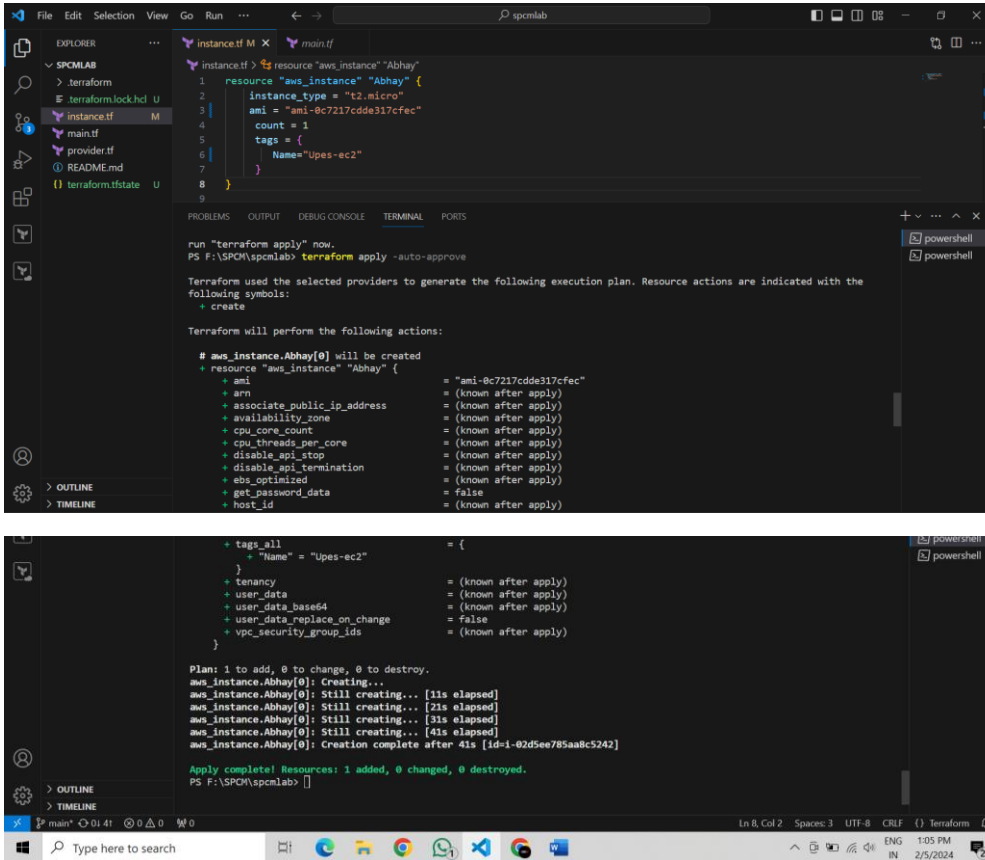
```
+ spot_instance_request_id = (known after apply)
+ subnet_id                = (known after apply)
+ tags                     = {
  + "Name" = "Upes-ec2"
}
+ tags_all                 = {
  + "Name" = "Upes-ec2"
}
+ user_data                = (known after apply)
+ user_data_base64         = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids   = (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

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 apply" now.
PS F:\SPCM\spcm\lab> terraform apply -auto-approve
```

Step 6: Apply Changes: Apply the changes to create the AWS resources:

Type yes when prompted.



The screenshot shows a Visual Studio Code editor with a Terraform configuration file named `instance.tf` open. The configuration defines an AWS EC2 instance named "Abhay" with the following details:

- Resource: `aws_instance` "Abhay"
- Instance type: `t2.micro`
- AMI: `ami-0c7217cde317cfec`
- Count: `1`
- Tags: `{ Name = "Upes-ec2" }`

The terminal window shows the output of the `terraform apply` command. It indicates that the instance will be created and lists the actions Terraform will perform. The output shows the instance is being created and the process is complete.

```
run "terraform apply" now.
PS F:\SPOH\spcnlab> terraform apply -auto-approve

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

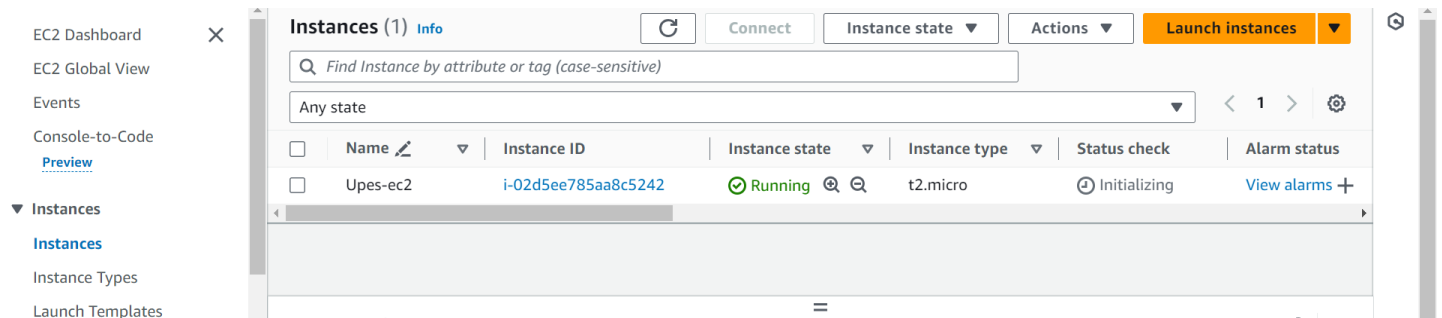
Terraform will perform the following actions:

# aws_instance.Abhay[0] will be created
+ resource "aws_instance" "Abhay" {
  + ami                    = "ami-0c7217cde317cfec"
  + ami                   = (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)
  + instance_type          = "t2.micro"
  + tags_all               = {
    + "Name" = "Upes-ec2"
  }
  + tenancy              = (known after apply)
  + user_data             = (known after apply)
  + user_data_base64     = (known after apply)
  + user_data_replace_on_change = false
  + vpc_security_group_ids = (known after apply)
}

Plans: 1 to add, 0 to change, 0 to destroy.
aws_instance.Abhay[0]: Creating... [1s elapsed]
aws_instance.Abhay[0]: Still creating... [2s elapsed]
aws_instance.Abhay[0]: Still creating... [3s elapsed]
aws_instance.Abhay[0]: Still creating... [4s elapsed]
aws_instance.Abhay[0]: creation complete after 41s [id=i-02d5ee785aa8c5242]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS F:\SPOH\spcnlab>
```

Step 7: Verify Resources: After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created.



The screenshot shows the AWS Management Console EC2 Instances page. The table displays the following instance:

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	Upes-ec2	i-02d5ee785aa8c5242	Running	t2.micro	Initializing	View alarms

Step 8: Cleanup Resources: When you are done experimenting, run the following command to destroy the created resources:

File Edit Selection View Go Run ...

SPCMLAB

- > terraform
- > .terraform.lock.hcl U
- instance.tf M
- main.tf
- provider.tf
- README.md
- terraform.tfstate U
- terraform.tfstate... U

```
instance.tf X main.tf
1 resource "aws_instance" "Abhay"
2   resource "aws_instance" "Abhay" {
3     instance_type = "t2.micro"
4     ami = "ami-0c7217cdde317cFec"
5     count = 1
6     tags = {
7       Name="Upes-ec2"
8     }
9   }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
aws_instance.Abhay[0]: Still creating... [21s elapsed]
aws_instance.Abhay[0]: Still creating... [31s elapsed]
aws_instance.Abhay[0]: Still creating... [41s elapsed]
aws_instance.Abhay[0]: Creation complete after 41s [id=i-02d5ee785aa8c5242]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS F:\SPCM\spcmlab> terraform destroy -auto-approve
aws_instance.Abhay[0]: Refreshing state... [id=i-02d5ee785aa8c5242]

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:
```

Plan: 0 to add, 0 to change, 1 to destroy.

```
aws_instance.Abhay[0]: Destroying... [id=i-02d5ee785aa8c5242]
aws_instance.Abhay[0]: Still destroying... [id=i-02d5ee785aa8c5242, 10s elapsed]
aws_instance.Abhay[0]: Still destroying... [id=i-02d5ee785aa8c5242, 20s elapsed]
aws_instance.Abhay[0]: Still destroying... [id=i-02d5ee785aa8c5242, 30s elapsed]
aws_instance.Abhay[0]: Destruction complete after 33s

Destroy complete! Resources: 1 destroyed.
PS F:\SPCM\spcmlab>
```

Ln 8, Col 2 Spaces: 3 UTF-8 CRLF {} Terraform

Type here to search

EC2 Dashboard

EC2 Global View

Events

Console-to-Code

Preview

Instances

Instances

Instances (1) Info

Find Instance by attribute or tag (case-sensitive)

Any state

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	Upes-ec2	i-02d5ee785aa8c5242	Terminated	t2.micro	-	View alarms

Lab Exercise 4– Terraform Variables

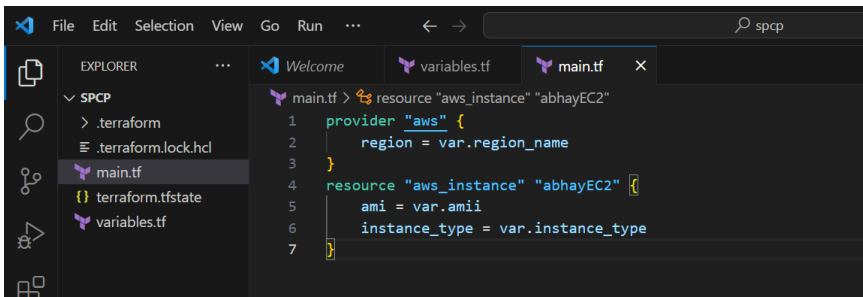
Objective: Learn how to define and use variables in Terraform configuration.

Steps: 1. Create a Terraform Directory:

- Create a new directory for your Terraform project.

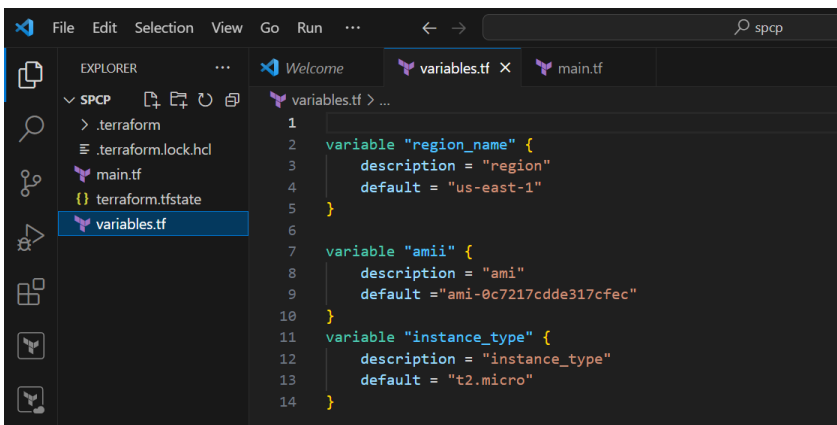
2. Create a Terraform Configuration File:

- Create a file named main.tf within your project directory. # main.tf



```
File Edit Selection View Go Run ... sscp
EXPLORER
  SPCP
    .terraform
    .terraform.lock.hcl
    main.tf
    terraform.tfstate
    variables.tf
main.tf > resource "aws_instance" "abhayEC2"
1 provider "aws" {
2   region = var.region_name
3 }
4 resource "aws_instance" "abhayEC2" {
5   ami = var.ami
6   instance_type = var.instance_type
7 }
```

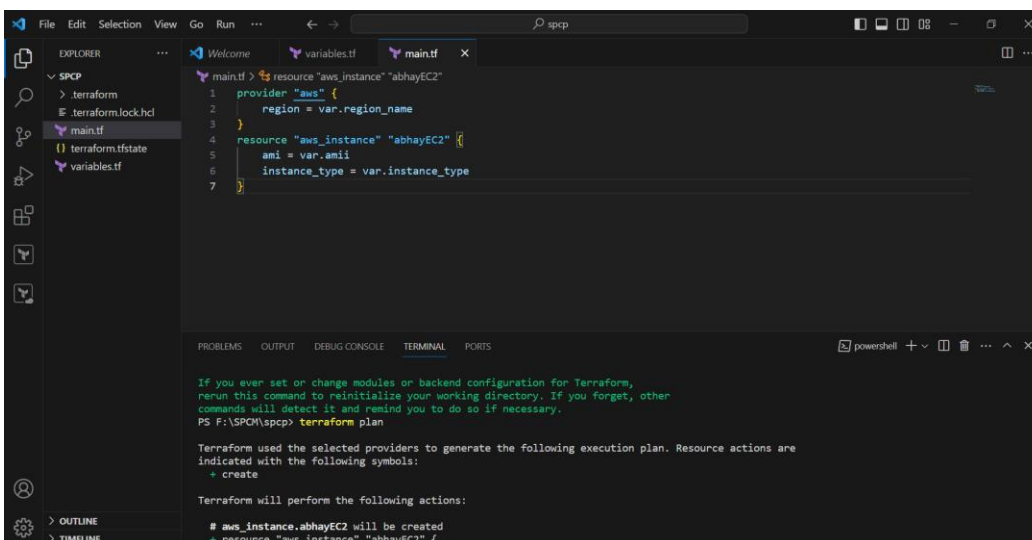
3. Define Variables: • Open a new file named variables.tf. Define variables for region, ami, and instance_type. # variables.tf



```
File Edit Selection View Go Run ... sscp
EXPLORER
  SPCP
    .terraform
    .terraform.lock.hcl
    main.tf
    terraform.tfstate
    variables.tf
variables.tf > ...
1
2 variable "region_name" {
3   description = "region"
4   default = "us-east-1"
5 }
6
7 variable "ami" {
8   description = "ami"
9   default = "ami-0c7217cdde317cfec"
10 }
11 variable "instance_type" {
12   description = "instance_type"
13   default = "t2.micro"
14 }
```

4. Use Variables in main.tf: • Modify main.tf to use the variables. # main.tf

5. Initialize and Apply: • Run the following Terraform commands to initialize and apply the configuration.



```
File Edit Selection View Go Run ... sscp
EXPLORER
  SPCP
    .terraform
    .terraform.lock.hcl
    main.tf
    terraform.tfstate
    variables.tf
main.tf > resource "aws_instance" "abhayEC2"
1 provider "aws" {
2   region = var.region_name
3 }
4 resource "aws_instance" "abhayEC2" {
5   ami = var.ami
6   instance_type = var.instance_type
7 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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 F:\SPCH\spcp> terraform plan

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

Terraform will perform the following actions:

# aws_instance.abhayEC2 will be created
+ resource "aws_instance" "abhayEC2" {
```

```
+ user_data_base64      = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.
aws_instance.abhayEC2: Creating...
aws_instance.abhayEC2: Still creating... [10s elapsed]
aws_instance.abhayEC2: Still creating... [20s elapsed]
aws_instance.abhayEC2: Still creating... [30s elapsed]
aws_instance.abhayEC2: Creation complete after 39s [id=i-0118fe49083c1586f]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS F:\SPCH\spcp>
```

EC2 Dashboard

EC2 Global View

Events

Console-to-Code

Instances

Instance Types

Launch Templates

Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

Any state

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	Upes-ec2	i-02d5ee785aa8c5242	Terminated	t2.micro	-	View alarms
<input type="checkbox"/>		i-0118fe49083c1586f	Running	t2.micro	Initializing	View alarms

Select an instance

Observe how the region changes based on the variable override.

6. Clean Up: After testing, you can clean up resources.

```
main.tf
resource "aws_instance" "abhayEC2" {
  provider = "aws"
  region = var.region_name
  resource "aws_instance" "abhayEC2" {
    ami = var.ami
    instance_type = var.instance_type
  }
}
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS F:\SPCH\spcp> terraform destroy -auto-approve
aws_instance.abhayEC2: Refreshing state... [id=i-0118fe49083c1586f]

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.abhayEC2 will be destroyed
resource "aws_instance" "abhayEC2" {
  ami           = "ami-0c7217cdd337cfeec" -> null
  arm           = "arn:aws:ec2:us-east-1:618010924820:instance/i-0118fe49083c1586f" -> null
}
```

```
- volume_size      = 8 -> null
- volume_type      = "gp2" -> null
}

Plan: 0 to add, 0 to change, 1 to destroy.
aws_instance.abhayEC2: Destroying... [id=i-0118fe49083c1586f]
aws_instance.abhayEC2: Still destroying... [id=i-0118fe49083c1586f, 10s elapsed]
aws_instance.abhayEC2: Still destroying... [id=i-0118fe49083c1586f, 20s elapsed]
aws_instance.abhayEC2: Still destroying... [id=i-0118fe49083c1586f, 30s elapsed]
aws_instance.abhayEC2: Destruction complete after 33s

Destroy complete! Resources: 1 destroyed.
PS F:\SPCH\spcp>
```

EC2 Dashboard

EC2 Global View

Events

Console-to-Code

Instances

Instance Types

Launch Templates

Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

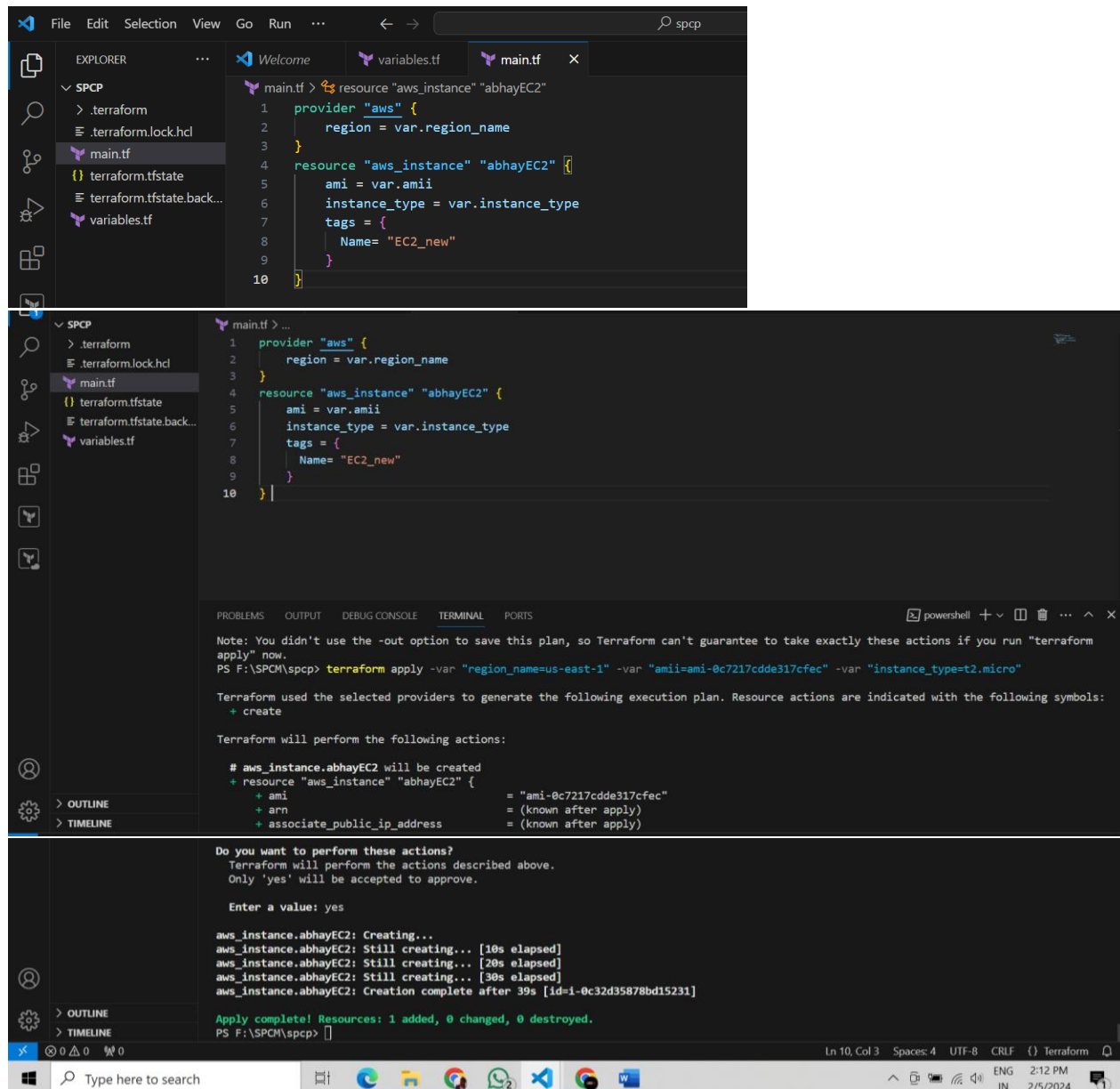
Any state

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	Upes-ec2	i-02d5ee785aa8c5242	Terminated	t2.micro	-	View alarms
<input type="checkbox"/>		i-0118fe49083c1586f	Terminated	t2.micro	-	View alarms

Lab Exercise 5– Terraform Variables with Command Line Arguments

. Use Command Line Arguments: • Open a terminal and navigate to your Terraform project directory. • Run the terraform init command:

Run the terraform apply command with command line arguments to set variable values:



The screenshot shows a Visual Studio Code editor with a Terraform project. The Explorer pane on the left shows the project structure: SPCP, .terraform, .terraform.lock.hcl, main.tf, terraform.tfstate, terraform.tfstate.backup, and variables.tf. The main.tf file is open, showing the following Terraform configuration:

```
1 provider "aws" {
2   region = var.region_name
3 }
4 resource "aws_instance" "abhayEC2" {
5   ami = var.ami
6   instance_type = var.instance_type
7   tags = {
8     Name = "EC2_new"
9   }
10 }
```

The terminal pane at the bottom shows the output of the `terraform apply` command. It displays the execution plan, the actions to be performed, and the progress of the resource creation.

```
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 apply" now.
PS F:\SPCM\spcp> terraform apply -var "region_name=us-east-1" -var "ami=ami-0c7217cdd317cfec" -var "instance_type=t2.micro"

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:

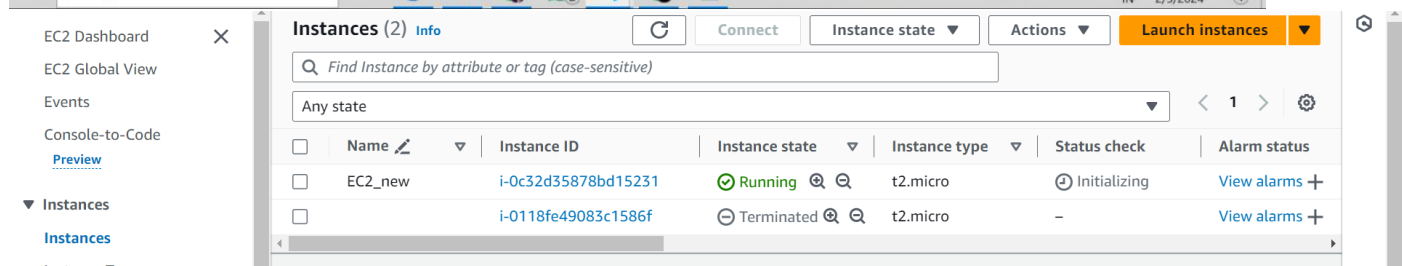
# aws_instance.abhayEC2 will be created
+ resource "aws_instance" "abhayEC2" {
  + ami              = "ami-0c7217cdd317cfec"
  + arn              = (known after apply)
  + associate_public_ip_address = (known after apply)

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.abhayEC2: Creating...
aws_instance.abhayEC2: Still creating... [10s elapsed]
aws_instance.abhayEC2: Still creating... [20s elapsed]
aws_instance.abhayEC2: Still creating... [30s elapsed]
aws_instance.abhayEC2: Creation complete after 39s [id=i-0c32d35878bd15231]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS F:\SPCM\spcp>
```



The screenshot shows the AWS Management Console 'Instances' page. It displays a table of EC2 instances with columns for Name, Instance ID, Instance state, Instance type, Status check, and Alarm status. The table shows two instances: 'EC2_new' (Running) and 'i-0118fe49083c1586f' (Terminated).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
EC2_new	i-0c32d35878bd15231	Running	t2.micro	Initializing	View alarms +
	i-0118fe49083c1586f	Terminated	t2.micro	-	View alarms +

Clean Up: After testing, you can clean up resources:

