

Task 7

Terraform:

- Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp that allows users to define and provision cloud and on-premises resources using a declarative configuration language.
 - It enables users to manage infrastructure throughout its lifecycle by writing human-readable configuration files that can be versioned, reused, and shared.

Terraform Command:

Creation of EC2

1) terraform init:

2) terraform plan:

The screenshot shows a terminal window with the following content:

```
PS C:\Users\SRINATH\OneDrive\Desktop\AWS & DEVOPS\Devops\Terraform modules> 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.srinath-server will be created
+ resource "aws_instance" "srinath-server" {
    + ami                      = "ami-03978d951b279ec0b"
    + arn                      = (known after apply)
    + associate_public_ip_address = (known after apply)
    + availability_zone        = (known after apply)
    + disable_api_stop          = (known after apply)
    + disable_api_termination   = (known after apply)
    + ebs_optimized             = (known after apply)
    + enable_primary_ipv6       = (known after apply)
    + force_destroy              = false
    + get_password_data         = false
    + host_id                   = (known after apply)
    + host_resource_group_arn   = (known after apply)
    + iam_instance_profile      = (known after apply)
    + id                        = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle        = (known after apply)
    + instance_state             = (known after apply)
    + instance_type              = "t3.micro"
    + ipv6_address_count        = (known after apply)
    + ipv6_addresses            = (known after apply)
```

3) terraform apply

The screenshot shows a Visual Studio Code interface with two tabs open: 'main.tf' and 'provider.tf'. The 'main.tf' tab contains the following Terraform code:

```
resource "aws_instance" "srinath-server" {
  ami           = "ami-03978d951b279ec0b"
  instance_type = "t3.micro"

  tags = {
    Name = "web-server"
  }
}
```

The 'TERMINAL' tab shows the command `terraform apply` being run in a PowerShell window. The terminal output indicates that Terraform will create a new AWS instance named 'srinath-server'.

```
PS C:\Users\SRINATH\OneDrive\Desktop\AWS & DEVOPS\Devops\Terraform modules> 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.srinath-server will be created
+ resource "aws_instance" "srinath-server" {
  + ami           = "ami-03978d951b279ec0b"
  + arn           = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone        = (known after apply)
  + disable_api_stop          = (known after apply)
  + disable_api_termination   = (known after apply)
  + ebs_optimized             = (known after apply)
  + enable_primary_ipv6       = (known after apply)
  + force_destroy             = false
  + get_password_data         = false
  + host_id                  = (known after apply)
  + host_resource_group_arn   = (known after apply)
  + iam_instance_profile      = (known after apply)
  + id                       = (known after apply)
  + instance_initiated_shutdown_behavior = (known after apply)
  + instance_lifecycle        = (known after apply)
  + instance_state             = (known after apply)
```

Output

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with 'EC2' selected. Under 'Instances', it lists 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', and 'Capacity Manager'. Below that is 'Images' with 'AMIs' and 'AMI Catalog'. The main area is titled 'Instances (1) Info' with a search bar and filter buttons for 'Instance state = running' and 'Clear filters'. A table displays one instance: 'web-server' (Instance ID: i-048811b0ee167bec3), which is 'Running' (Status check: Initializing). It's an 't3.micro' instance type, located in 'us-west-1a' availability zone, with a public IP of 'ec2-54-1-'. At the bottom, there are buttons for 'Connect', 'Actions', and 'Launch instances'.

Creation of IAM:

1) terraform plan:

The screenshot shows the Visual Studio Code interface with a Terraform workspace. The left sidebar has an 'EXPLORER' view showing files like 'provider.tf', 'iam User.tf', 'output.tf.bak', etc. The main editor pane shows 'iam User.tf' with the following code:

```
provider "aws" {
  region = "cloud"
}

resource "aws_iam_user" "cloud" {
  name = "cloud-user"
}

resource "aws_iam_user" "cloud-count" {
  count = 5
  name  = "cloud-user-${count.index + 1}"
}

resource "aws_iam_user" "cloud-forset-user" {
  for_each = toset(["arun", "ram", "varun"])
  name    = each.value
}
```

Below the code, the terminal window shows the command 'terraform plan' being run in a PowerShell session. The output indicates that Terraform will create three AWS IAM users: 'cloud-user', 'cloud-user-1', and 'cloud-user-2'. The terminal also shows the execution plan symbols: '+' for create and '-' for destroy.

2) terraform apply:

```

provider.tf
iam User.tf
provider.hcl

TERRAFORM MODULES
> terraform
> Terraform-files-of-AWS
  .terraform.lock.hcl
  deploy.sh
  E2 instance.tf.bak
  iam User.tf
  output.tf.bak
  provider.tf
  s3 Bucket.tf.bak
  shellsh
  terraform.tfstate
  terraform.tfstate.backup
  terraform.tvars.bak
  variables.tf.bak
  vpc.tf.bak

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
PS C:\Users\SRINATH\Desktop\AWS & DEVOPS\Devops\Terraform modules> 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_iam_user.cloud will be created
+ resource "aws_iam_user" "cloud" {
    + arn          = (known after apply)
    + force_destroy = false
    + id           = (known after apply)
    + name         = "cloud-user"
    + path          = "/"
    + tags_all     = (known after apply)
    + unique_id    = (known after apply)
}

```

Output:

User name	Path	Group	Last activity	MFA	Password age	Console last sign-in
arun	/	0	-	-	-	-
cloud-user	/	0	-	-	-	-
cloud-user-1	/	0	-	-	-	-
cloud-user-2	/	0	-	-	-	-
cloud-user-3	/	0	-	-	-	-
cloud-user-4	/	0	-	-	-	-
cloud-user-5	/	0	-	-	-	-
Gowtham	/	0	2 days ago	-	3 days	2 days ago
ram	/	0	-	-	-	-

3) terraform destroy:

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure with files like `provider.tf`, `iam User.tf`, `provider.sh`, and `variables.tf`.
- Code Editor:** Displays the `iam User.tf` file containing Terraform code to create IAM users.
- Terminal:** Shows the command `terraform destroy` being run in a PowerShell terminal.
- Output:** Displays the execution plan and actions Terraform will perform.

```
PS C:\Users\SRINATH\Desktop\AWS & DEVOPS\devops\Terraform modules> terraform destroy
aws_iam_user.cloud-count[4]: Refreshing state... [id=cloud-user-5]
aws_iam_user.cloud-count[2]: Refreshing state... [id=cloud-user-3]
aws_iam_user.cloud-forset-user["arun"]: Refreshing state... [id=arun]
aws_iam_user.cloud-count[1]: Refreshing state... [id=cloud-user-2]
aws_iam_user.cloud: Refreshing state... [id=cloud-user]
aws_iam_user.cloud-forset-user["varun"]: Refreshing state... [id=varun]
aws_iam_user.cloud-forset-user["ram"]: Refreshing state... [id=ram]
aws_iam_user.cloud-count[3]: Refreshing state... [id=cloud-user-4]
aws_iam_user.cloud-count[0]: Refreshing state... [id=cloud-user-1]

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:
```