TERRAFORM – POC

Installing aws cli and configuring

AWS configure

Providing credentials region and output types

1. **Create Two Resources in Different Regions**

Here's how you can create two resources, such as EC2 instances, in different regions

Provider.tf

```
main.tf    provider.tf ×    .terraform.lock.hcl
provider.tf > provider "aws" > alias
# Define the AWS provider for the US East region
provider "aws" {
  alias  = "us_east"
  region = "us-east-1"
}

# Define the AWS provider for the US West region
provider "aws" {
  alias  = "us_west"
  region = "us-west-2"
}
```

Main.tf

```
main.tf ×    provider.tf    .terraform.lock.hcl
main.tf > resource "aws_instance" "instance_west" > tags
# Resource in US East region
resource "aws_instance" "instance_east" {
  provider = aws.us_east
  ami      = "ami-0ba9883b710b05ac6" # Example AMI ID
  instance_type = "t2.micro"
  tags = {
    Name = "Instance-EAST"
  }
}

# Resource in US West region
resource "aws_instance" "instance_west" {
  provider = aws.us_west
  ami      = "ami-0440fa9465661a496" # Example AMI ID
  instance_type = "t2.micro"
  tags = {
    Name = "Instance-WEST"
```

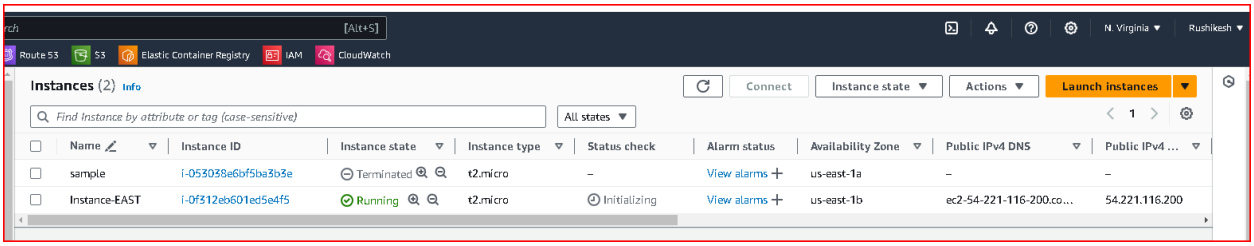
```
PS E:\Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0

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.
```

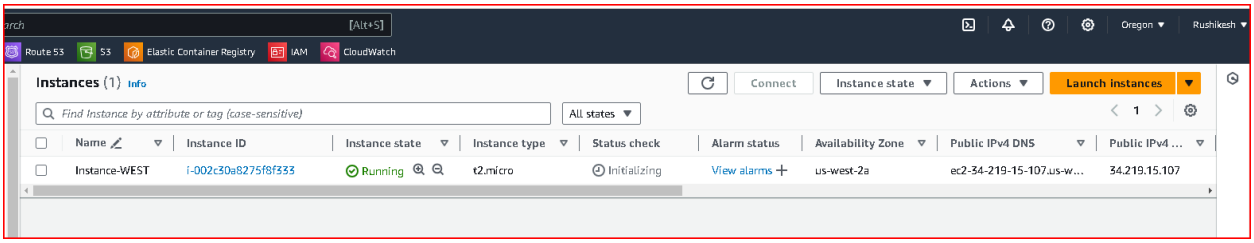
```
PS E:\Terraform> terraform plan

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

Terraform will perform the following actions:

  # aws_instance.instance_east will be created
  + resource "aws_instance" "instance_east" {
      + ami                                  = "ami-0ba9883b710b05ac6"
      + 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)
      + 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                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
```

```
PS E:\Terraform> terraform apply

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

Terraform will perform the following actions:

  # aws_instance.instance_east will be created
  + resource "aws_instance" "instance_east" {
      + ami                                  = "ami-0ba9883b710b05ac6"
      + 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)
      + 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                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
```

```
    }

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.instance_east: Creating...
aws_instance.instance_west: Creating...
aws_instance.instance_east: Still creating... [10s elapsed]
aws_instance.instance_west: Still creating... [10s elapsed]
aws_instance.instance_east: Still creating... [20s elapsed]
aws_instance.instance_west: Still creating... [20s elapsed]
aws_instance.instance_east: Still creating... [30s elapsed]
aws_instance.instance_west: Still creating... [30s elapsed]
aws_instance.instance_east: Creation complete after 36s [id=i-0f312eb601ed5e4f5]
aws_instance.instance_west: Creation complete after 37s [id=i-002c30a8275f8f333]

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

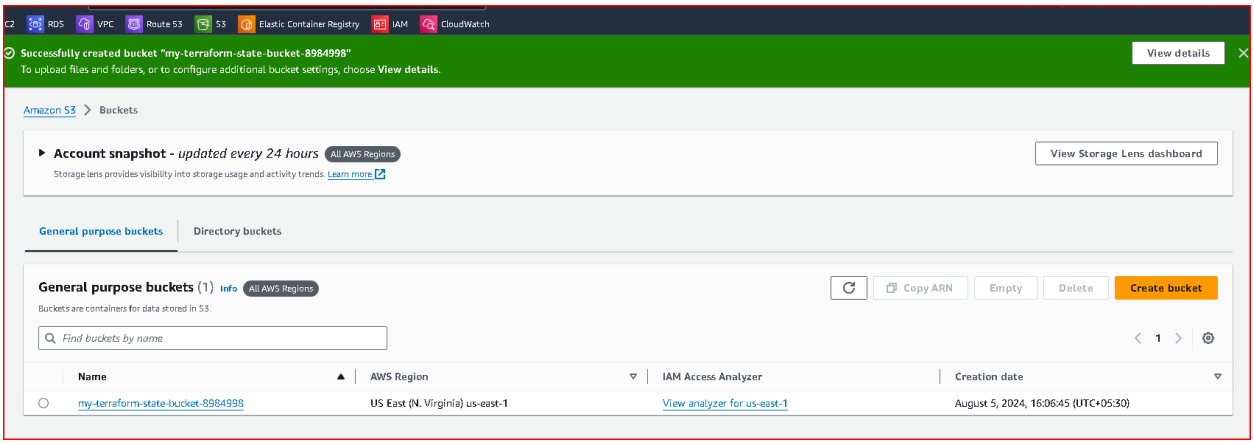
Successfully created resources within two different regions



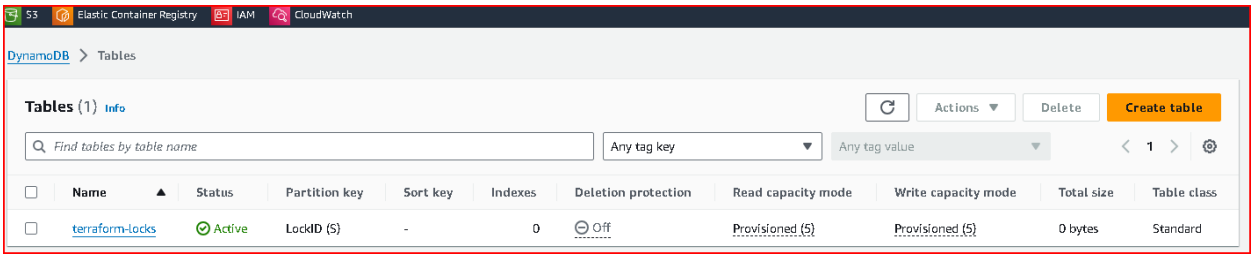


2. Perform state lock

Creating S3 bucket



Creating dynamoDB table



Initializing backend to store state

```
PS E:\Terraform> terraform init
Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "s3" backend. No existing state was found in the newly
  configured "s3" backend. Do you want to copy this state to the new "s3"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes

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

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0

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
```

main.tf .\ | main.tf ec2 | provider.tf ec2 | backend.tf ec2 1 | backend.tf .\ | provider.tf .\ | .terraform.lock.hcl

backend.tf > terraform

```
terraform {
  backend "s3" {
    bucket         = "my-terraform-state-bucket-8984998"  # Replace with your bucket name
    key            = "terraform/state.tfstate"    # Path within the bucket
    region         = "us-east-1"                  # AWS region
    encrypt        = true                         # Enable server-side encryption
    dynamodb_table = "terraform-locks"            # Replace with your DynamoDB table name
  }
}
```

PROBLEMS 1 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | CODE REFERENCE LOG

```
PS E:\Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0

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.
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,
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.
```

```
should now work.
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,
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 E:\Terraform> terraform apply
Acquiring state lock. This may take a few moments...
aws_instance.instance_east: Refreshing state... [id=i-0f312eb601ed5e4f5]
aws_instance.instance_west: Refreshing state... [id=i-002c30a8275f8f333]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differe
Releasing state lock. This may take a few moments...

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
PS E:\Terraform> terraform apply
Acquiring state lock. This may take a few moments...
aws_instance.instance_east: Refreshing state... [id=i-0f312eb601ed5e4f5]
aws_instance.instance_west: Refreshing state... [id=i-002c30a8275f8f333]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differe
Releasing state lock. This may take a few moments...

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

created new terraform project state lock with similar codes and Backend to store statefile

main.tf .\ | main.tf ec2 | provider.tf ec2 | backend.tf ec2 1 | backend.tf .\ | provider.tf .\ | .terraform.lock.hcl

ec2 > backend.tf > terraform > backend "s3"

```
terraform {
  backend "s3" {
    key            = "terraform/state.tfstate"    # Path within the bucket
    region         = "us-east-1"                   # AWS region
    encrypt        = true                         # Enable server-side encryption
    dynamodb_table = "terraform-locks"             # Replace with your DynamoDB table name
  }
```

```
PS E:\Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0

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.
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,
```

```
PS E:\Terraform> terraform apply
Acquiring state lock. This may take a few moments...

  Error: Error acquiring the state lock

  Error message: operation error DynamoDB: PutItem, https response error StatusCode: 400, RequestID: FKIDFS6LPSLKE5MJ5MD1N841IBVV4KQNSO5AEMVJF66Q9ASUAAJG,
  ConditionalCheckFailedException: The conditional request failed
  Lock Info:
    ID:        c2dc68f6-b32a-fd87-dbe4-315540be45df
    Path:      my-terraform-state-bucket-8984998/terraform/state.tfstate
    Operation: OperationTypeApply
    Who:       CYBAGE\rushikeshja@GVC1502
    Version:   1.9.3
    Created:   2024-08-05 11:05:55.1949527 +0000 UTC
    Info:


  Terraform acquires a state lock to protect the state from being written
  by multiple users at the same time. Please resolve the issue above and try
  again. For most commands, you can disable locking with the "-lock=false"
  flag, but this is not recommended.

PS E:\Terraform>
```

3. Create 3 identical resources with different name using loops.

provider.tf: Configures the AWS provider with the desired region i.e “us-east-1”

```
Welcome | main.tf | outputs.tf | provider.tf ●
provider.tf > ...
1  # provider.tf
2  provider "aws" {
3    region = "us-east-1"
4  }
5
```

main.tf:

a. variable "instance_count": Defines the number of instances to create.
b. variable "instance_names": Defines the names for each instance.
c. resource "aws_instance" "example": Uses the count meta-argument to create multiple instances. The count.index is used to index into the instance_names variable to assign unique names.

Welcome | main.tf | outputs.tf | provider.tf

main.tf > resource "aws_instance" "example" > ami

```
# main.tf
variable "instance_count" {
  default = 3
}

variable "instance_names" {
  default = ["instance-1", "instance-2", "instance-3"]
}

resource "aws_instance" "example" {
  count         = var.instance_count
  ami           = "ami-0ad21ae1d0696ad58"
  instance_type = "t2.micro"
  tags = {
    Name = var.instance_names[count.index]
  }
}
```

outputs.tf:

output "instance_ids": Outputs the IDs of the created instances.

output "instance_public_ips": Outputs the public IP addresses of the created instances.

```
Welcome   main.tf   outputs.tf ×   provider.tf ●
outputs.tf > ...
# outputs.tf

# Output the IDs of the instances
output "instance_ids" {
  description = "The IDs of the instances"
  value       = [for instance in aws_instance.example : instance.id]
}

# Output the public IP addresses of the instances
output "instance_public_ips" {
  description = "The public IP addresses of the instances"
  value       = [for instance in aws_instance.example : instance.public_ip]
}
```

successfully executing following commands

Terraform init-

```
PS C:\Users\karanpa\Documents\Terraform\terraform-loops> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.59.0...
- Installed hashicorp/aws v5.59.0 (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
```

Terraform plan

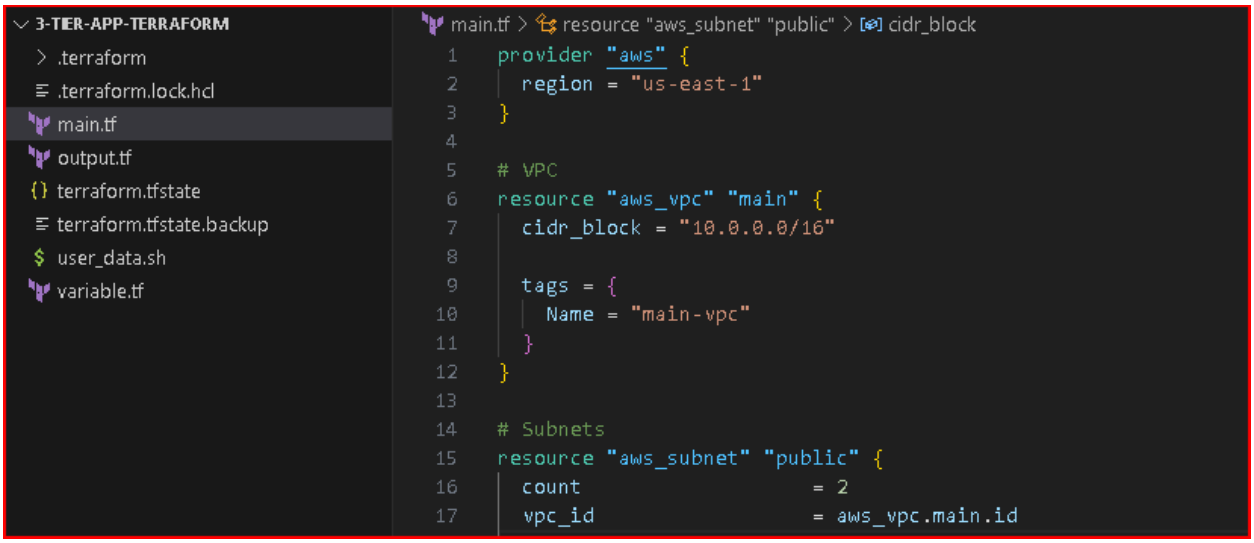
```
PS C:\Users\karanpa\Documents\Terraform\terraform-loops> terraform plan

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.example[0] will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0ad21ae1d0696ad58"
      + 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)
```

Terraform apply with outputs

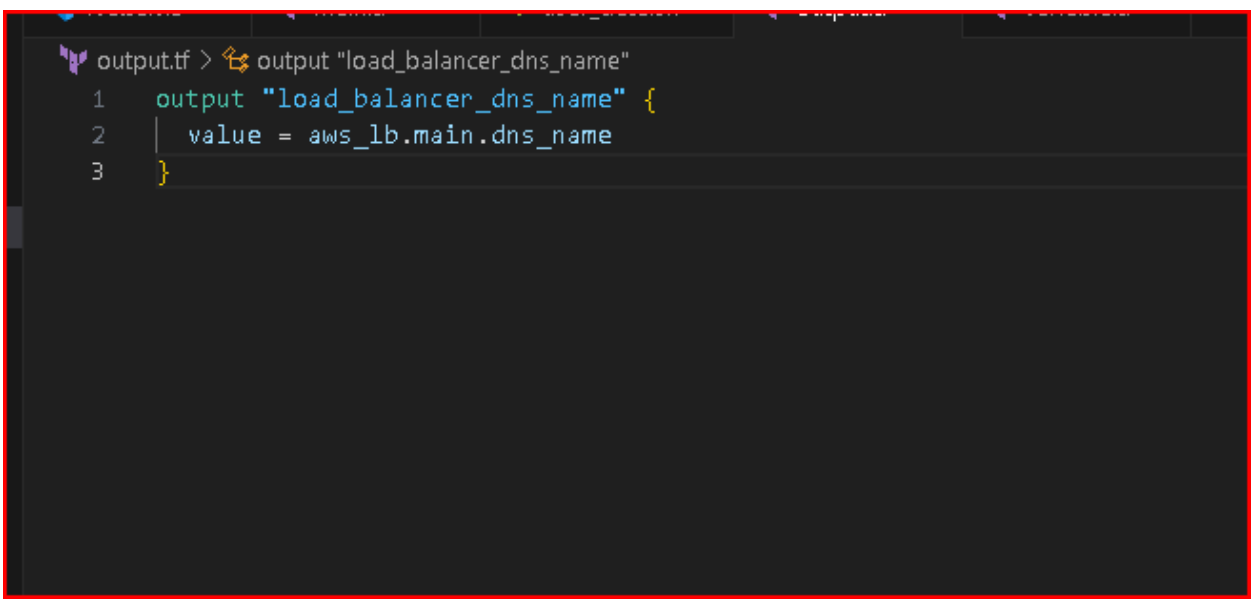
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

aws_instance.example[2]: Still creating... [30s elapsed]
aws_instance.example[0]: Creation complete after 32s [id=i-0b196e763d8cfe84f]
aws_instance.example[1]: Creation complete after 32s [id=i-037dd7591b7d42934]
aws_instance.example[2]: Creation complete after 32s [id=i-0a86fc61029ef6e6a]

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

Outputs:

instance_ids = [
  "i-0b196e763d8cfe84f",
  "i-037dd7591b7d42934",
  "i-0a86fc61029ef6e6a",
]
instance_public_ips = [
  "13.201.21.101",
  "43.204.229.130",
  "43.204.143.85",
]
```

4. Deploy 3-tier application using terraform

a. Set up environment: Install Terraform and AWS CLI.
b. Create directory structure: Organize Terraform files.



c. Write configuration: Define provider, variables, resources, and outputs.



main.tf > resource "aws_subnet" "public" > cidr_block

```tf
provider "aws" {
  region = "us-east-1"
}

# VPC
resource "aws_vpc" "main" {
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "main-vpc"
  }
}

# Subnets
resource "aws_subnet" "public" {
  count                   = 2
  vpc_id                  = aws_vpc.main.id
  cidr_block              = "10.0.${count.index + 1}.0/24"
  map_public_ip_on_launch = true
  availability_zone       = element(data.aws_availability_zones.available.names, count.index)

  tags = {
    Name = "public-subnet-${count.index + 1}"
  }
}

resource "aws_subnet" "private" {
  count          = 2
  vpc_id         = aws_vpc.main.id
  cidr_block     = "10.0.${count.index + 3}.0/24"
  availability_zone = element(data.aws_availability_zones.available.names, count.index)

  tags = {
    Name = "private-subnet-${count.index + 1}"
  }
}

data "aws_availability_zones" "available" {}

# Internet Gateway
resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

  tags = {
    Name = "main-gateway"
  }
}
```

3-TIER-APP-TERRAFORM

- .terraform
- .terraform.lock.hcl
- main.tf
- output.tf
- terraform.tfstate
- terraform.tfstate.backup
- user_data.sh
- variable.tf

variable.tf > variable "EnvironmentName"

```tf
variable "AMI" {
  description = "The AMI ID for the EC2 instances"
  type        = string
  default     = "ami-04a81a99f5ec58529"  # Update with a valid AMI ID
}

variable "KeyPair" {
  description = "The key pair name for SSH access"
  type        = string
  default     = "3-tier-key"  # Update this with your key pair name
}

variable "EnvironmentName" {
  description = "The environment name used for tagging"
  type        = string
  default     = "dev"
}
```

d. Initialize and apply: Use Terraform commands to deploy and verify resources.

```
PS E:\3-Tier-App-Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0

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.
PS E:\3-Tier-App-Terraform>
```

```
PS E:\3-Tier-App-Terraform> terraform plan
data.aws_availability_zones.available: Reading...
aws_vpc.main: Refreshing state... [id=vpc-00bb32d5375a6de2d]
data.aws_availability_zones.available: Read complete after 1s [id=us-east-1]
aws_subnet.public[1]: Refreshing state... [id=subnet-06f23f78695f7ad1b]
aws_subnet.private[1]: Refreshing state... [id=subnet-0c0bcd8eb74108b5e]
aws_lb_target_group.web_tg: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:533267249366:targetgroup/web-tg/ea8a2ed818dc5685]
aws_subnet.private[0]: Refreshing state... [id=subnet-026d56a52d6d73441]
aws_security_group.web_sg: Refreshing state... [id=sg-0da6ec3f9d9af9667]
aws_subnet.public[0]: Refreshing state... [id=subnet-097e81aafa1930dec]
aws_internet_gateway.main: Refreshing state... [id=igw-09aac14be0a9c70c2]
aws_route_table.public: Refreshing state... [id=rtb-078149e93890800f7]
aws_security_group.app_sg: Refreshing state... [id=sg-020a87c3da7ec2e4c]
aws_lb.main: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:533267249366:loadbalancer/app/main-lb/77a3dfe22098d2a5]
aws_instance.web[1]: Refreshing state... [id=i-0b91b9d669ff008ea]
aws_instance.web[0]: Refreshing state... [id=i-0611a4bf8c0781f8b]
aws_db_subnet_group.default: Refreshing state... [id=main-subnet-group]
aws_route_table_association.public[0]: Refreshing state... [id=rtbassoc-0929f2b18b21440c4]
aws_route_table_association.public[1]: Refreshing state... [id=rtbassoc-0c3186d922a0d01dd]
aws_instance.app: Refreshing state... [id=i-0804bffecdb12e518]
aws_security_group.db_sg: Refreshing state... [id=sg-0caf14ec30618310c]
aws_lb_listener.web_listener: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:533267249366:listener/app/main-lb/77a3dfe22098d2a5/ac2841ed16c0b58b]
aws_db_instance.default: Refreshing state... [id=db-O5706WHHT7AGPYM3UWRKJSJKRY]
aws_lb_target_group_attachment.web_tg_attachment[0]: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:533267249366:targetgroup/web-tg/ea8a2ed818dc5685-20240807073259374900
aws_lb_target_group_attachment.web_tg_attachment[1]: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:533267249366:targetgroup/web-tg/ea8a2ed818dc5685-20240807073258910300

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create
  ~ update in-place
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    CODE REFERENCE LOG
aws_lb_listener.web_listener: Creating...
aws_lb_listener.web_listener: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:533267249366:listener/app/main-lb/77a3dfe22098d2a5/ac28
aws_lb.main: Still creating... [3m11s elapsed]
ng... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Creation complete after 3m37s [id=db-O5706WHHT7AGPYM3UWRKJSJKRY]
ng... [3m20s elapsed]
ng... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Creation complete after 3m37s [id=db-O5706WHHT7AGPYM3UWRKJSJKRY]
ng... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
ng... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Creation complete after 3m37s [id=db-O5706WHHT7AGPYM3UWRKJSJKRY]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Creation complete after 3m37s [id=db-O5706WHHT7AGPYM3UWRKJSJKRY]


Apply complete! Resources: 22 added, 0 changed, 0 destroyed.

Outputs:
Outputs:

load_balancer_dns_name = "main-lb-1928893195.us-east-1.elb.amazonaws.com"
```

e. Copy LB dns and hit

http://main-lb-1928893195.us-east-1.elb.amazonaws.com/

f. Clean up: Destroy resources when no longer needed.

```
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 1m40s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 1m50s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 2m0s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 2m10s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 2m20s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 2m30s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 2m40s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 2m50s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 3m0s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 3m10s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 3m20s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 3m30s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 3m40s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 3m50s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 4m0s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 4m10s elapsed]
aws_db_instance.default: Still destroying... [id=db-05706WHHT7AGPYM3UWRKJSJKRY, 4m20s elapsed]
aws_db_instance.default: Destruction complete after 4m29s
aws_security_group.db_sg: Destroying... [id=sg-0caf14ec30618310c]
aws_db_subnet_group.default: Destroying... [id=main-subnet-group]
aws_db_subnet_group.default: Destruction complete after 1s
aws_subnet.private[0]: Destroying... [id=subnet-026d56a52d6d73441]
aws_subnet.private[1]: Destroying... [id=subnet-0c0bcd8eb74108b5e]
aws_security_group.db_sg: Destruction complete after 2s
aws_security_group.app_sg: Destroying... [id=sg-020a87c3da7ec2e4c]
aws_subnet.private[1]: Destruction complete after 1s
aws_subnet.private[0]: Destruction complete after 1s
aws_security_group.app_sg: Destruction complete after 1s
aws_security_group.web_sg: Destroying... [id=sg-0da6ec3f9d9af9667]
aws_security_group.web_sg: Destruction complete after 2s
aws_vpc.main: Destroying... [id=vpc-00bb32d5375a6de2d]
aws_vpc.main: Destruction complete after 0s

Destroy complete! Resources: 21 destroyed.
```