**Terraform Remote State**

In this project we are doing to explore the importance of using terraform Remote State by creating an AWS Dynamodb table to collect the Tf.state file and later saving the State file in an S3 bucket. This is important because it prevent multiple people from doing changes on the same terraform repo by locking the state file when a change a about to be made by one person. Using S3 buckets to remotely safe the state file also serves as a backup in case the local state file is damaged. Using the remote location also make is possible for team collaboration.

1. Create a new directory for your Terraform project and create a main.tf file:

**mkdir terraform-remote-state-example && cd terraform-remote-state-example touch main.tf**

1. Provide AWS provider and EC2 resource in your main.tf file:

**provider "aws" {**

**region = "us-east-1"**

**}**

**resource "aws\_instance" "example" {**

**ami = "ami-007855ac798b5175e"**

**instance\_type = "t2.micro"**

**}**

1. Initialize and apply your Terraform configuration:

**terraform init**

**terraform plan**

**terraform apply**

1. Create a new S3 bucket to store the Terraform state remotely. Replace your-unique-state-bucket-name with a unique bucket name:

**aws s3api create-bucket --bucket tcb-devops-state-demo-x922x --region us-east-1**

1. Note that the backend.tf configuration also specifies a DynamoDB table (terraform-lock-table) for state locking. State locking helps prevent concurrent modification of the state file, which can lead to corruption. Let's create the DynamoDB table manually:

**aws dynamodb create-table \\**

**--table-name tcb-devops-state-lock-table \\**

**--attribute-definitions AttributeName=LockID,AttributeType=S \\**

**--key-schema AttributeName=LockID,KeyType=HASH \\**

**--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \\**

**--region us-east-1**

1. Create a backend.tf file in your project directory:

**touch backend.tf**

1. Edit the backend.tf file with the following content. Replace your-unique-state-bucket-name with the name of the bucket you created:

**terraform {**

**backend "s3" {**

**bucket = "tcb-devops-state-demo-x922x"**

**key = "terraform.tfstate"**

**region = "us-east-1"**

**encrypt = true**

**dynamodb\_table = "tcb-devops-state-lock-table"**

**}**

**}**

1. Initialize Terraform:

**terraform init**

(During the initialization, Terraform will prompt you to confirm the migration of the local state to the remote state. Type yes and press Enter to confirm.)

1. Apply the Terraform configuration:

**terraform apply**

**(**Now, Terraform will store the state remotely in the specified S3 bucket. You can verify this by checking the contents of the S3 bucket using the AWS CLI or the AWS Management Console.)

1. Destroy the resources:

**terraform destroy**

**(**After destroying the resources, you can delete the remote state bucket and the DynamoDB table using the AWS CLI or the AWS Management Console.)