Lesson 10 Demo 05

Implementing Terraform State Locking

Objective: To implement state locking in Terraform, ensuring concurrent operations on the Terraform state file are managed to prevent conflicts and corruption

Tools required: Visual Studio Code

Prerequisites: Ensure you have created and implemented the AWS access key and secret key before starting this demo. Refer to Lesson 08 Assisted Practice 02 for detailed steps

Note: The folder structure created in the previous demos is used here. It is also included in the resources section of LMS. Please refer Lesson_10_demo_01

Steps to be followed:

1. Update the Terraform configuration
2. Generate a Terraform state lock
3. Specify a Terraform lock timeout

Step 1: Update the Terraform configuration

1.1 Update the **main.tf** file to change the tags value of your aws_instance.web_server_2 block by using the following code:

```
resource "aws_instance" "web_server_2" {
  ami          = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnets["public_subnet_2"].id
  tags = {
    Name = "Web EC2 Server 2"
  }
}
```

```
main.tf   terraform.tf   variables.tf
main.tf
360
361  # Terraform Resource Block - To Build EC2 instance in Public Subnet
362  resource "aws_instance" "web_server_2" {
363    ami           = data.aws_ami.ubuntu.id
364    instance_type = "t2.micro"
365    subnet_id     = aws_subnet.public_subnets["public_subnet_2"].id
366    tags = {
367      Name = "Web EC2 Server 2"
368    }
369  }
```

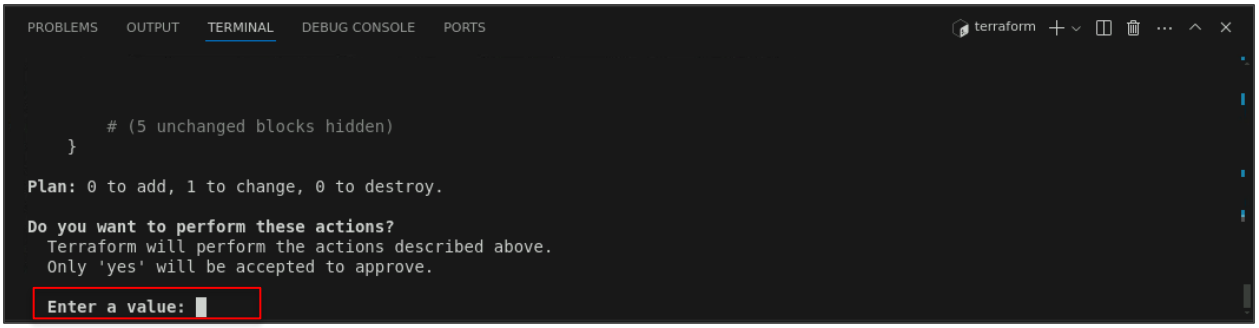
Step 2: Generate a Terraform state lock

2.1 Generate a lock on your state file by using the following command:
terraform apply

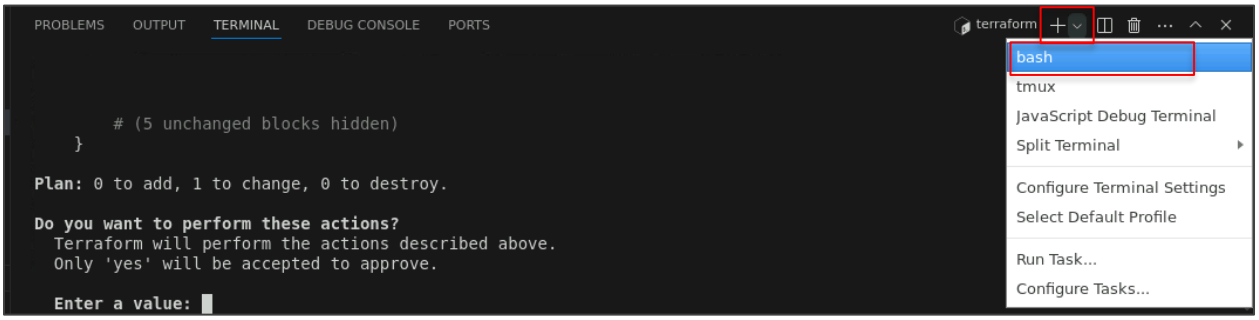
```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE   PORTS
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply
tls_private_key.generated: Refreshing state... [id=6e0401af932d02263816c82fb45a6bf99c9fa346]
local_file.private_key_pem: Refreshing state... [id=1f539a7fa34d88b2e8a5cf0935e11ccb75d0ebf0]
random_string.random: Refreshing state... [id=sN[Q3QR+lE]
aws_key_pair.generated: Refreshing state... [id=MyAWSKey]
aws_vpc.vpc: Refreshing state... [id=vpc-077c667402e032623]
aws_internet_gateway.internet_gateway: Refreshing state... [id=igw-01ffeafe8937357ef]
aws_security_group.vpc-ping: Refreshing state... [id=sg-05c866dc8db7d73c8]
aws_security_group.vpc-web: Refreshing state... [id=sg-043d9a3a6e8acc8ae]
aws_subnet.private_subnets["private_subnet_1"]: Refreshing state... [id=subnet-0e613e87598c2a02e]
aws_subnet.private_subnets["private_subnet_2"]: Refreshing state... [id=subnet-03a44332a19e49d93]
aws_subnet.private_subnets["private_subnet_3"]: Refreshing state... [id=subnet-0fa417dc9f4fbc2e4]
aws_security_group.ingress-ssh: Refreshing state... [id=sg-045f4f75838c0cc9a]
aws_subnet.public_subnets["public_subnet_2"]: Refreshing state... [id=subnet-00149eda4bed0de7b]
aws_subnet.public_subnets["public_subnet_3"]: Refreshing state... [id=subnet-019ef181be686d113]
aws_subnet.public_subnets["public_subnet_1"]: Refreshing state... [id=subnet-0d049c1ebbd0a0872]
bash
bash
```

2.2 When prompted with the following, do not provide any answer at this time:
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value:



2.3 Open a new terminal within the same working directory by clicking on the **+** icon and then selecting **bash**



2.4 In the newly opened terminal, run the following command to check if the state lock is applied properly:
terraform apply

```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE   PORTS

sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply

  Error: Error acquiring the state lock

  Error message: resource temporarily unavailable
  Lock Info:
    ID:        70044291-190c-3ab4-dd96-fcdcf1148fd7
    Path:      terraform.tfstate
    Operation: OperationTypeApply
    Who:       sakshiguptasimp@ip-172-31-22-2
    Version:   1.1.6
    Created:   2024-06-27 10:40:42.018342317 +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.

sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

This error message indicates that the state lock is applied successfully.

Step 3: Specify a Terraform lock timeout

3.1 In the newly created terminal, use the following command to apply a lock timeout value:
terraform apply -lock-timeout=60s

```
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$ terraform apply -lock-timeout=60s
Acquiring state lock. This may take a few moments...
```

3.2 Go to the previous terminal and answer **no** to the terraform apply command to free the state lock

```
        }
        # (27 unchanged attributes hidden)




        # (5 unchanged blocks hidden)
    }

Plan: 0 to add, 1 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: no
```

3.3 Go to the newly created terminal, and you should observe the terraform apply process is successful once the state lock is released

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE    PORTS
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.web_server_2: Modifying... [id=i-036af2cfbd9f2d55f]
aws_instance.web_server_2: Modifications complete after 1s [id=i-036af2cfbd9f2d55f]

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

Outputs:

public_dns = "ec2-44-201-46-186.compute-1.amazonaws.com"
public_dns_server_subnet_1 = "ec2-34-227-195-55.compute-1.amazonaws.com"
public_ip = "44.201.46.186"
public_ip_server_subnet_1 = "34.227.195.55"
size = "t2.micro"
sakshiguptasimp@ip-172-31-22-2:~/Desktop/Terraform$
```

By following the above steps, you have successfully implemented state locking in Terraform and ensured concurrent operations on the Terraform state file are managed.