**Terraform State List and Show commands**

* These two commands comes under **Terraform Inspecting State**
* **terraform state list:** This command is used to list resources within a Terraform state.
* **terraform state show:** This command is used to show the attributes of a single resource in the Terraform state.

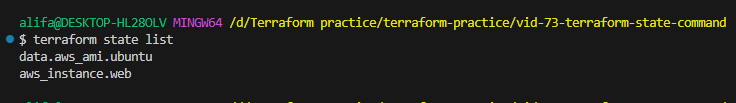
# List Resources from Terraform State

terraform state list

# Show the attributes of a single resource from Terraform State

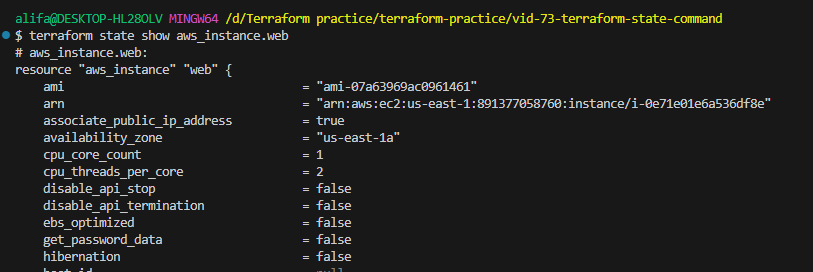
terraform state show data.aws\_ami.amzlinux

terraform state show aws\_instance.my-ec2-vm



After applying terraform plan, terraform apply, and terraform state list, you will be able to see the list of resources in the state list.

Then when you write terraform state show aws\_instance.web (or any resource name from the list within the state list) you will be able to see the plan/information of it



**Step-05-02: Terraform State mv command**

* This commands comes under **Terraform Moving Resources**
* This command will move an item matched by the address given to the destination address.
* This command can also move to a destination address in a completely different state file
* Very dangerous command
* Very advanced usage command
* Results will be unpredictable if concept is not clear about terraform state files mainly desired state and current state.
* Try this in production environments, only when everything worked well in lower environments.

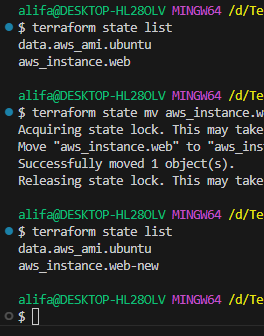
# Terraform List Resources

terraform state list

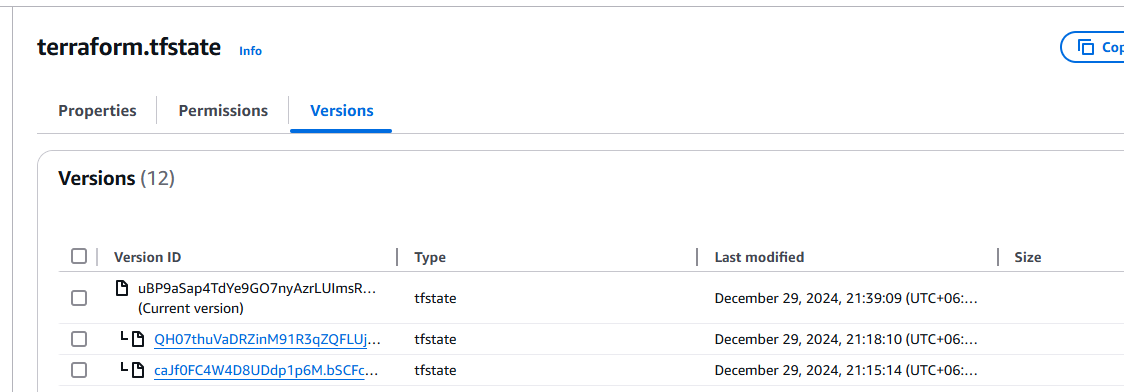
# Terraform State Move Resources to different name

terraform state mv -dry-run aws\_instance.my-ec2-vm aws\_instance.my-ec2-vm-new

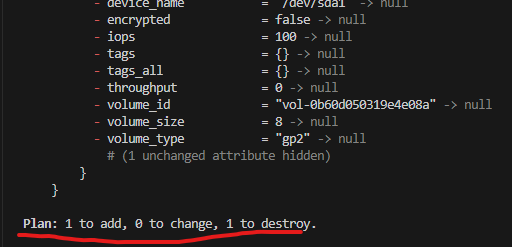
terraform state mv aws\_instance.web aws\_instance.web-new



ls -lrta



New state file created.



When terraform plan was applied,

This is wrong approach as the current ec2 will be destroyed and new will be created. We don’t want that.

Observation:

1) It should create a backup file of terraform.tfstate as something like this "terraform.tfstate.1611929587.backup"

1) It renamed the name of " aws\_instance.web " in state file to " aws\_instance.web -new".

2) Run terraform plan and observe what happens in next run of terraform plan and apply

-----------------------------

# WRONG APPROACH

-----------------------------

# WRONG APPROACH OF MOVING TO TERRAFORM PLAN AND APPLY AFTER ABOVE CHANGE terraform state mv CHANGE

# WE NEED TO UPDATE EQUIVALENT RESOURCE in terraform manifests to match the same new name.

# Terraform Plan

terraform plan

Observation: It will show "Plan: 1 to add, 0 to change, 1 to destroy."

1 to add: New EC2 Instance will be added

1 to destroy: Old EC2 instance will be destroyed

DON'T DO TERRAFORM APPLY because it shows make changes. Nothing changed other than state file local naming of a resource. Ideally nothing on current state (real cloud environment should not change due to this)

-----------------------------

-----------------------------

# RIGHT APPROACH

-----------------------------

Update "c3-ec2-instance.tf"

Before: resource "aws\_instance" "my-ec2-vm" {

After: resource "aws\_instance" "my-ec2-vm-new" {

Update all references of this resources in your terraform manifests

Update c5-outputs.tf

Before: value = aws\_instance.my-ec2-vm.public\_ip

After: value = aws\_instance.my-ec2-vm-new.public\_ip

Before: value = aws\_instance.my-ec2-vm.public\_dns

After: value = aws\_instance.my-ec2-vm-new.public\_dns

Now run terraform plan and you should see no changes to Infra

# Terraform Plan

terraform plan

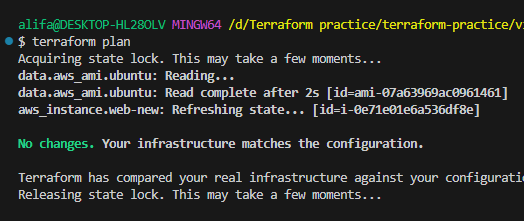
Observation:

1) Message-1: No changes. Infrastructure is up-to-date

2) Message-2: This means that Terraform did not detect any differences between your

configuration and real physical resources that exist. As a result, no

actions need to be performed.



This is what we get when we changed the code to match out infrastructure.