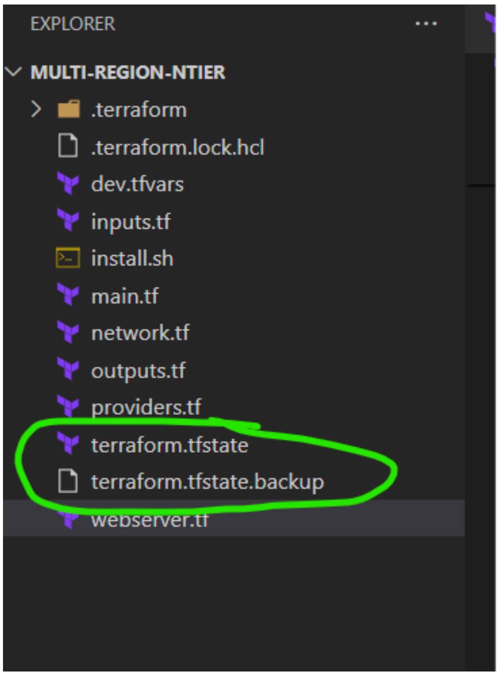
# Terraform state

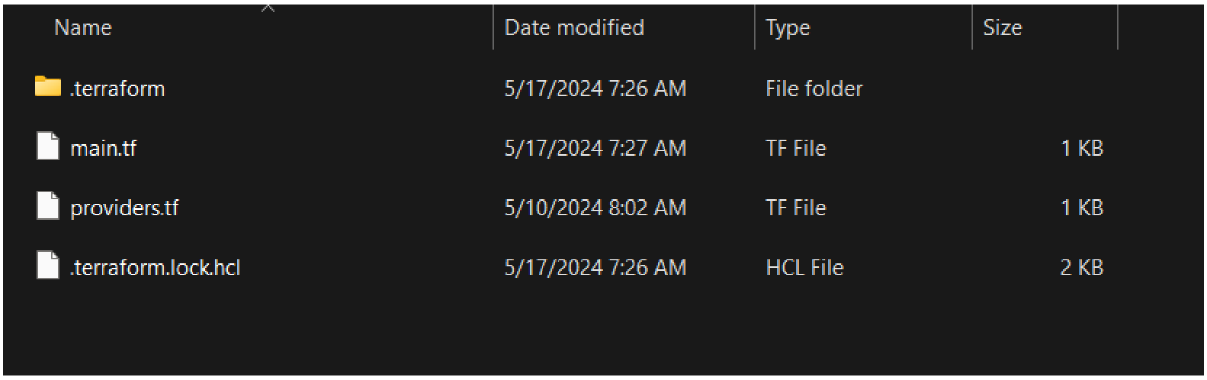
- When we execute terraform apply, we would see couple of files as highlighted below



EXPLORER

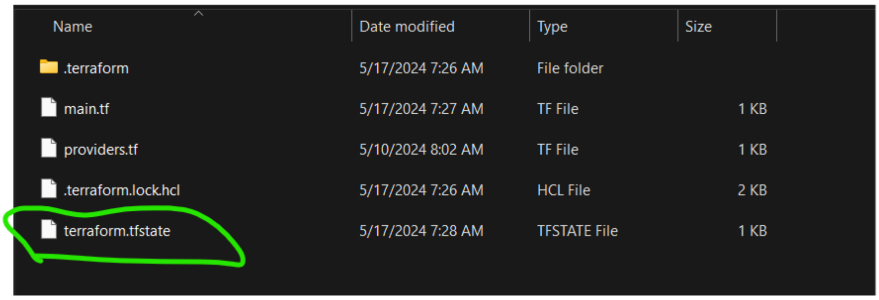
MULTI-REGION-NTIER
- .terraform
- .terraform.lock.hcl
- dev.tfvars
- inputs.tf
- install.sh
- main.tf
- network.tf
- outputs.tf
- providers.tf
- terraform.tfstate
- terraform.tfstate.backup
- webserver.tf

- These files represent terraform state.
- Lets try to understand what terraform state is.
  - Terraform state is a persistent data structure used by terraform to keep track of the infrastructure managed by it.
- Experimenting: Created a new template
  - Before first apply



| Name | Date modified | Type | Size |
|---|---|---|---|
| .terraform | 5/17/2024 7:26 AM | File folder | |
| main.tf | 5/17/2024 7:27 AM | TF File | 1 KB |
| providers.tf | 5/10/2024 8:02 AM | TF File | 1 KB |
| .terraform.lock.hcl | 5/17/2024 7:26 AM | HCL File | 2 KB |

  - After succesfull apply i.e. resources getting created

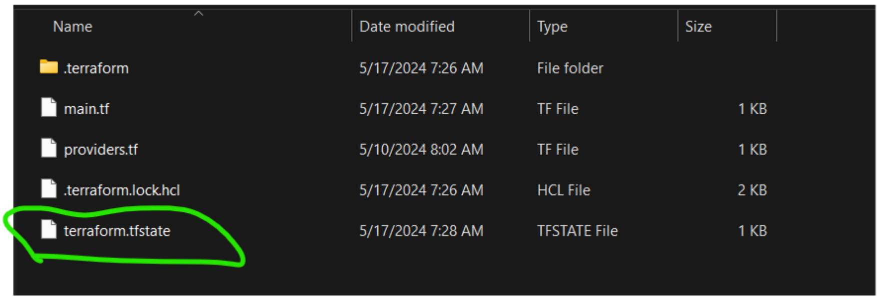


| Name | Date modified | Type | Size |
|---|---|---|---|
| .terraform | 5/17/2024 7:26 AM | File folder | |
| main.tf | 5/17/2024 7:27 AM | TF File | 1 KB |
| providers.tf | 5/10/2024 8:02 AM | TF File | 1 KB |
| .terraform.lock.hcl | 5/17/2024 7:26 AM | HCL File | 2 KB |
| terraform.tfstate | 5/17/2024 7:28 AM | TFSTATE File | 1 KB |

  - Observerations: terraform.tfstate is created after successful creation of resources by terraform
  - Contents of this file are json reprsentation of what has been managed by terraform

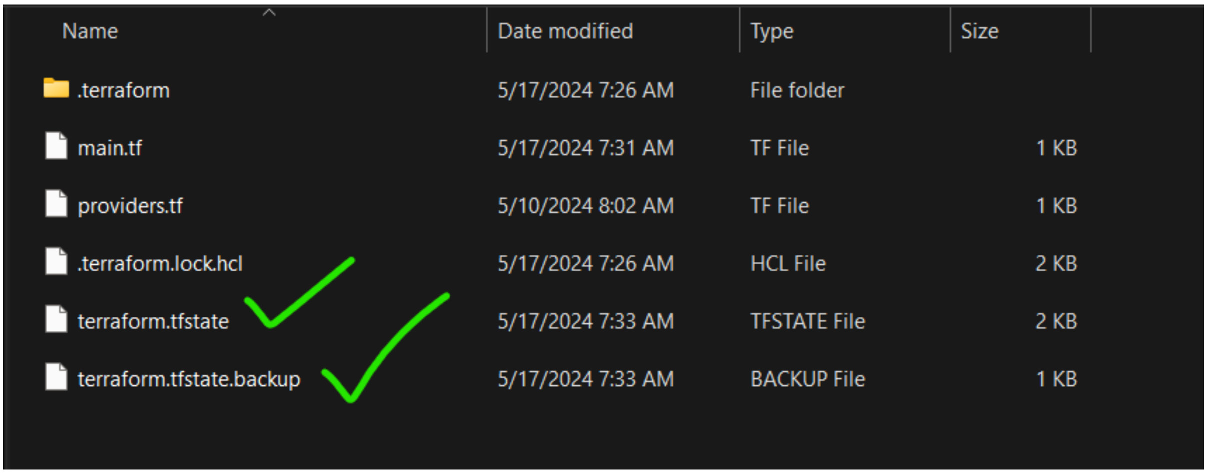
```
{
    "version": 4,
    "terraform_version": "1.8.0",
    "serial": 1,
    "lineage": "306861d6-b36e-585d-c8ea-3d1dbed07f9f",
    "outputs": {},
    "resources": [
        {
        "mode": "managed",
        "type": "azurerm_resource_group",
        "name": "sample",
        "provider": "provider[\"registry.terraform.io/hashicorp/azurerm\"]",
        "instances": [
            {
            "schema_version": 0,
            "attributes": {
                "id": "/subscriptions/7ee23928-6bf0-4a1b-8e1d-b854f8f98d81/resourceGroups/dummy",
                "location": "eastus",
                "managed_by": "",
                "name": "dummy",
                "tags": null,
                "timeouts": null
            },
            "sensitive_attributes": [],
            "private": "eyJlMmJmYjczMC1lY2FhLTExZTYtOGY4OC0zNDM2M2JjN2M0YzAiOnsiY3JlYXRlIjo1NDAwMDAwMDAwMDAwLCJkZWxldGUiOjU0MDAwMDAwMDAwMDAsInJlYWQiOjMwMDAwMDAwMDAwMCwidXBkYXRlIjo1NDAwMDAwMDAwMDAwfX0="
            }
        ]
        }
    ],
    "check_results": null
    }
```

- Now lets make some more changes in the template, we have added tags
- Before apply the folder structure is as shown below



| Name | Date modified | Type | Size |
|---|---|---|---|
| .terraform | 5/17/2024 7:26 AM | File folder | |
| main.tf | 5/17/2024 7:27 AM | TF File | 1 KB |
| providers.tf | 5/10/2024 8:02 AM | TF File | 1 KB |
| .terraform.lock.hcl | 5/17/2024 7:26 AM | HCL File | 2 KB |
| terraform.tfstate | 5/17/2024 7:28 AM | TFSTATE File | 1 KB |

- After succesful apply the folder structure is as shown below

  | Name | Date modified | Type | Size |
  |---|---|---|---|
  | .terraform | 5/17/2024 7:26 AM | File folder | |
  | main.tf | 5/17/2024 7:31 AM | TF File | 1 KB |
  | providers.tf | 5/10/2024 8:02 AM | TF File | 1 KB |
  | .terraform.lock.hcl | 5/17/2024 7:26 AM | HCL File | 2 KB |
  | terraform.tfstate ✓ | 5/17/2024 7:33 AM | TFSTATE File | 2 KB |
  | terraform.tfstate.backup ✓ | 5/17/2024 7:33 AM | BACKUP File | 1 KB |

- The backup files is back up of tfstate before the current apply i.e. it contains previous state

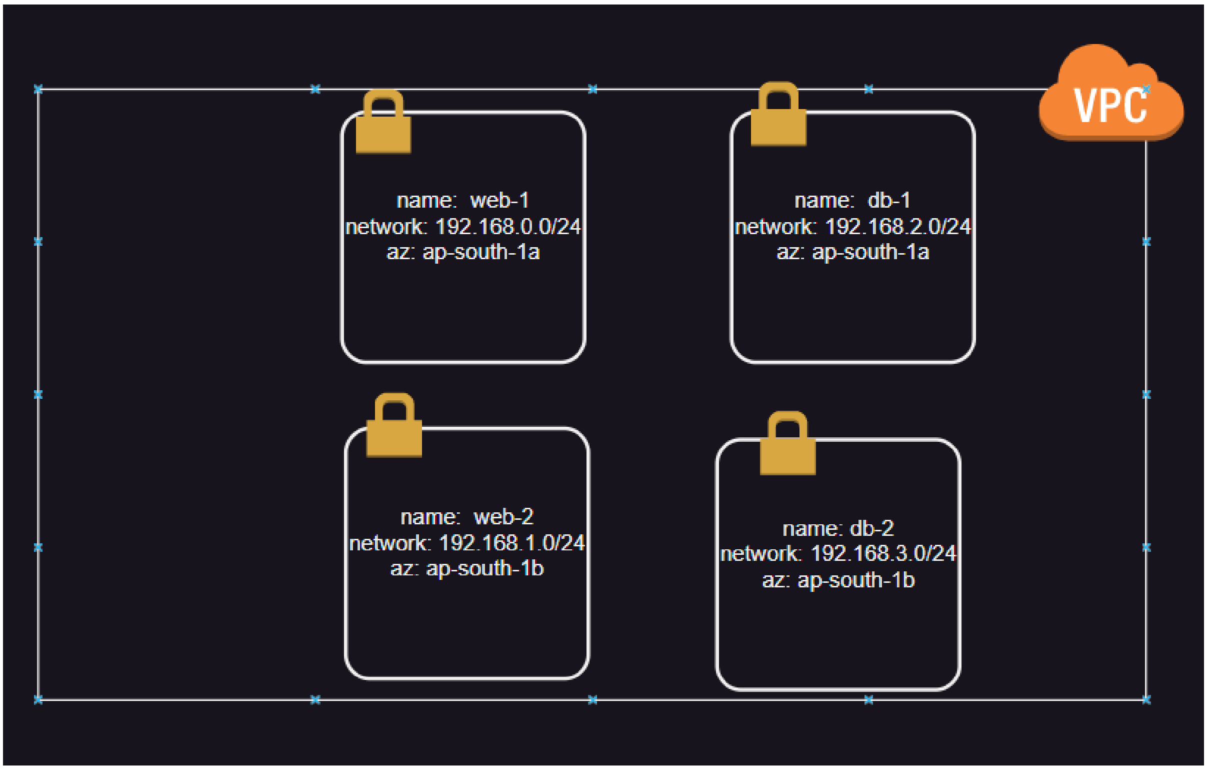
- Whenever we execute terraform apply, terraform will refresh state to get the current status of resources managed by terraform and this will be compared with desired (written in the template), difference between actual state and desired state becomes terraform plan.
- This plan when executed creates the desired state.
- State file should never be edited manually.
- By default state is stored in the local folder from where the terraform execution is done.

## Setup AWS Network for ntier application

- Lets create the aws network to host the ntier application as what we have done in azure

- Steps:

  - ensure we have vpc with 4 subnets

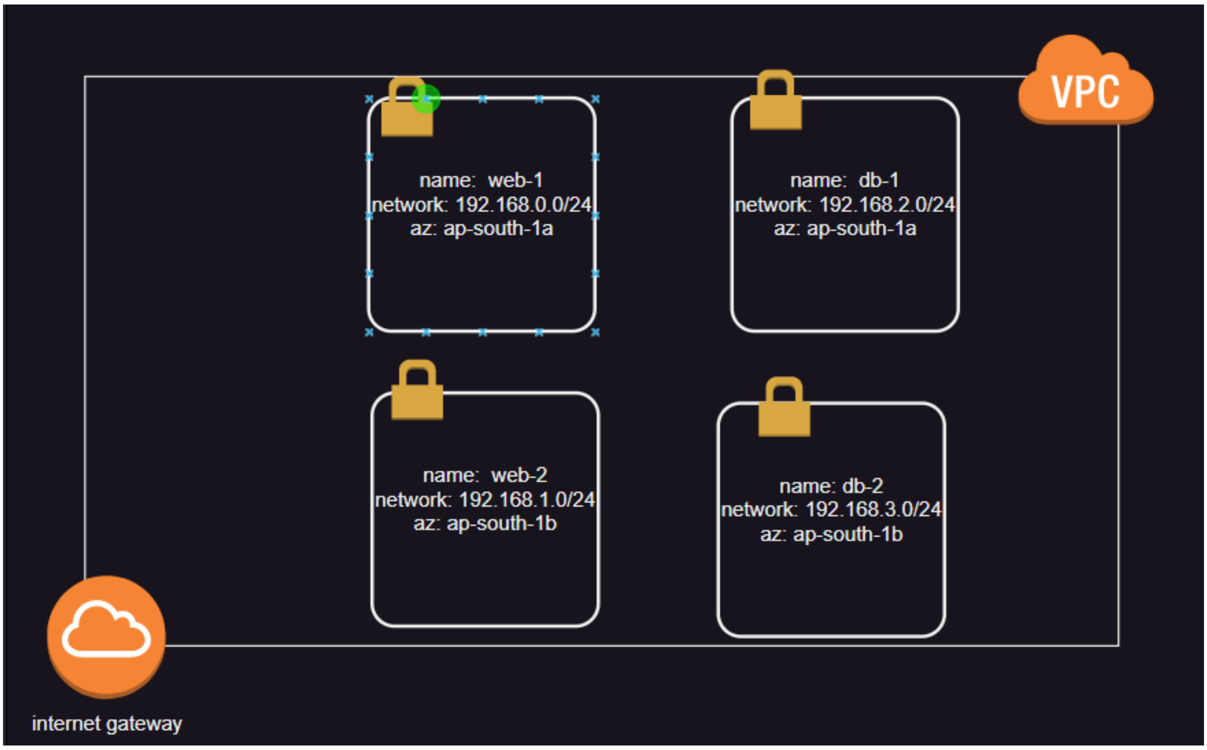
  name: web-1
  network: 192.168.0.0/24
  az: ap-south-1a

  name: db-1
  network: 192.168.2.0/24
  az: ap-south-1a

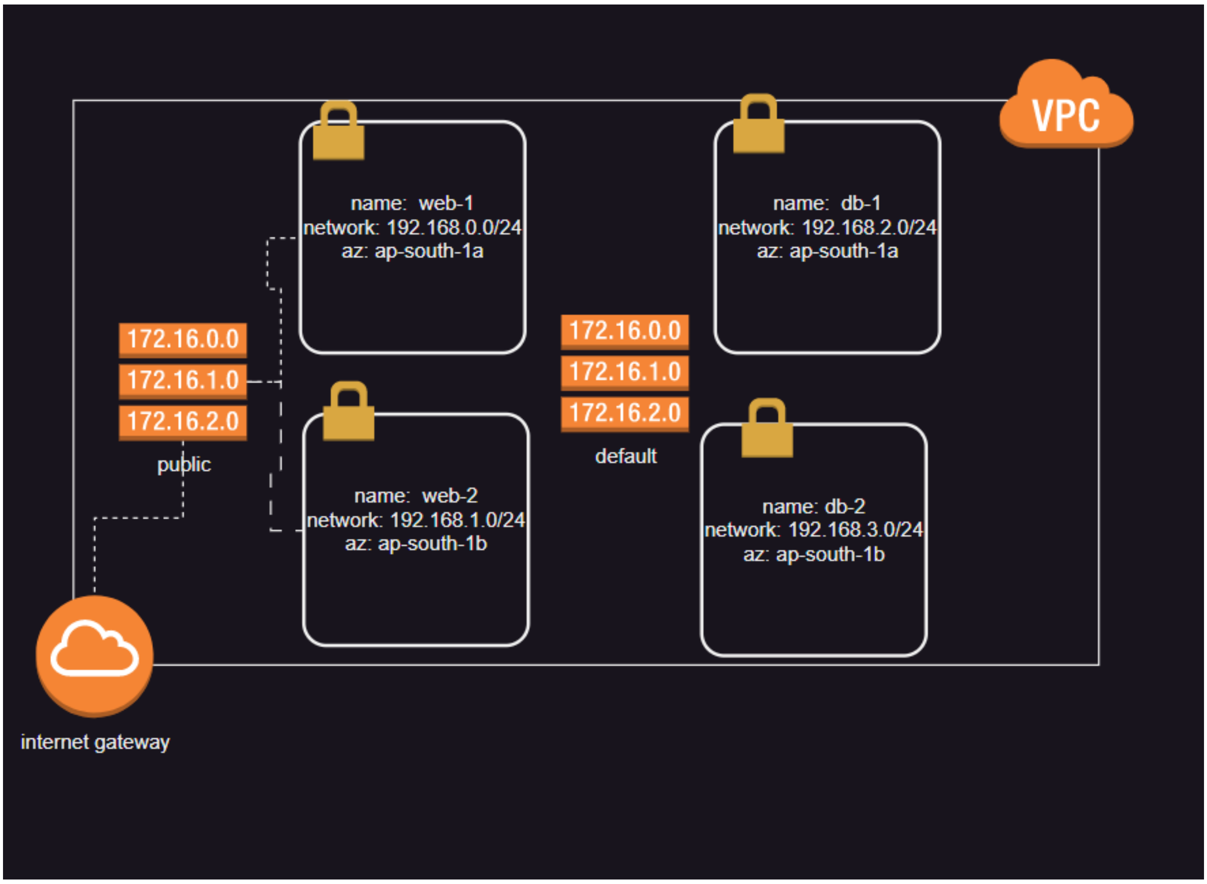
  name: web-2
  network: 192.168.1.0/24
  az: ap-south-1b

  name: db-2
  network: 192.168.3.0/24
  az: ap-south-1b

- create an internet gateway and attach to the vpc



- create a route table and call it as public
- associate web1, web2 to public route table
- add a route in public route table table to reach internet gateway



- We need to create a security group for allowing 80,22 port traffic
- create an ec2 instance in web 1 subnet with public ip and security group created above and then login in and execute the script

```
sudo apt update
sudo apt install nginx unzip -y
cd /tmp && wget https://www.free-css.com/assets/files/free-css-templates/download/page295/kider.zip
unzip /tmp/kider.zip
sudo mv /tmp/preschool-website-template /var/www/html/preschool
```

- We need a terraform template to create vpc with
  - public subnets
  - private subnets
- Refer Here for the template written in terraform.