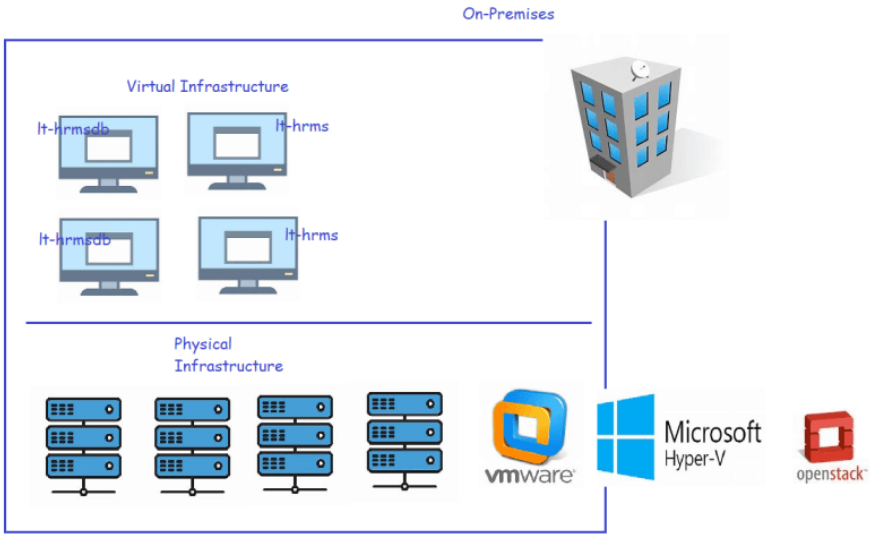
An Organizational Scenario

- This is scenario of an organization called as Learning Thoughts.
- Learning Thoughts has an application for HRMS => It-hrms[ADP, Docker_install]
- Learning Thoughts has many customers and Each of customers have their own server infrastructure. Some organizations have

On-Premise Infrastructure:



Cloud Infrastructure

- So, how does It-hrms handle the variations in server infrastructure
- Automating deployment for multiple server infrastructures is also a challenge
    - VMWare has its own way of automation
    - HyperV has its own ways of automation
    - OpenStack Has its own way of automation
    - Azure => Azure CLI/Powershell or ARM Templates
    - AWS => AWS CLI or CloudFormation
    - GCloud => GCloud cli or Gcloud templates

- So, we need an effective solution which can deploy It-hrms app in multiple server infrastructure (Creating servers) and then to deploy application into individual server using shell/PowerShell/ansible/chef etc

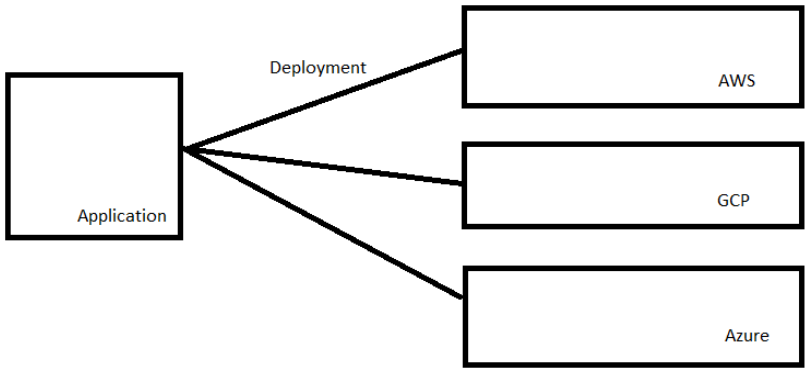
What are possible solutions?

  - Writing a script for all the possible infrastructures might be complex and making changes will become difficult
  - Create your own tool with the help of dev team. Maintenance becomes a challenge
- To solve these kind of issues, we need InfraProvisioning tools. We have the infraprovisioning tools such as AWS Cloudformation, Azure ARM Tempalte, Openstack Heat but they work only their environments
- Terraform is an infra provisioning tool which can work with various server infrastructures and the way we approach to write the template in terraform is declarative

Terraform supports Declarative Language

- Terraform is an Opensource tool developed by HashiCorp using Go Language. Terraform is all about copying one binary(executable) in the system
- 5000 feet overview of how terraform works

Diagram to represent terraform



Terraform Installation:

URL: https://www.terraform.io/downloads

macOS **Windows** Linux FreeBSD OpenBSD Solaris

```
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.r
sudo apt update && sudo apt install terraform
```

```
Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
```

- Infrastructure as Code (IaC):
  - This means expressing our infra needs in the form of some template.
  - While creating/realizing the infra, pass the dynamic values.
- Terraform:
  - IaC which runs on any virtual platform
- What do we have to do
  - Define your infrastructure as a template in Terraform
  - Execute the template to create infrastructure.
- In which Language do we need to write these templates?
  - Hashicorp Configuration Language (HCL)
- Here we express what we want in the template which is referred as Desired State
- Now when execute terraform will try to create infra to match your Desired State.
- Idempotence: This is property which states that executing once or multiple times will have the same result
- For infra-provisioning we need a template which helps in meeting desired state and is idempotent.
- There are two popular tools
  - Terraform
  - Pulumi

Note: Consider docker install shell script, and how it manages to install docker in multiple environments.

Terraform

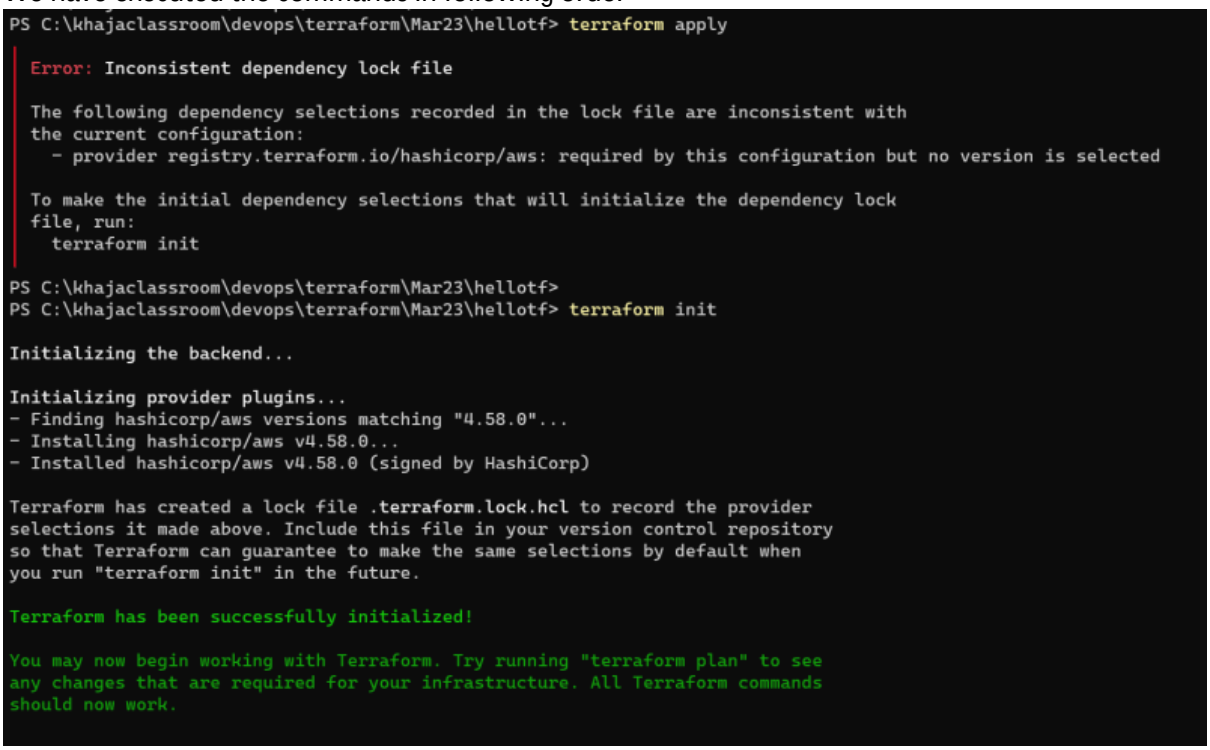
- Terraform is an opensource tool developed by HashiCorp which can create infra in almost any virtual platform
- Terraform uses a language which is called as Hashicorp Configuration Language (HCL) to express desired State.

Terms

- Resource: This is the infrastructure which you want to create
- Provider: This refers to where you want to create infrastructure
- The inputs which we express in teraform are called as arguments
- The output given by terraform is referred as attribute

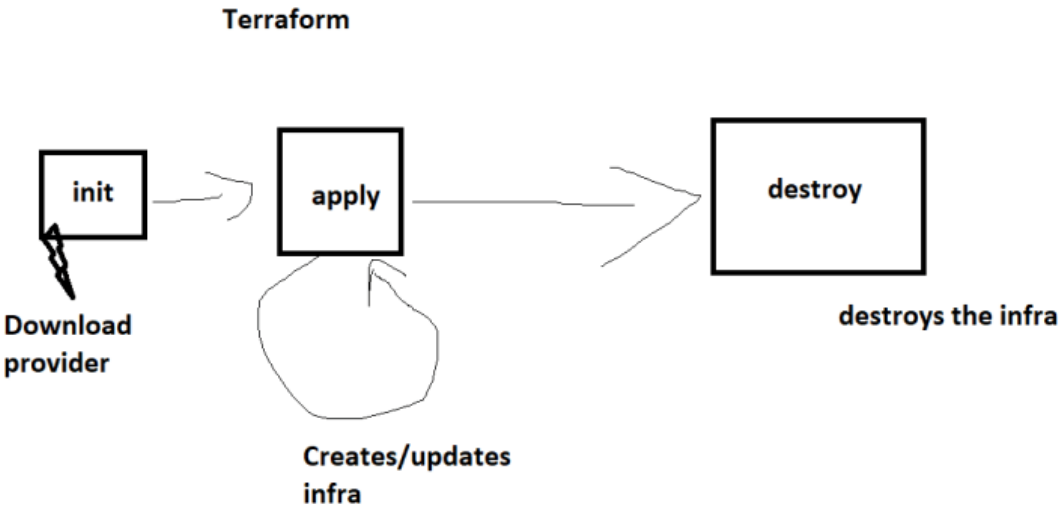
Installing terraform

- https://developer.hashicorp.com/terraform/downloads
- https://developer.hashicorp.com/terraform
- We have executed the commands in following order



```
PS C:\khajaclassroom\devops\terraform\Mar23\hellotf> terraform apply

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_s3_bucket.b will be created
  + resource "aws_s3_bucket" "b" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "qt-tf-test-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Environment" = "Dev"
          + "Name"        = "My bucket"
        }
    }

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_s3_bucket.b: Creating...
aws_s3_bucket.b: Creation complete after 8s [id=qt-tf-test-bucket]
aws_s3_bucket_acl.example: Creating...
aws_s3_bucket_acl.example: Creation complete after 1s [id=qt-tf-test-bucket,private]

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

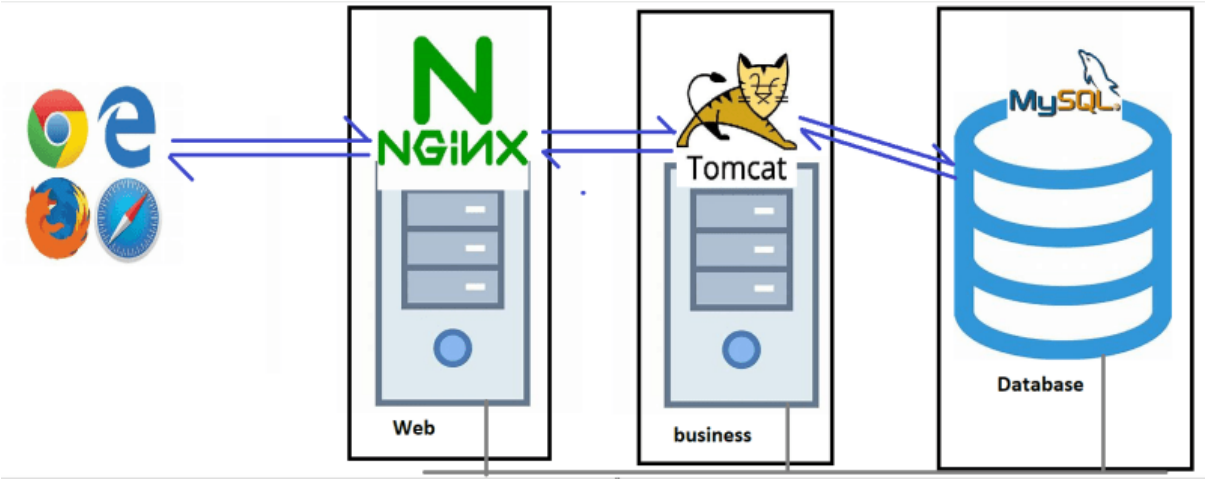
Terraform basic workflow



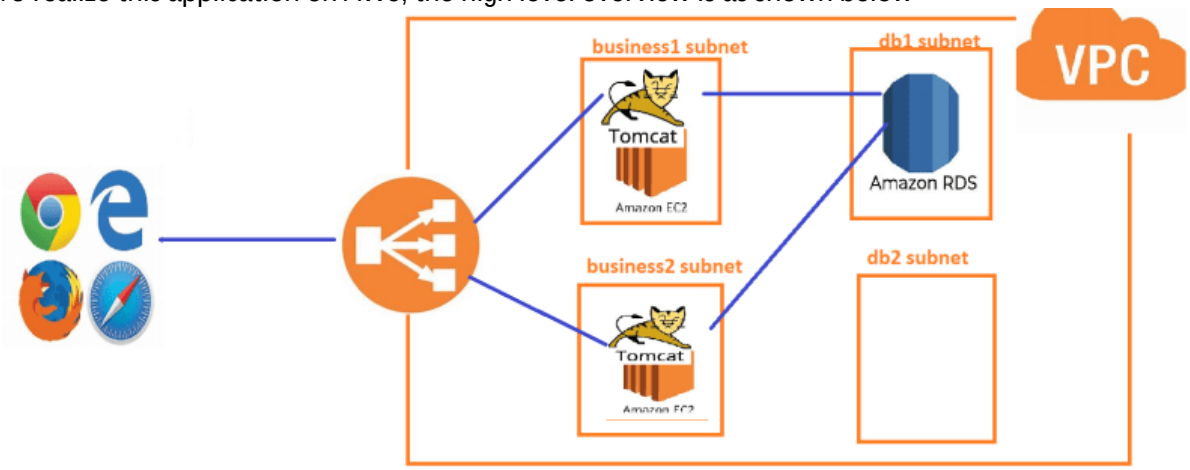
Casestudy: Terraform as cost effective solution.

N-Tier Application

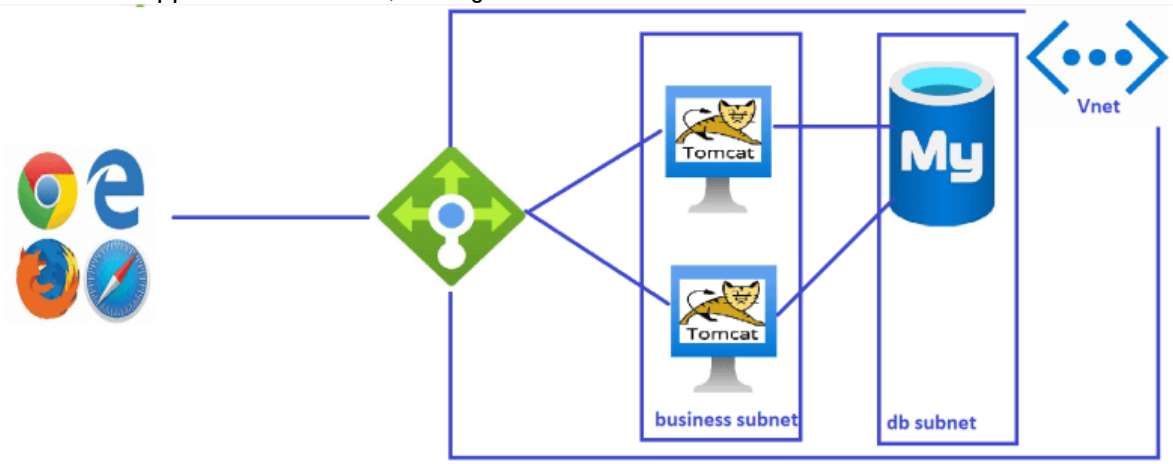
- Consider the following architecture of a typical web application (ticket booking)



- To realize this application on AWS, the high level overview is as shown below



- To realize this application on Azure, the high level overview is as shown below



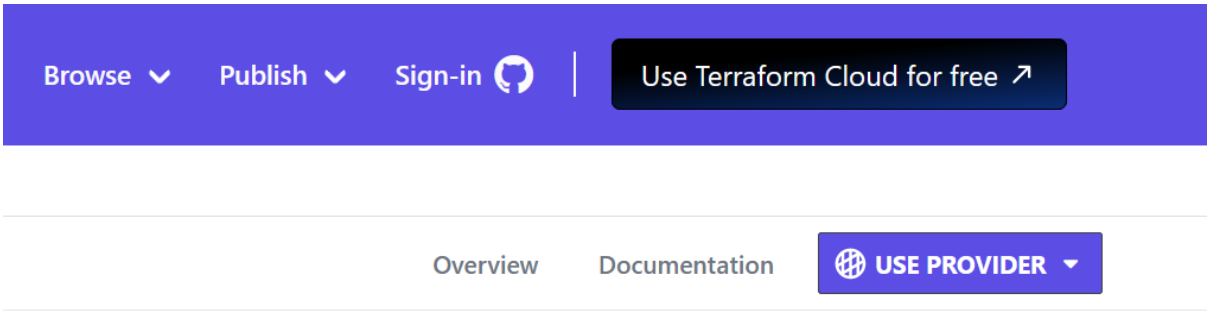
WOW (Ways of Working)

- Let's realize the architecture manually, make a note of
  - resource
  - inputs
  - outputs
- Find resources in Terraform to acheive the above manual steps

Configuring a Provider in Terraform

AWS Provider

- Terraform aws provider uses the AWS APIs to get the infra created.
- To Create infrastructure in your AWS Account, it needs AWS programatic credentials (Secret key and access key)
- To configure these keys



- Create IAM Secret key and access key https://sst.dev/chapters/create-an-iam-user.html for manual steps
- Lets write provider configuration

```
provider "aws" {
  region = "us-west-2"
  access_key = "AKIAZ4ECZC3PP5DPPHG6"
  secret_key = "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"
}
```

- This is not a great way as we are having sensitive information in the text format.
- Best way is to install aws cli on the machine with terraform and terraform will automatically pickup credentials from there.
- Installing aws cli https://registry.terraform.io/providers/hashicorp/aws/latest
- Now your provider can be as simple as

```
provider "aws" {
  region = "us-west-2"
}
```

Note: If you don't want to share your key then do aws configure, and update your key over their. Terraform will pick key from them.

Azure Provider

- https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs for the provider documentation
- To install azure cli https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt#option-1-install-with-one-command To authenticate azure cli

az login
az group list

Providers and resources

- In terraform to create any resource we need to configure provdier
- Every provider has a specific structure

```
provider "<name>" {
  <ARGUMENT-1> = <VALUE-1>
  ..
  ..
  ..
  <ARGUMENT-N> = <VALUE-N>

}
```

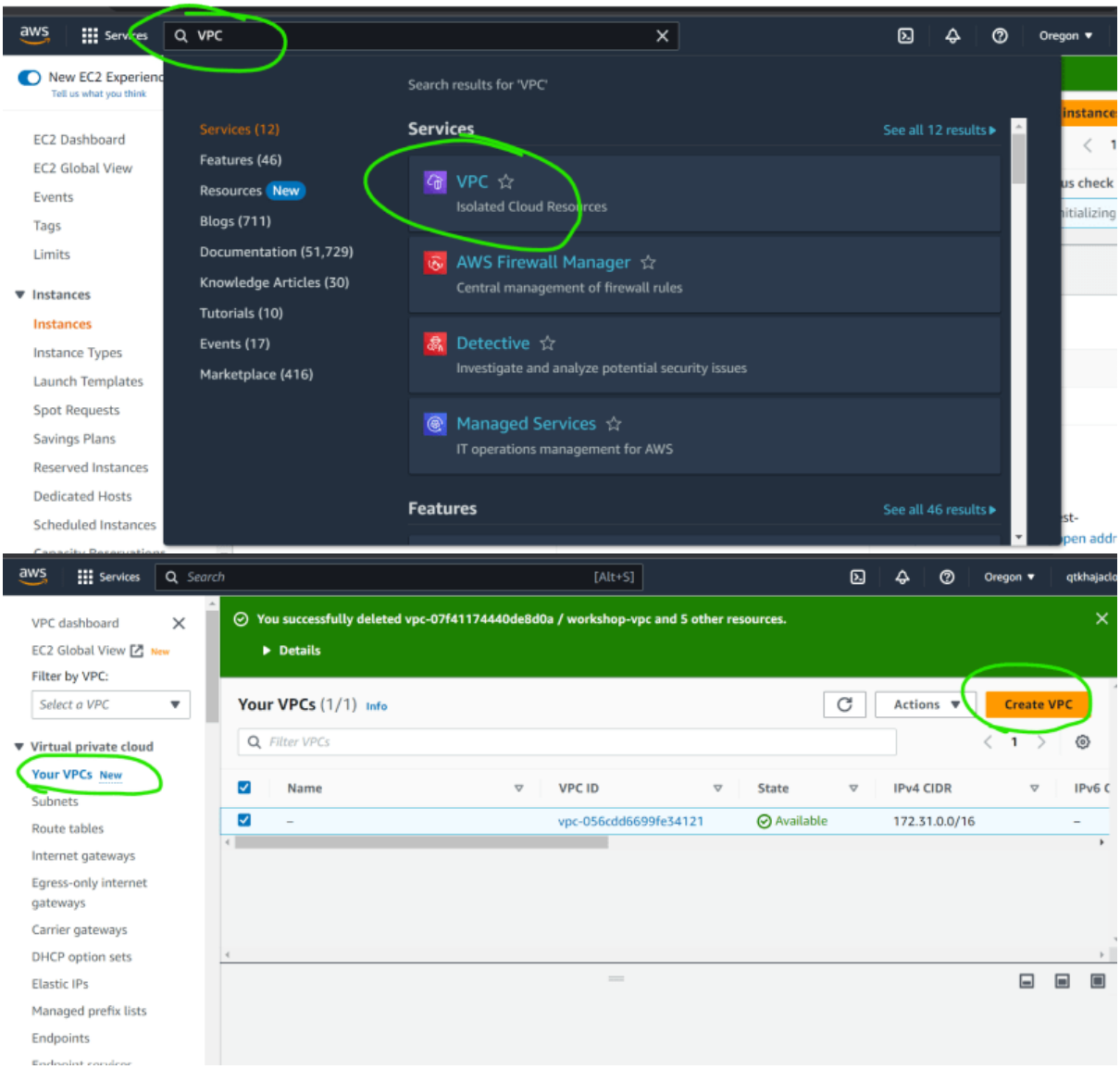
- AWS Provider argument reference https://registry.terraform.io/providers/hashicorp/aws/latest/docs#argument-reference
- Azure Provider argument reference https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs#argument-reference
- Resource: The syntax or structure of resource in terraform template is

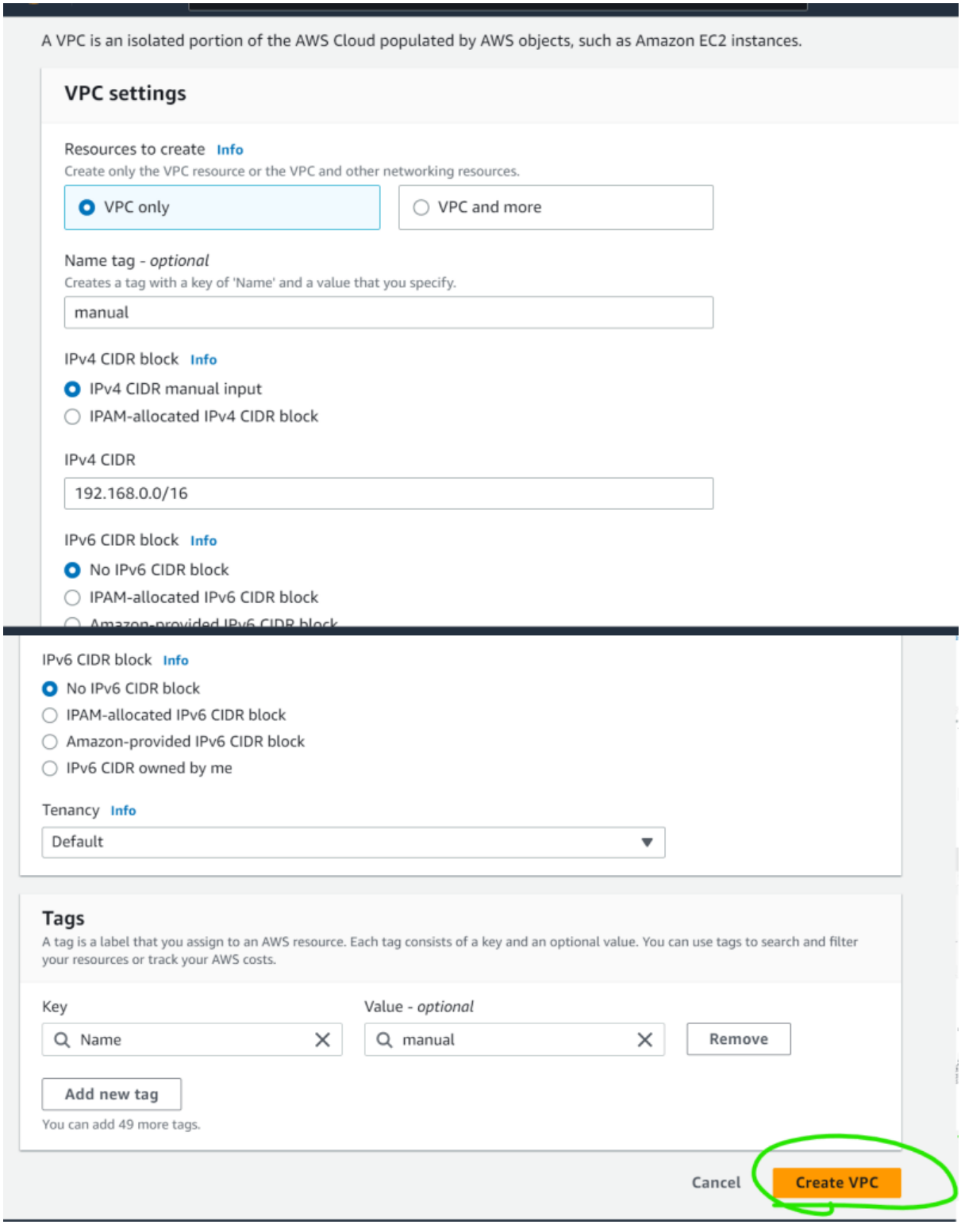
```
resource <type of resource> <name> {
  <ARGUMENT-1> = <VALUE-1>
  ..
  ..
  ..
  <ARGUMENT-N> = <VALUE-N>
}
```

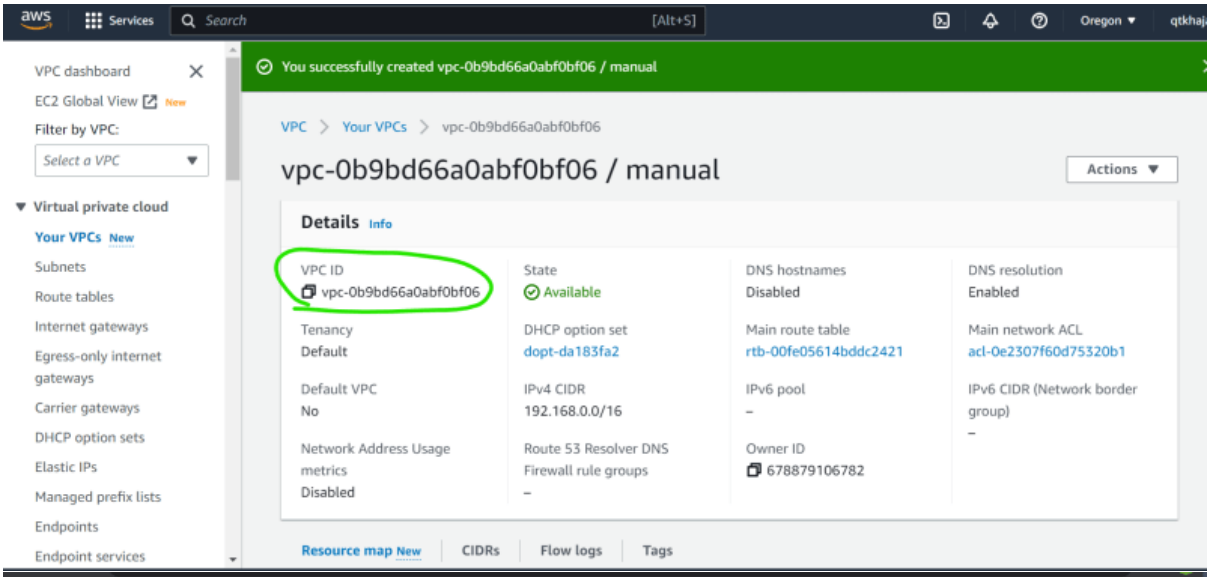
- the type of resource will be in the form of <provider>_<resource_type>

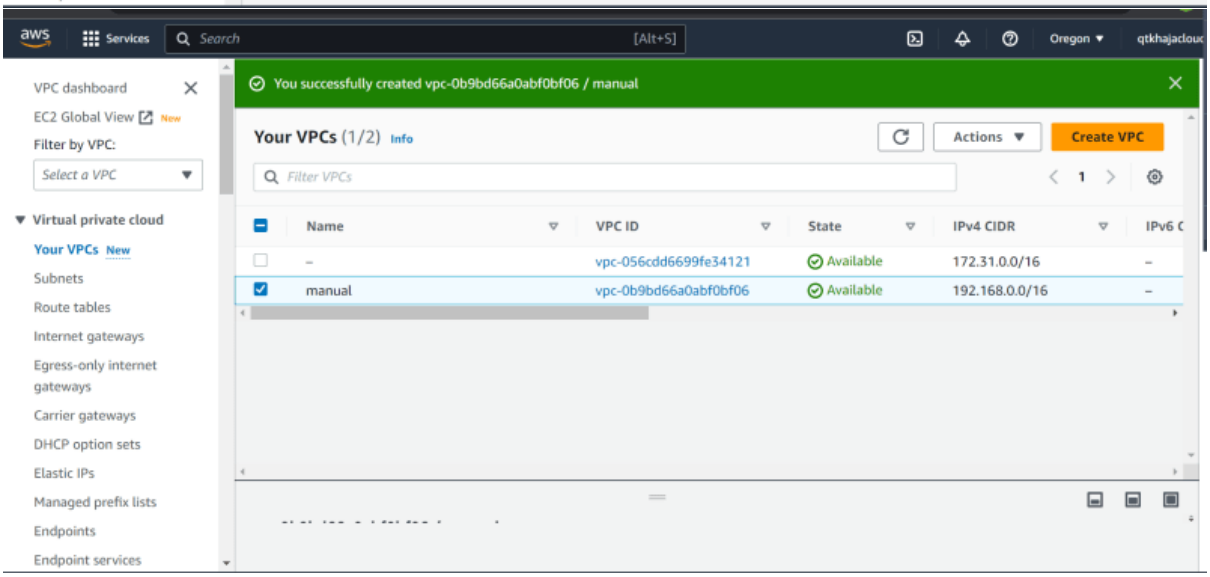
Manual Steps of VPC Creation

- Steps: Lets create a simple vpc

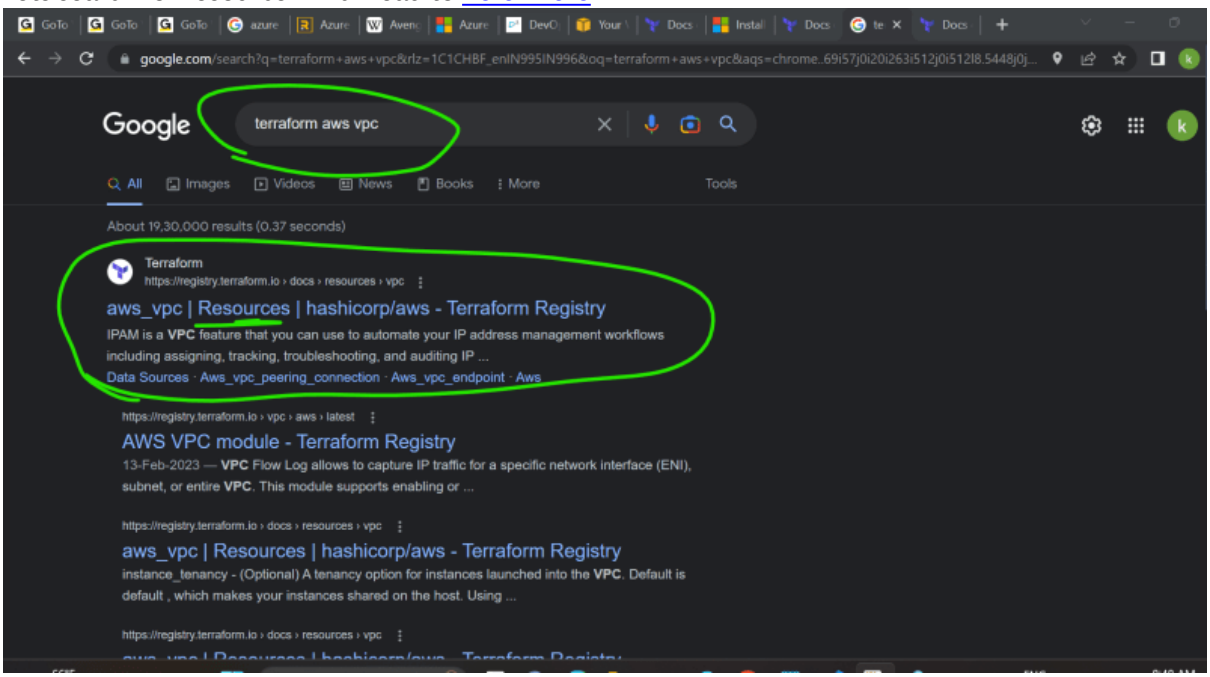



A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon EC2 instances.
VPC settings
Resources to create Info
Create only the VPC resource or the VPC and other networking resources.
VPC only
VPC and more
Name tag - optional
Creates a tag with a key of 'Name' and a value that you specify.
manual
IPv4 CIDR block Info
IPv4 CIDR manual input
IPAM-allocated IPv4 CIDR block
IPv4 CIDR
192.168.0.0/16
IPv6 CIDR block Info
No IPv6 CIDR block
IPAM-allocated IPv6 CIDR block
IPv6 CIDR block Info
No IPv6 CIDR block
IPAM-allocated IPv6 CIDR block
Amazon-provided IPv6 CIDR block
IPv6 CIDR owned by me
Tenancy Info
Default
Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Key
Value - optional
Name
manual
Remove
Add new tag
You can add 49 more tags.
Cancel
Create VPC





- Lets search for resource which lead to [Refer Here]

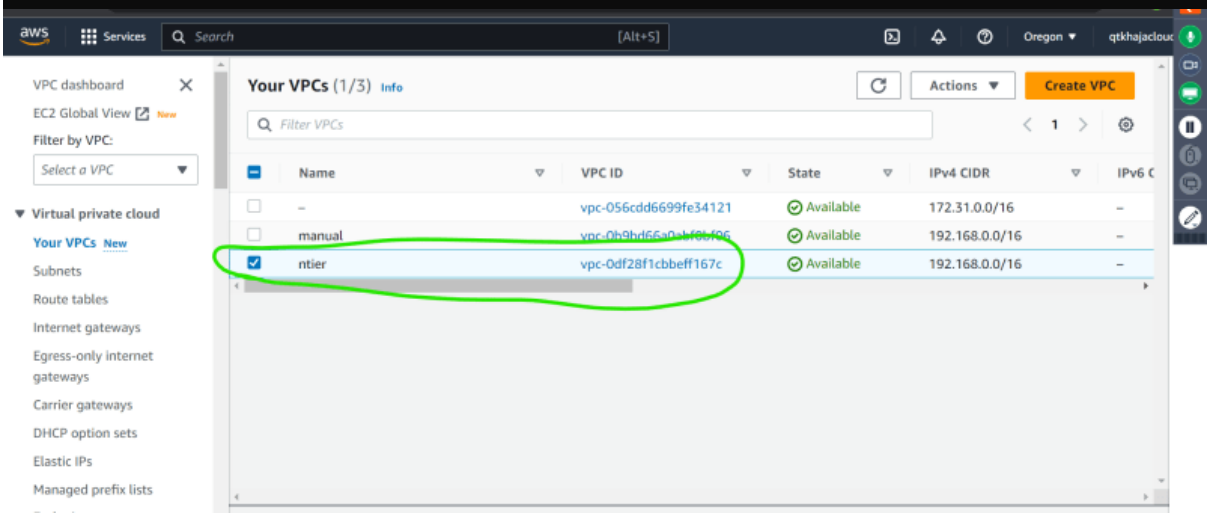


- Now look at arguments https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/vpc#argument-reference
- Create the template as shown in this changeset

```
provider "aws" {

}

resource "aws_vpc" "ntier" {
    cidr_block = "192.168.0.0/16"
    tags = {
        Name = "ntier"
    }

}
```

- Now validate and apply

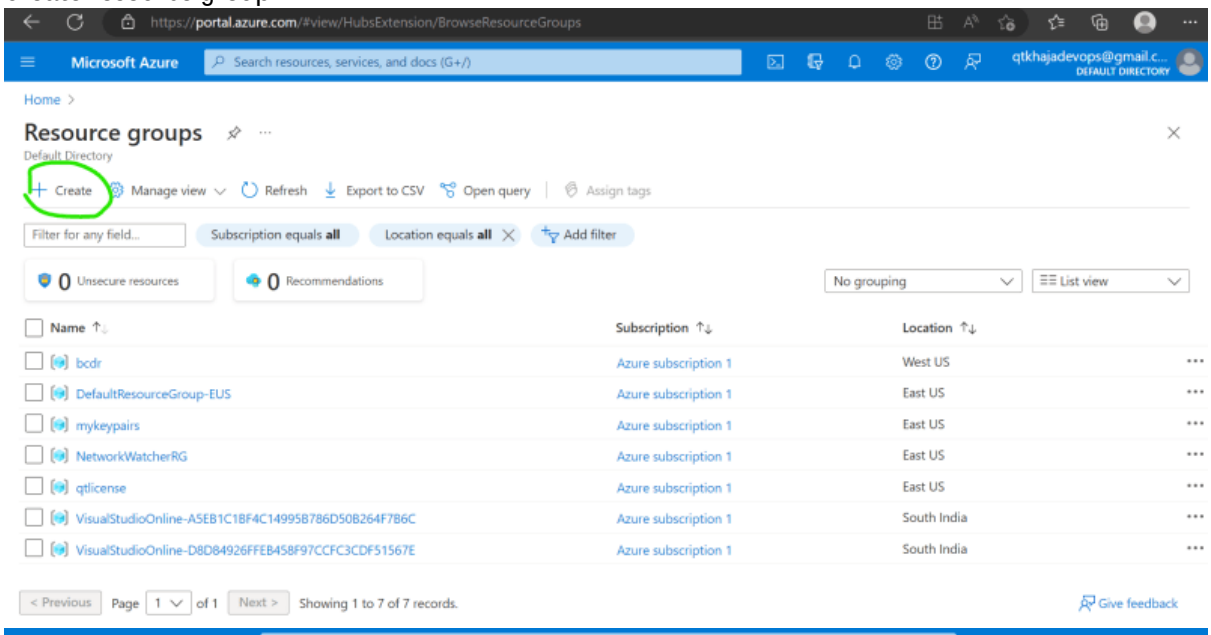
```
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "ntier"
        }
      + tags_all                             = {
          + "Name" = "ntier"
        }
    }

Plan: 1 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_vpc.ntier: Creating...
aws_vpc.ntier: Creation complete after 6s [id=vpc-0df28f1cbbeff167c]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\khajaclassroom\devops\terraform\Mar23\ntier\aws>
```

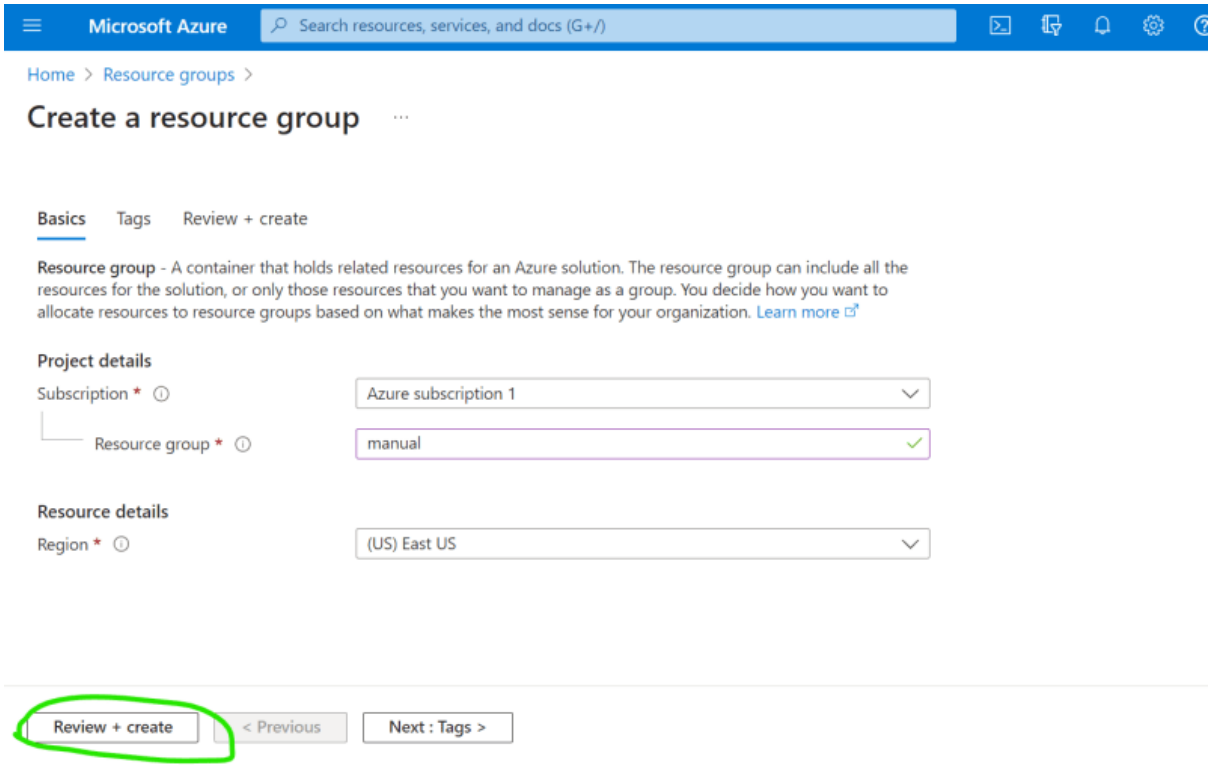


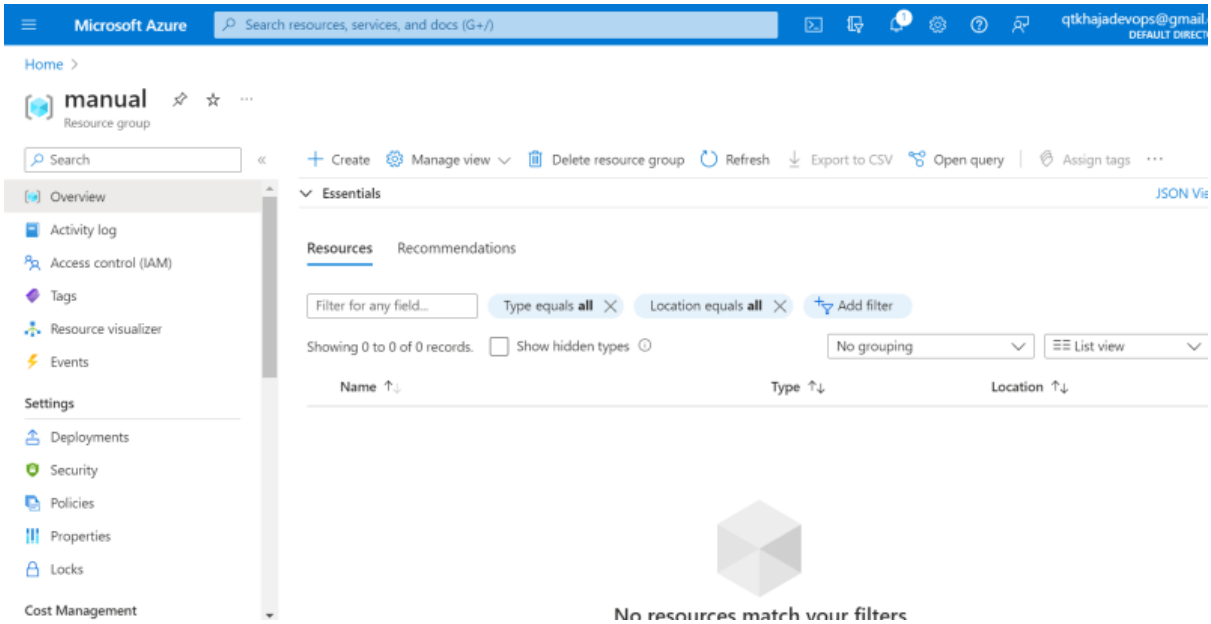
Activity: Create virtual network in Azure

- Manual Steps:

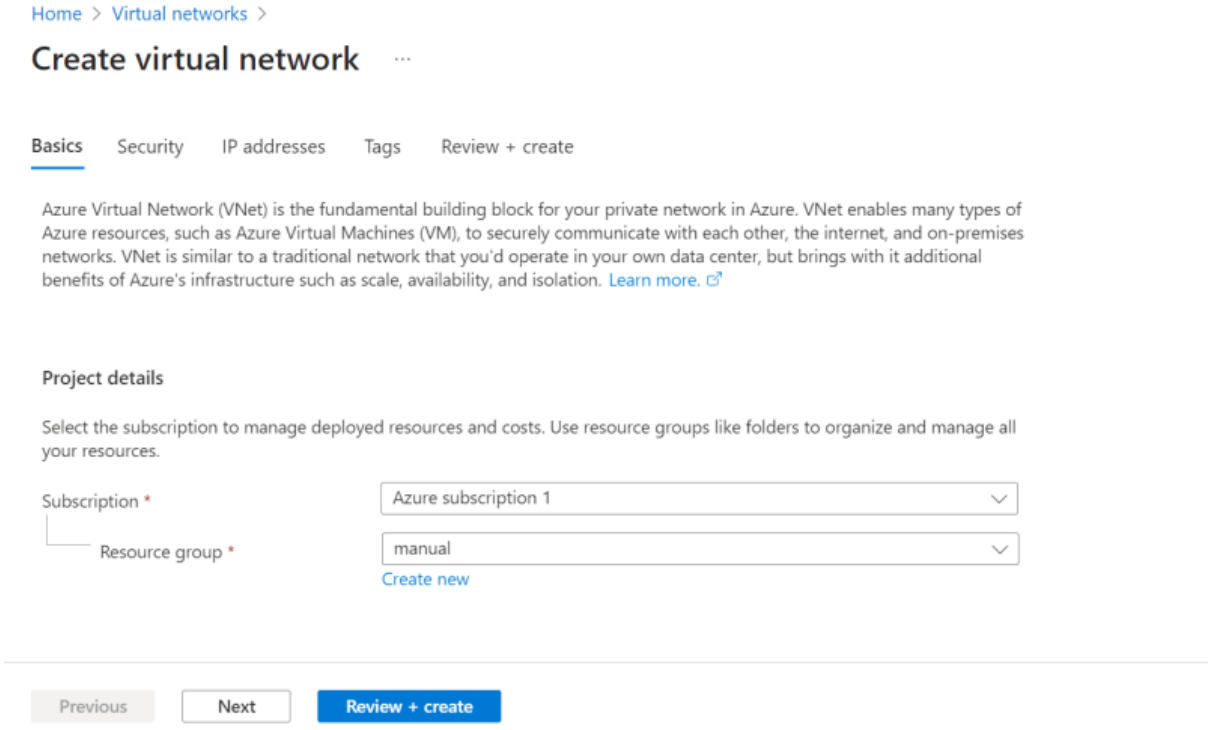
Create resource group

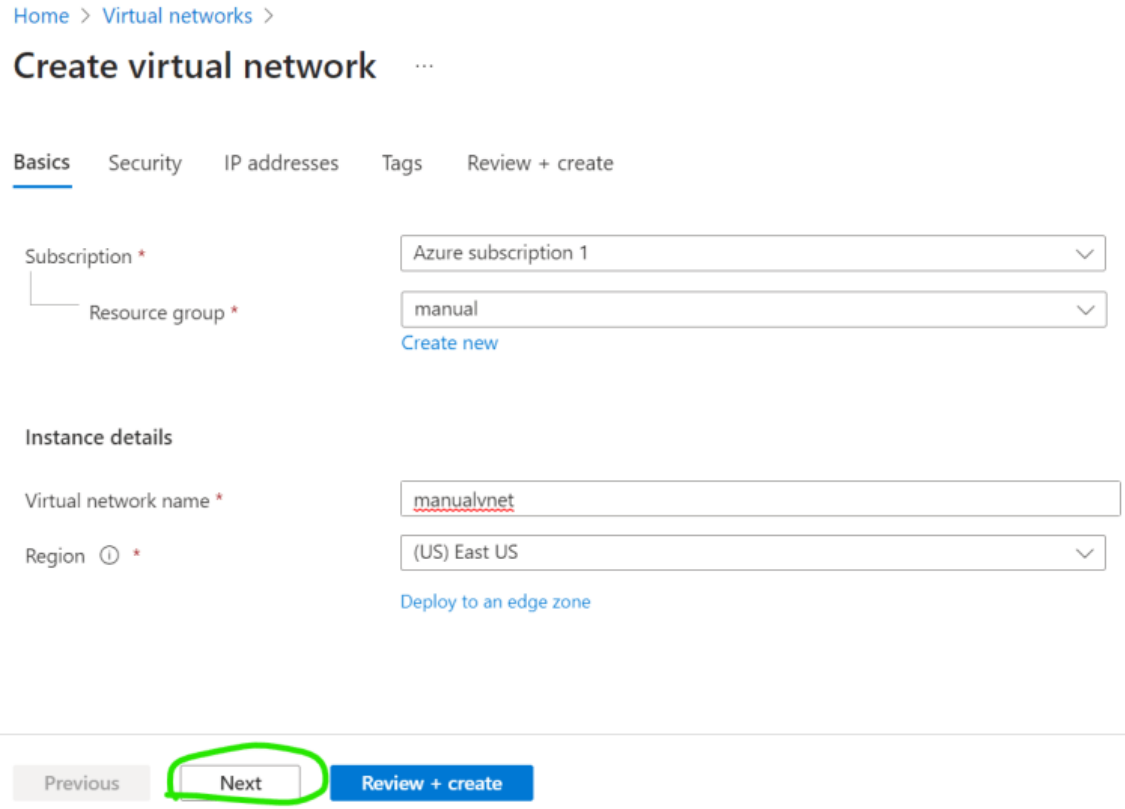






Create a VNET with cidr range 192.168.0.0/16



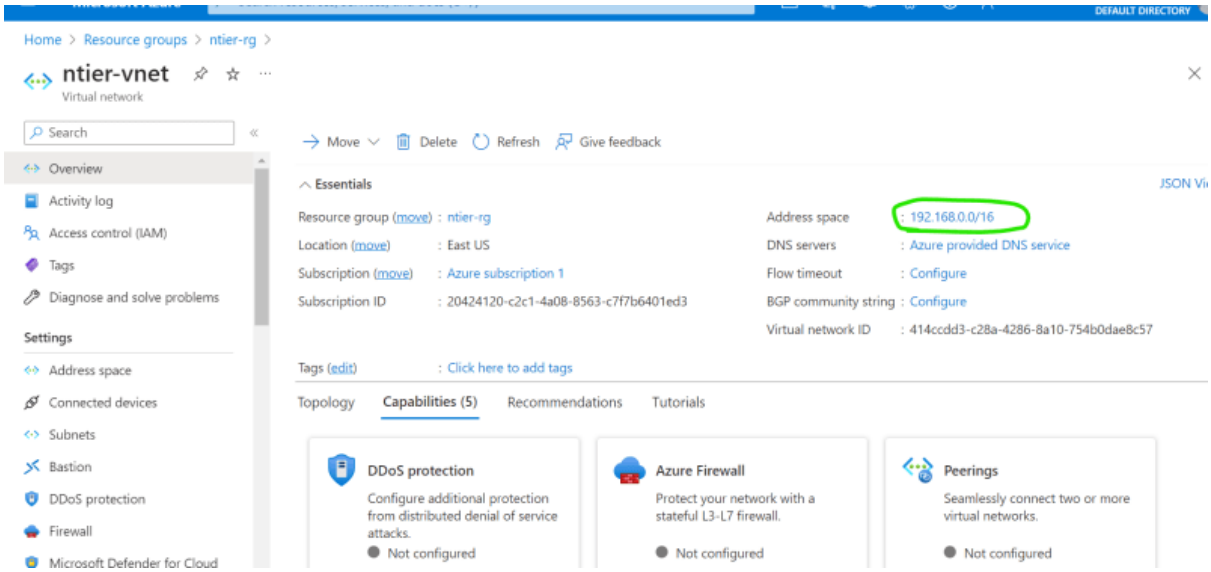


Writing template for the above in terraform

- Terraform takes the folder as input and reads all the .tf files and while validating,applying or destroying tries to treat all fo the .tf files as one file.
- To Create dependencies we can use depends on meta argument https://developer.hashicorp.com/terraform/language/meta-arguments/depends_on
- The resource name in terraform is <resource_type>.<name>

Code:

```
resource "azurerm_resource_group" "ntierrg" {
  location = "eastus"
  name     = "ntier-rg"
}

resource "azurerm_virtual_network" "ntiervnet" {
  name                = "ntier-vnet"
  resource_group_name = "ntier-rg"
  address_space       = ["192.168.0.0/16"]
  location            = "eastus"
  depends_on = [
    azurerm_resource_group.ntierrg
  ]

}
```



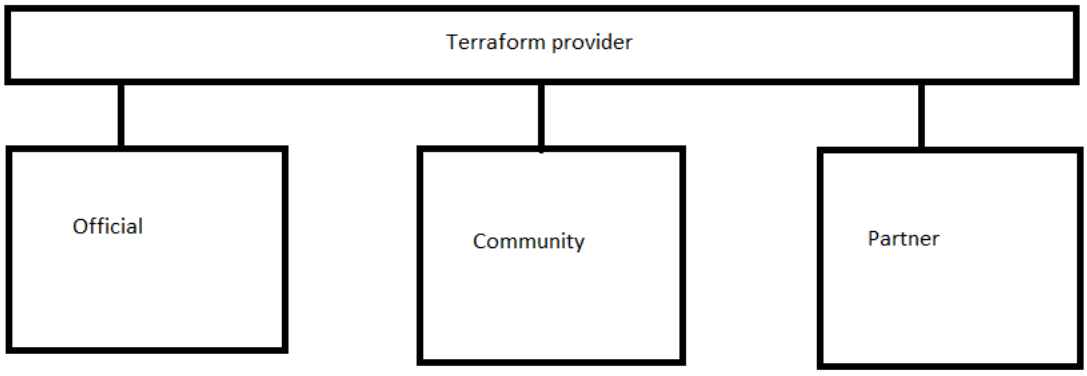
- Note:
  - The commands which we started following init, fmt, validate, apply

Focus Points

- To Work effectively with terrform templates we need to understand Hashicorp Configuration Language
- How to parametrized the template

Hashicorp Configuration Language (HCL) for Terraform

- For Specification https://github.com/hashicorp/hcl/blob/main/hclsyntax/spec.md
- Terraform release https://releases.hashicorp.com/terraform/
- Provider https://developer.hashicorp.com/terraform/language/providers



The terraform block helps in configuring the provider with version of the provider from registry https://developer.hashicorp.com/terraform/language/settings

- Specify which version of terraform you should be using use required version. To specify constraints https://developer.hashicorp.com/terraform/language/expressions/version-constraints
- main.tf file https://github.com/asquarezone/TerraformZone/commit/ccf6a7c9dd7c97a607d920323e16bc5b3a8339d3

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    aws = {
        source = "hashicorp/aws"
        version = ">= 4.47.0"
    }
  }
}


provider "aws" {

}
```

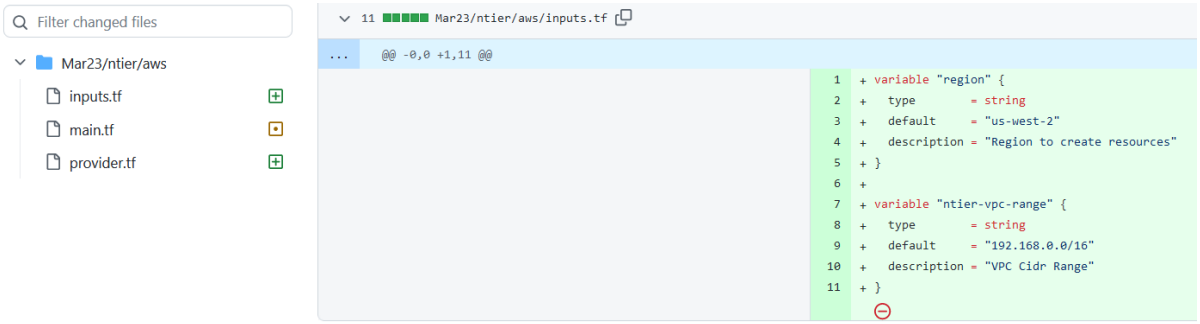
Note: By default, terraform pick latest configuration.

How to read version in terraform 1.4.5[<majorVersion><minorVersion><patchSet>]

Parametrizing Terraform

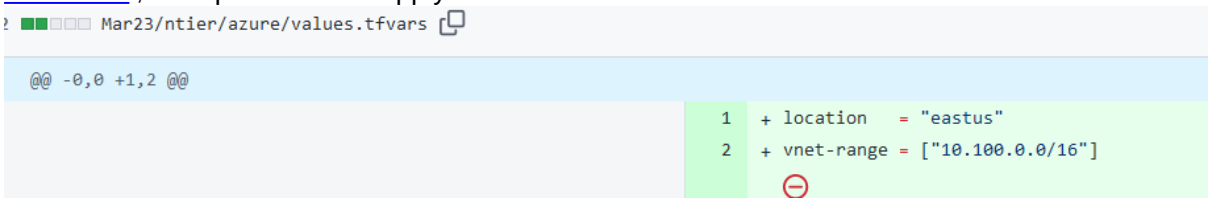
Input Variables

- https://developer.hashicorp.com/terraform/language/values/variables for input variables official docs
- For inputs terraform supports the following types
    - number
    - string
    - boolean
    - list()
    - set()
    - map()
    - object({ = , ... })
    - tuple([, ...])
- To pass variables while executing commands we have two options
    - -var
    - -var-file
- using -var https://developer.hashicorp.com/terraform/language/values/variables#variables-on-the-command-line



Command: terraform apply -var "region=ap-south-2" -var "ntier-vpc-range=10.10.0.0/16"

- using variable definitions https://developer.hashicorp.com/terraform/language/values/variables#variable-definitions-tfvars-files , example terraform apply -var-file values.tfvars



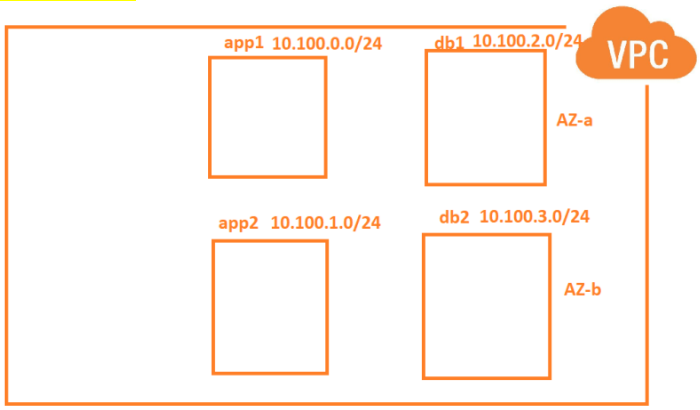
- Instead of creating multiple var file we can create 1 values.tfvars where we can define all the variable in once like above.

- Terraform fmt: The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability.

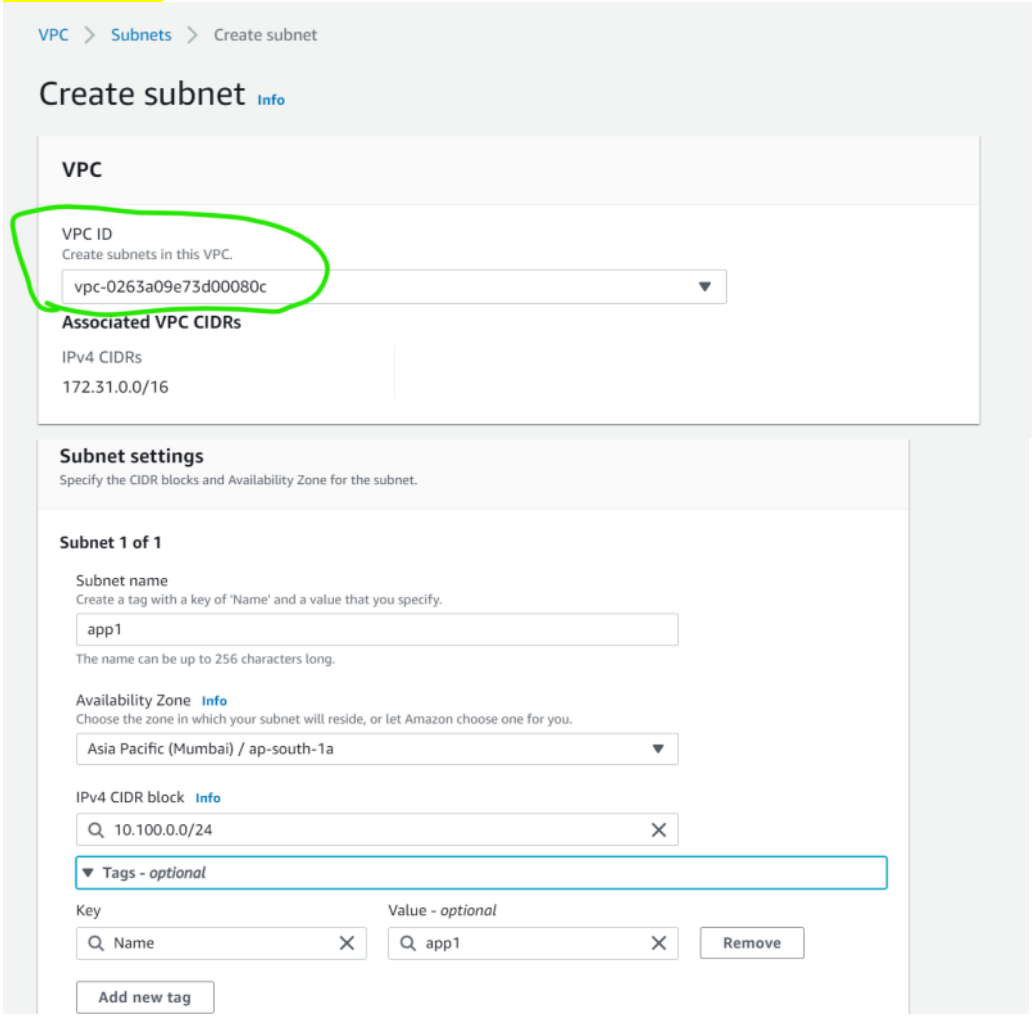
- Terraform console: This command provides an interactive command-line console for evaluating and experimenting with expressions.

- Region and AZ: Each AWS Region is a separate geographic area. Each AWS Region has multiple, isolated locations known as Availability Zones.

- Expectation



- Manual steps



- To access outputs i.e. attributes of a resource syntax is

  <resoure_type>.<name>.<atrribute-name>

Mar23/ntier/aws/values.tfvars

```
@@ -0,0 +1,6 @@
+ region          = "us-west-2"
+ ntier-vpc-range = "10.100.0.0/16"
+ ntier-app1-cidr = "10.100.0.0/24"
+ ntier-app2-cidr = "10.100.1.0/24"
+ ntier-db1-cidr  = "10.100.2.0/24"
+ ntier-db2-cidr  = "10.100.3.0/24"
```

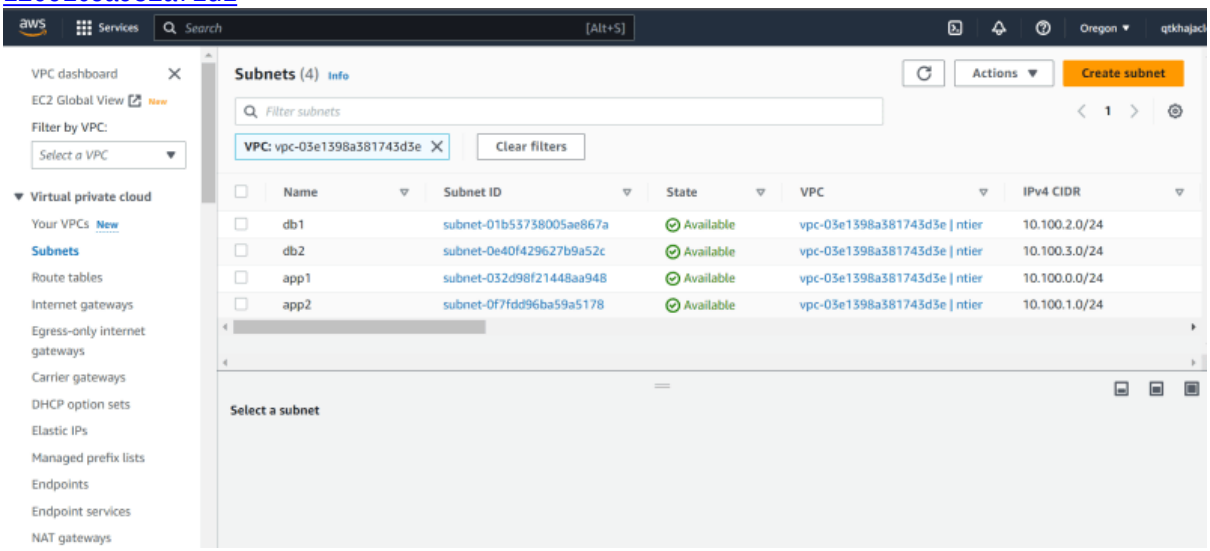
22 Mar23/ntier/aws/inputs.tf

```
@@ -8,4 +8,24 @@ variable "ntier-vpc-range" {
    type        = string
    default     = "192.168.0.0/16"
    description = "VPC Cidr Range"
- }
+ }
+
+ variable "ntier-app1-cidr" {
+   type    = string
+   default = "192.168.0.0/24"
+ }
+
+ variable "ntier-app2-cidr" {
+   type    = string
+   default = "192.168.1.0/24"
+ }
+
+ variable "ntier-db1-cidr" {
+   type    = string
+   default = "192.168.2.0/24"
+ }
+
+ variable "ntier-db2-cidr" {
+   type    = string
+   default = "192.168.3.0/24"
+ }
```

Mar23/ntier/aws/main.tf

```
@@ -1,9 +1,58 @@
resource "aws_vpc" "ntier" {
  cidr_block = var.ntier-vpc-range
  tags = {
    Name = "ntier"
  }
}

resource "aws_vpc" "ntier" {
  cidr_block = var.ntier-vpc-range
  tags = {
    Name = "ntier"
  }
}

resource "aws_subnet" "app1" {
  cidr_block        = var.ntier-app1-cidr
  availability_zone = "${var.region}a"
  vpc_id            = aws_vpc.ntier.id #implicit dependency
  tags = {
    Name = "app1"
  }
  depends_on = [
    aws_vpc.ntier
  ]

}

resource "aws_subnet" "app2" {
  cidr_block        = var.ntier-app2-cidr
  availability_zone = "${var.region}b"
  vpc_id            = aws_vpc.ntier.id #implicit dependency
  tags = {
    Name = "app2"
  }
  depends_on = [
    aws_vpc.ntier
  ]
}

  tags = {
    Name = "db1"
  }
  depends_on = [
    aws_vpc.ntier
  ]
}

resource "aws_subnet" "db2" {
  cidr_block        = var.ntier-db2-cidr
  availability_zone = "${var.region}b"
  vpc_id            = aws_vpc.ntier.id #implicit dependency
  tags = {
    Name = "db2"
  }
  depends_on = [
    aws_vpc.ntier
  ]
}
```

Link:https://github.com/asquarezone/TerraformZone/commit/02832c0fdfa13de630a841d512091cea952a71d1



Meta argument for terraform: https://developer.hashicorp.com/terraform/language/meta-arguments/

Count> for creating multiple resource
https://developer.hashicorp.com/terraform/language/meta-arguments/count

Depends on> to specify dependency
https://developer.hashicorp.com/terraform/language/meta-arguments/depends_on

```
resource "aws_instance" "server" {
  count = 4 # create four similar EC2 instances

  ami           = "ami-a1b2c3d4"
  instance_type = "t2.micro"

  tags = {
    Name = "Server ${count.index}"
  }
}
```

For each> when you want to run thing for all
https://developer.hashicorp.com/terraform/language/meta-arguments/for_each

```
resource "azurerm_resource_group" "rg" {
  for_each = {
    a_group = "eastus"
    another_group = "westus2"
  }
  name     = each.key
  location = each.value
}
```

Let’s apply the count for subnet creation
https://github.com/asquarezone/TerraformZone/commit/148ceb1385c2b4f08537d4579dc2a3aa98168fbf for changes

23 Mar23/ntier/aws/inputs.tf

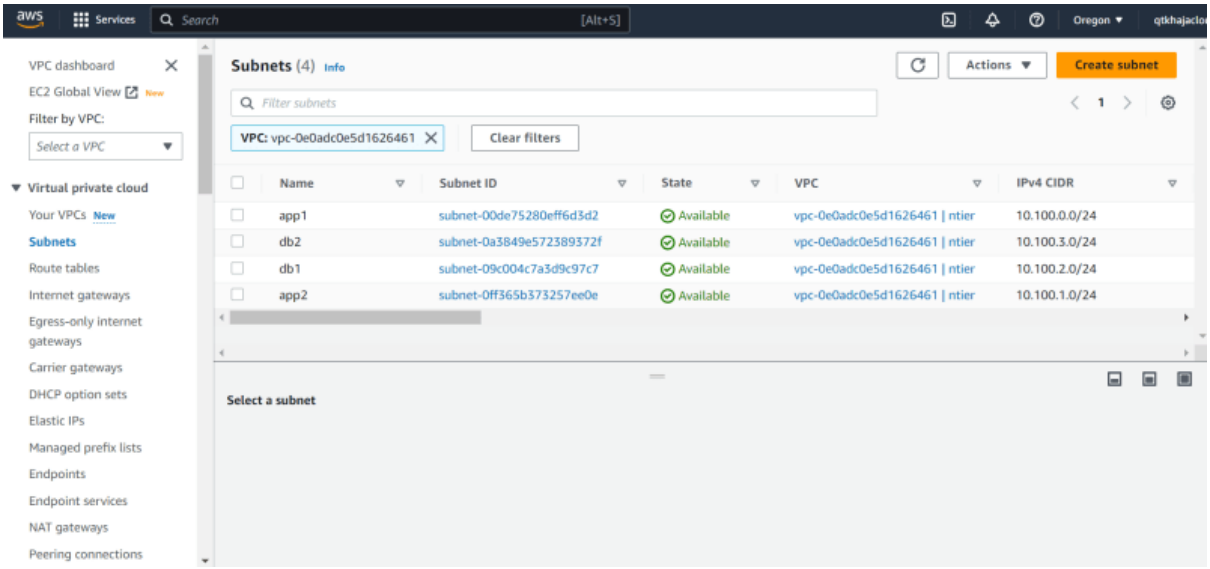
```
@@ -10,22 +10,19 @@ variable "ntier-vpc-range" {
  description = "VPC Cidr Range"
}

- variable "ntier-app1-cidr" {
-   type    = string
-   default = "192.168.0.0/24"
+ variable "ntier-subnet-cidrs" {
+   type    = list(string)
+   default = ["192.168.0.0/24", "192.168.1.0/24", "192.168.2.0/24", "192.168.3.0/24"]
}

- variable "ntier-app2-cidr" {
-   type    = string
-   default = "192.168.1.0/24"
- }

- variable "ntier-db1-cidr" {
-   type    = string
-   default = "192.168.2.0/24"
}

- variable "ntier-db2-cidr" {
-   type    = string
-   default = "192.168.3.0/24"
+ variable "ntier-subnet-azs" {
+   type    = list(string)
+   default = ["a", "b", "a", "b"]
+ }
+ variable "ntier-subnet-names" {
+   type    = list(string)
+   default = ["app1", "app2", "db1", "db2"]
}
```

9 Mar23/ntier/aws/values.tfvars

```
@@ -1,6 +1,3 @@
- region          = "us-west-2"
- ntier-vpc-range = "10.100.0.0/16"
- ntier-app1-cidr = "10.100.0.0/24"
- ntier-app2-cidr = "10.100.1.0/24"
- ntier-db1-cidr  = "10.100.2.0/24"
- ntier-db2-cidr  = "10.100.3.0/24"
+ region             = "us-west-2"
+ ntier-vpc-range    = "10.100.0.0/16"
+ ntier-subnet-cidrs = ["10.100.0.0/24", "10.100.1.0/24", "10.100.2.0/24", "10.100.3.0/24"]
```

44 Mar23/ntier/aws/main.tf

```
@@ -8,51 +8,17 @@ resource "aws_vpc" "ntier" {

}

- resource "aws_subnet" "app1" {
-   cidr_block        = var.ntier-app1-cidr
-   availability_zone = "${var.region}a"
+ resource "aws_subnet" "subnets" {
+   count             = 4
+   cidr_block        = var.ntier-subnet-cidrs[count.index]
+   availability_zone = "${var.region}${var.ntier-subnet-azs[count.index]}"
    vpc_id            = aws_vpc.ntier.id #implicit dependency
-   tags = {
-     Name = "app1"
-   }
    depends_on = [
      aws_vpc.ntier
    ]
```



- Let's start using terraform functions to make further improvements using terraform functions https://developer.hashicorp.com/terraform/language/functions
- Terraform functions: The Terraform language includes a number of built-in functions that you can call from within expressions to transform and combine values.
- Let's replace static count of 4 with length function https://developer.hashicorp.com/terraform/language/functions/length
- Changeset looks like:

Code:https://github.com/asquarezone/TerraformZone/commit/bc031c68f1be1291e05639df3dc544542da936e8

Terraform plan: The terraform plan command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure.

AWS

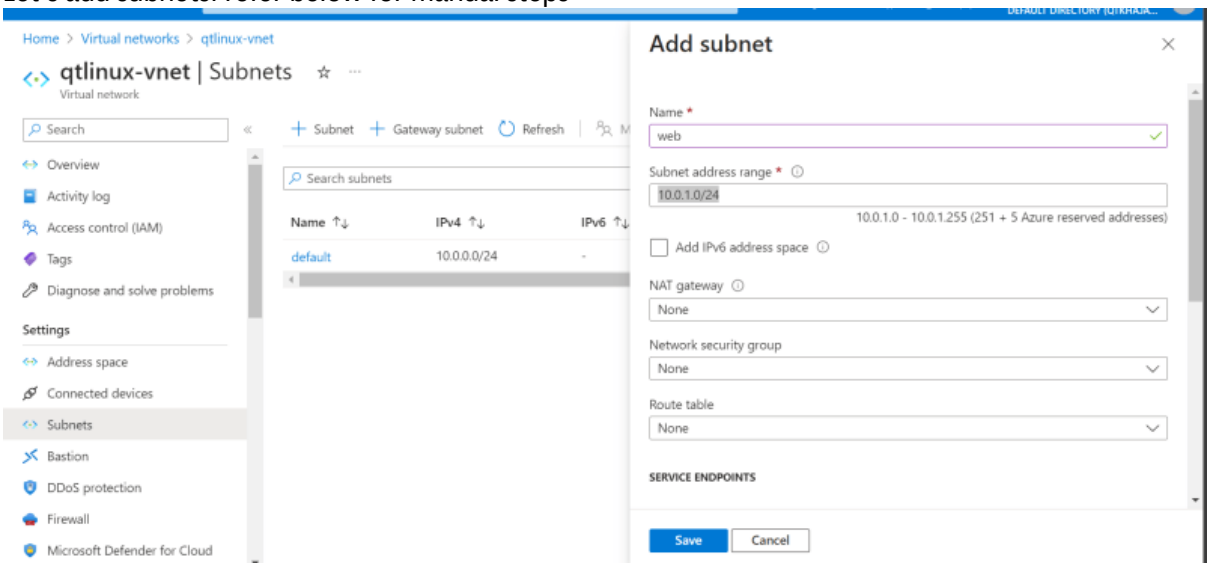
- Let’s generate subnet cidr range

```
PS C:\khajaclassroom\devops\terraform\Mar23\ntier\aws> terraform console
> cidrsubnet("192.168.0.0/16", 8, 0)
"192.168.0.0/24"
> cidrsubnet("192.168.0.0/16", 8,1)
"192.168.1.0/24"
```

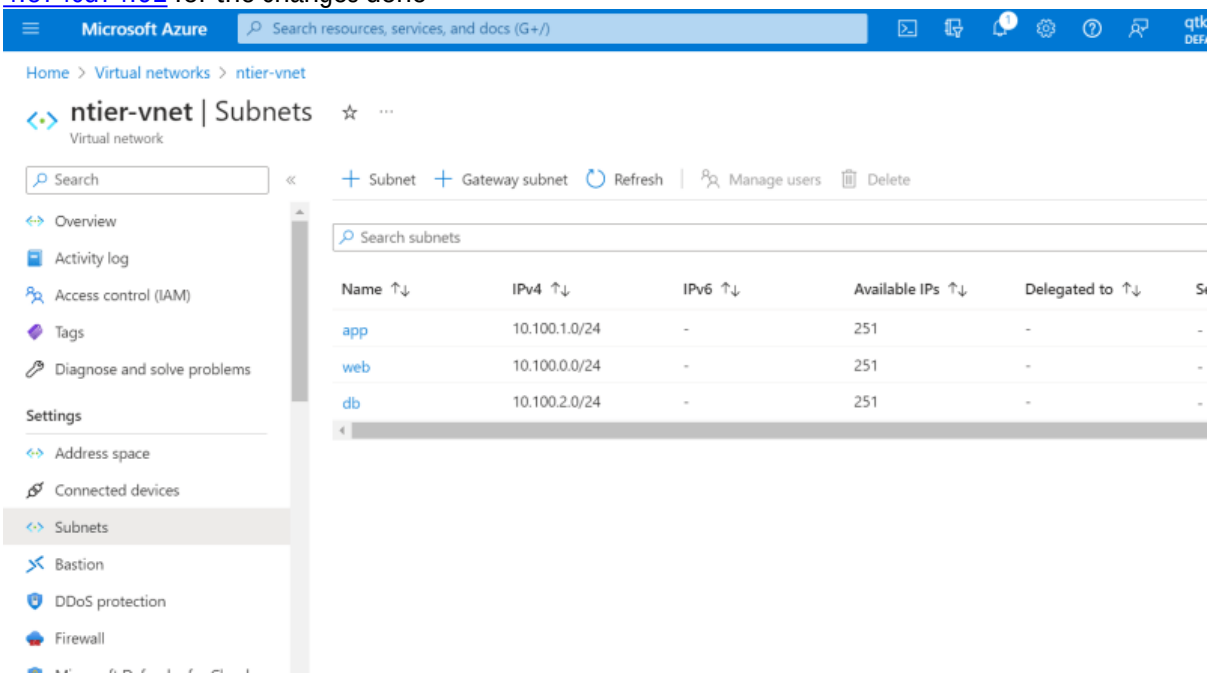
- https://github.com/asquarezone/TerraformZone/commit/ac2ffd5be12da24071bbab271bb341daade0ec7d for the changes done to use cidrsubnet function and object input type and
- Terraform plan is generated whenever we apply

Azure

- Let’s add subnets. refer below for manual steps

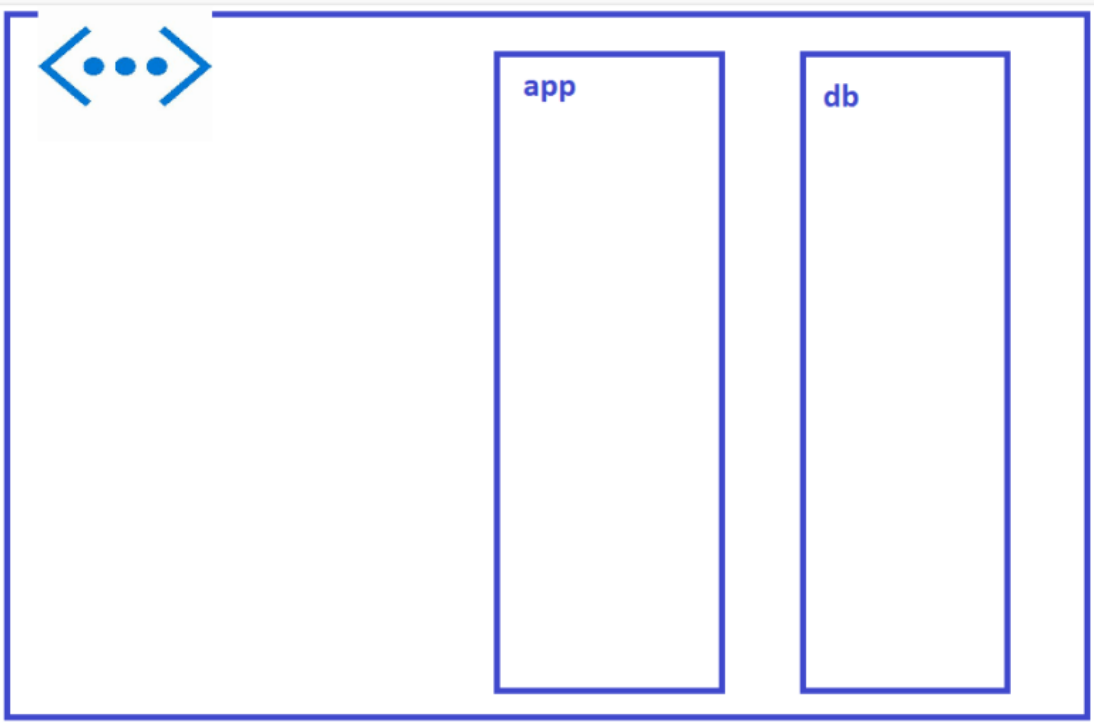


- https://github.com/asquarezone/TerraformZone/commit/d4f087ab33a45b1cf9520dc999804fe74cd74f91 for the changes done

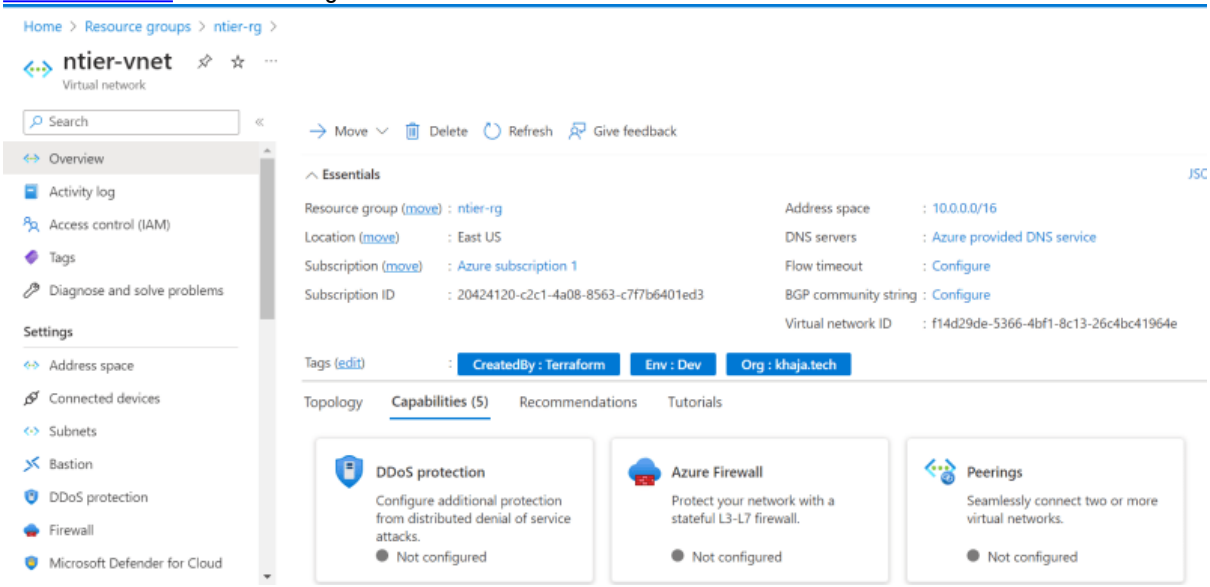


Ntier on Azure

- We need to create the following network



- replaced hardcode names for resource group and vnet with variable https://github.com/asquarezone/TerraformZone/commit/9959f23b5c4ee8b9fb993748803c3c872b1af061 for the changes



Let's try to Create Azure SQL Database

- Manual steps: https://learn.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?view=azuresql&tabs=azure-portal
- We need to Create SQL Server and then database
- https://github.com/asquarezone/TerraformZone/commit/5f2bf9efefacefd6712deb72c2624b380ba2970f for the changes done to create sql server
- Execution and Database creation are pending

How to manage multiple environments with Terraform

We create input.tf file where we define variable with default values, and we can create environment specific file like dev.tfvars , qa.tfvars and prod.tfvars to overwide existing variable.

What happen when you run the terraform destroy/apply command?

In terraform we have configuration present in 3 places-
1) Cloud provider, 2) statefile and 3) main.tf

At first terraform destroy command run internally terraform refresh command which get latest data in statefile. Later on, it compares statefile and main file configuration, and generate plan.

What are the 3 places where tf configuration is present:

1) Cloud provider configuration,
2) statefile[local configuration]
3) main.tf file.

What 3 command internally run when we run tf apply or destroy command?

Terraform refresh, terraform plan and terraform create.

What happen when we run tf plan command?

Terraform plan command first run terraform refresh to get latest infrastructure details from cloud provider, later on it compare changes with main.tf file and generate plan what to do.

What is –auto-approve means?

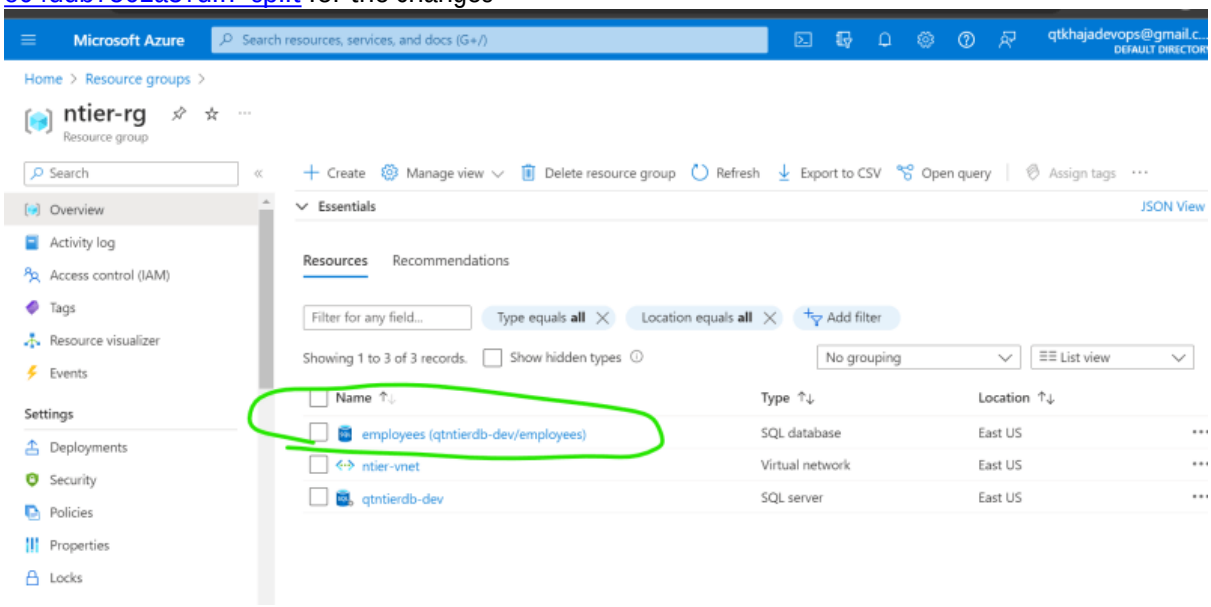
It means terraform directly execute the plan.

Configuration Drift:

- Difference between actual and desired state
- In Terraform plan represents drift

Creating Database in Azure using Terraform

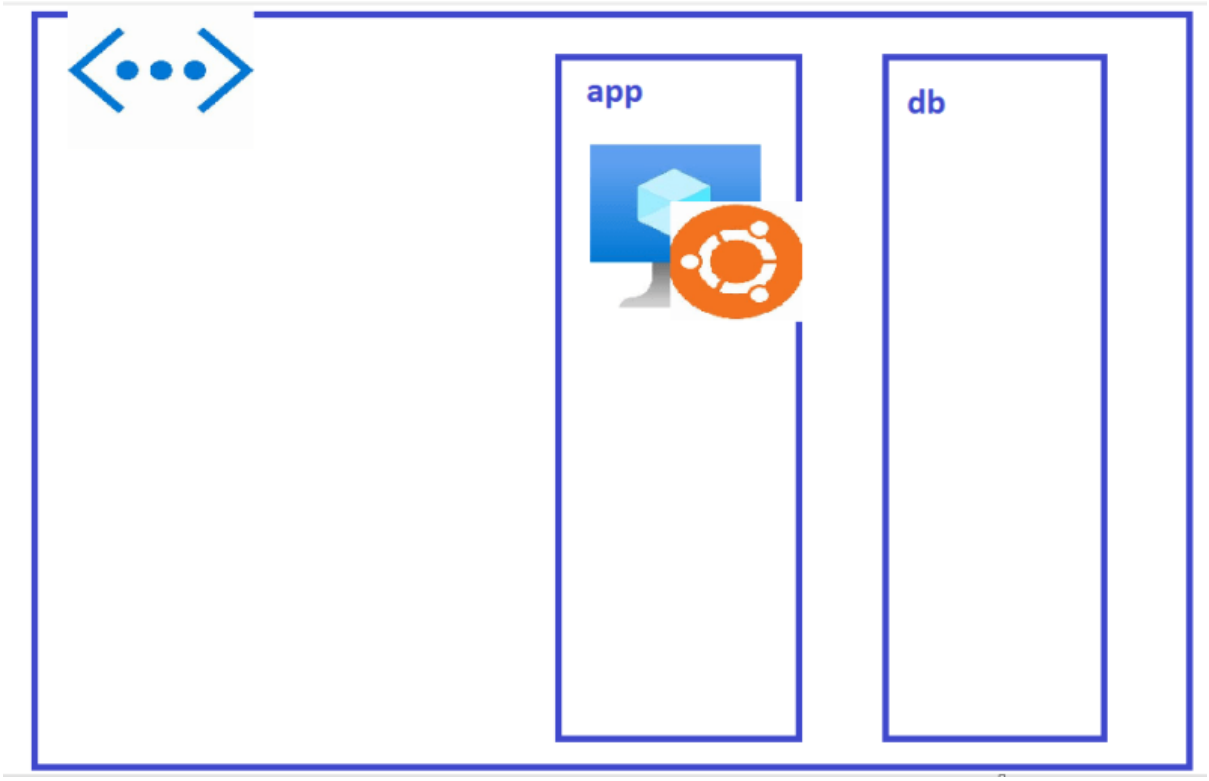
- We have added the resource to create sql server on azure and applied the template. https://github.com/asquarezone/TerraformZone/commit/998a87ab478a40a367f9d02c89bee04ddb7561a3?diff=split for the changes



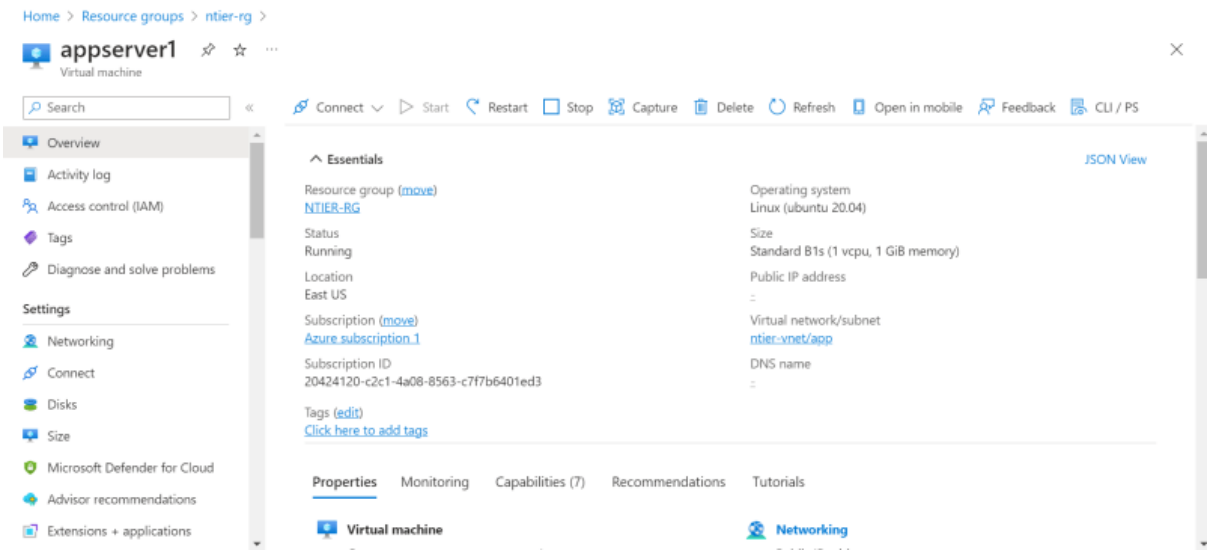
Terraform graph: command creates dependency graphs in dot format which can be visualized in Graphviz.[https://dreampuf.github.io/GraphvizOnline/]

Create VMs in Azure using terraform

- Overview of our goal



- https://learn.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal?tabs=ubuntu for manual steps
- https://github.com/asquarezone/TerraformZone/commit/d2cbaceb9595ef6fc884ed1f9c94f22737fb1078 for the changes.



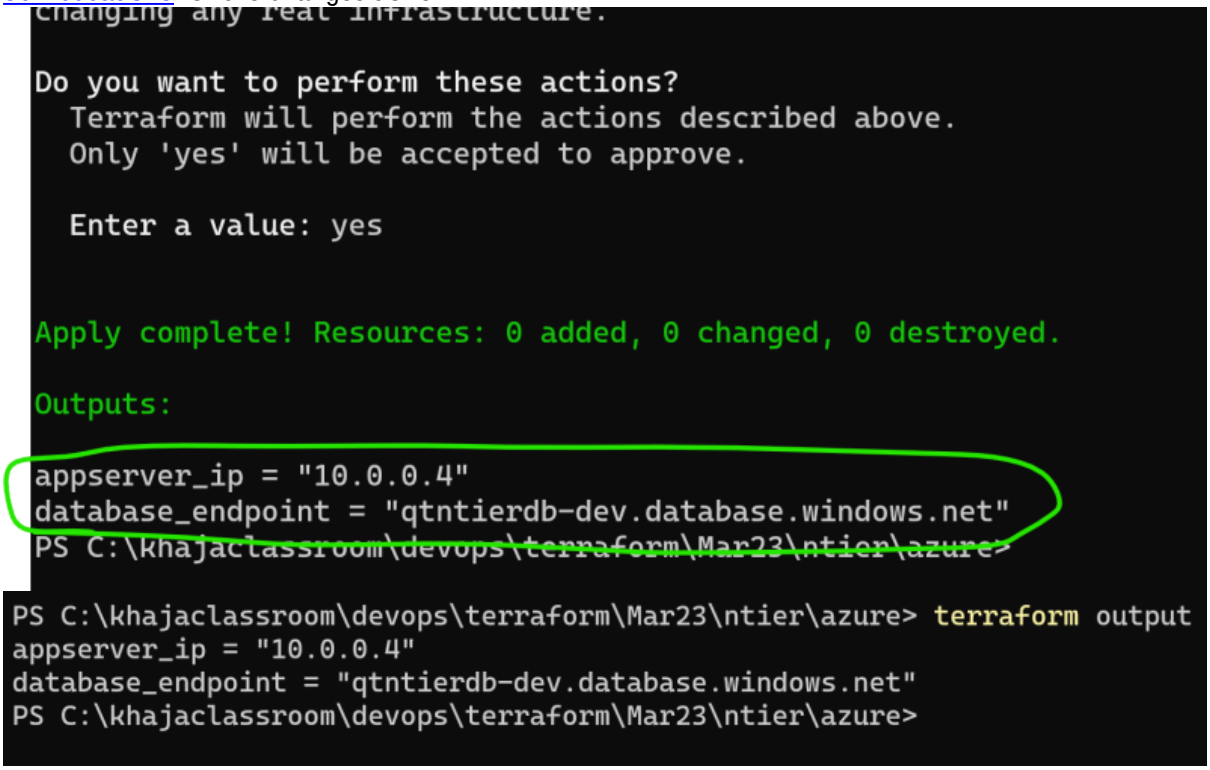
- We have created a vm without public ip and database connectivity between vm and sql is using internet. We will work on this actvities in next session
- Improvements:
    - Try to parametrize using variables and avoid creating too many variables by using object structure. https://github.com/asquarezone/TerraformZone/commit/3a9b434ae04e2f124b97271755424d765acec857 for changes

Terraform Datasources

- Till now we have used terraform to create resources in provider. Terraform can also query the provider for various information
- Refer Here for official docs

Terraform outputs

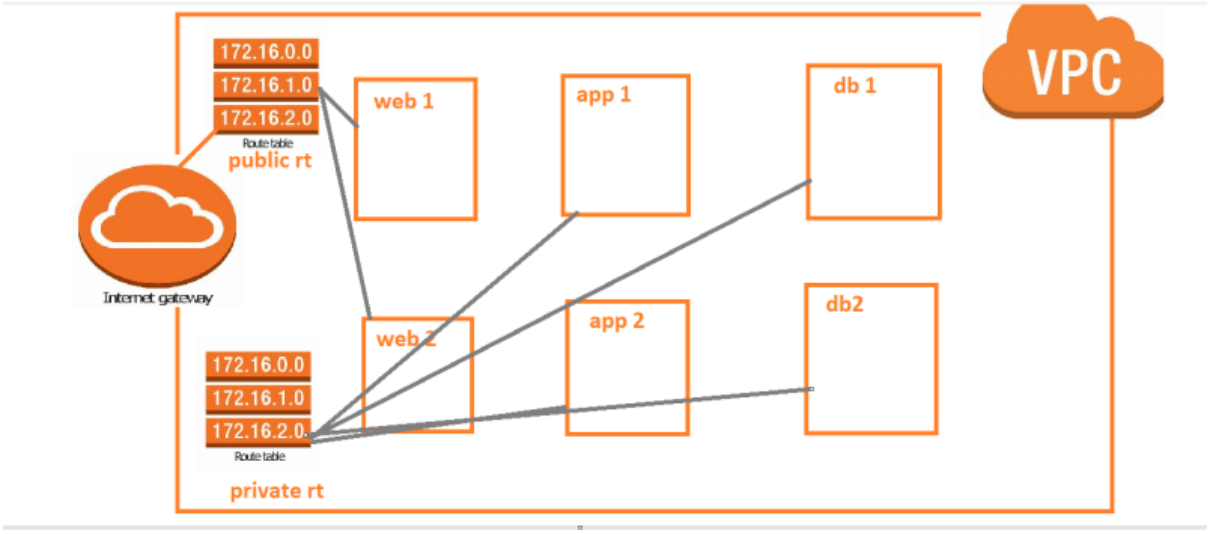
- Terraform output is result which would be shown in the stdout
- https://developer.hashicorp.com/terraform/language/data-sources for official docs
- Lets display private ip address and database endpoint as outputs https://github.com/asquarezone/TerraformZone/commit/3c5879a3b0cb89c1b7d9afa86730564fbcbad315 for the changes done



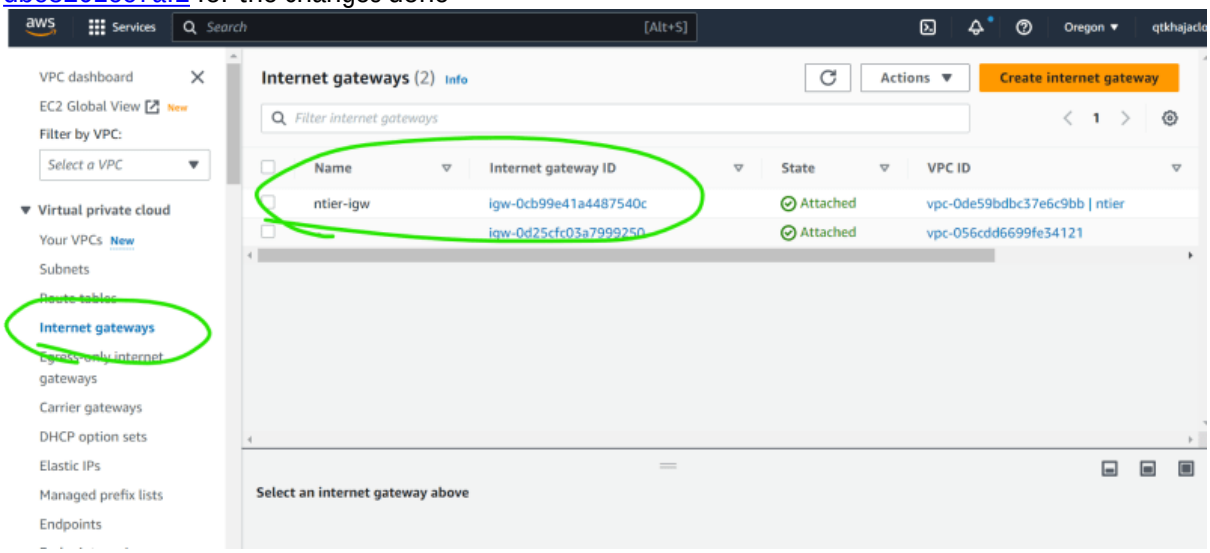
Example: Support I want to find latest AMI, then I would be using data source.

Making subnets public and private

- Overview



- Creating internet gateway and attach to vpc. https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/internet_gateway for resource
- https://github.com/asquarezone/TerraformZone/commit/09a06eb0e7ed5e2ad05278ab88edbe8261ee7af2 for the changes done

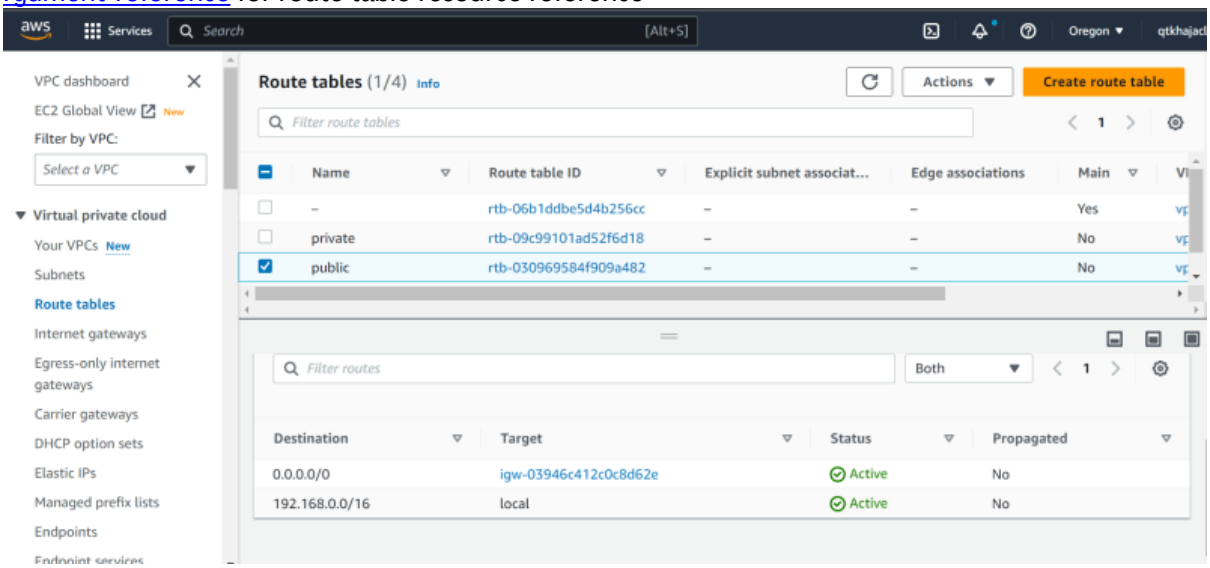


- Now let's create two route tables

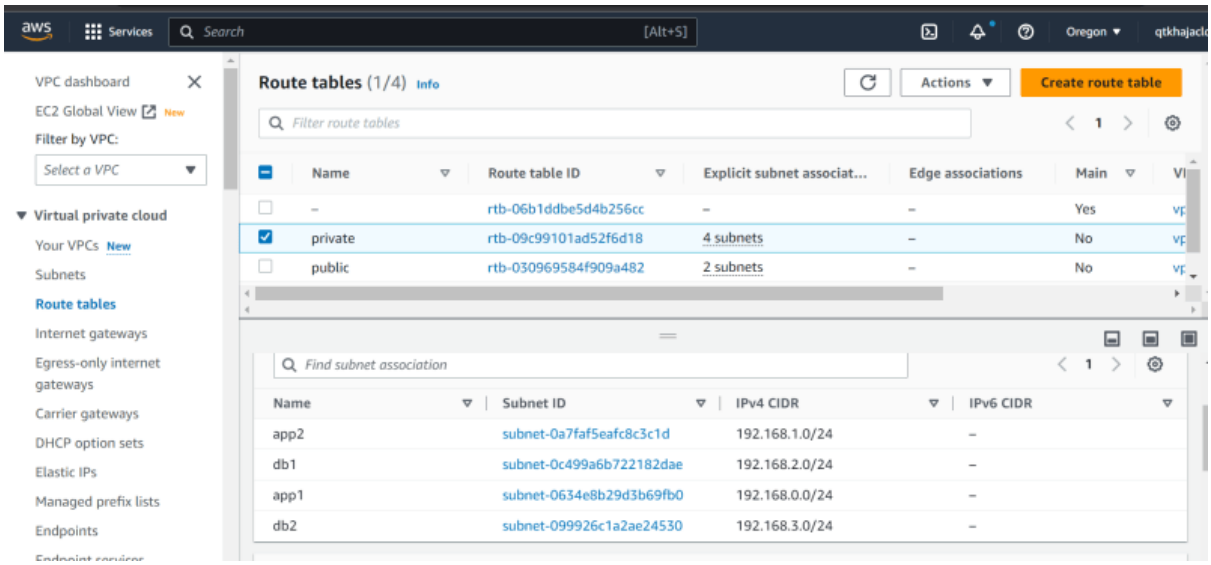
- So far, we have created vpc with 6 subnets and attached internet gateway
- Now let's create two route tables public and private
- Terraform has locals where we can define the value for usage within template https://developer.hashicorp.com/terraform/language/values/locals

Create Route tables

- https://github.com/asquarezone/TerraformZone/commit/a26a0a56bbb194f8aebc1dbee0ff8143c7d842ad for the changeset to add route tables and https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table#argument-reference for route table resource reference



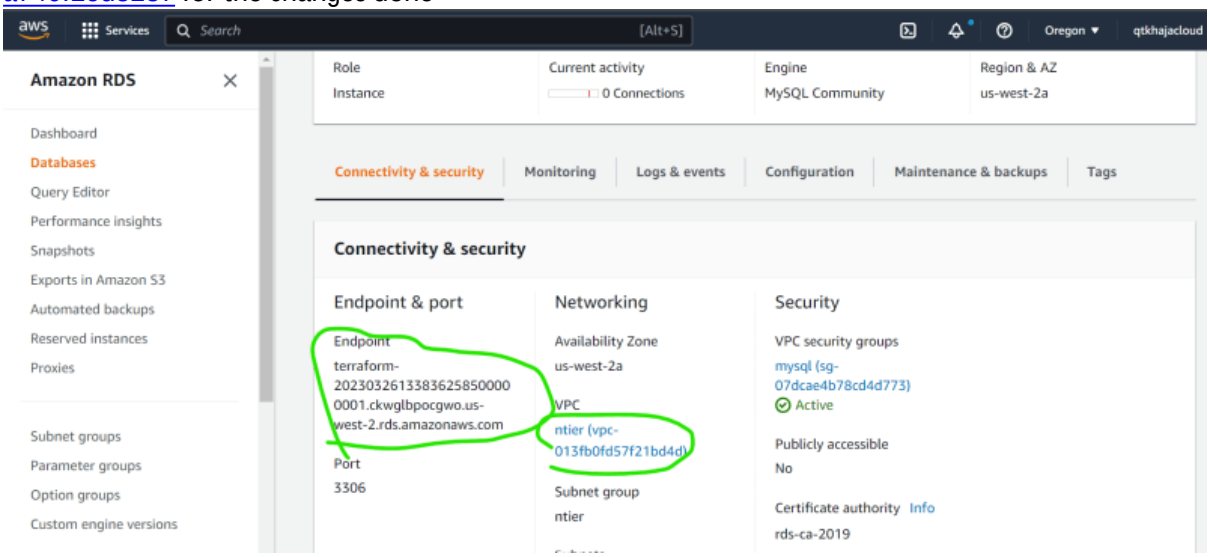
- Now we need to associate private route table with 4 subnets and public route table with 2 subnets. https://github.com/asquarezone/TerraformZone/commit/b158695ce14d57e573a9490fedc3a6e93ff44569 for the changes done



Creating RDS DB Instance (db)

- Manual Steps:
  - DB Subnet Group: This is more than one subnet where the databse has to be created
  - Security Group:
    - mysql: open 3306 port with in vpc
  - Database Engine: mysql
  - size: db.t2.micro

    - credentials: username and password
- Create security Group: https://github.com/asquarezone/TerraformZone/commit/01193b090c73deb206bb5a7958cb1ece8145c90a for changes
- Add db subnet group https://github.com/asquarezone/TerraformZone/commit/217c295f55a88521a0d2b887065f390b6dac3f29 for changes
- Create rds instance. https://github.com/asquarezone/TerraformZone/commit/a2d123044700aca6f3e850439236a746f2ed3287 for the changes done

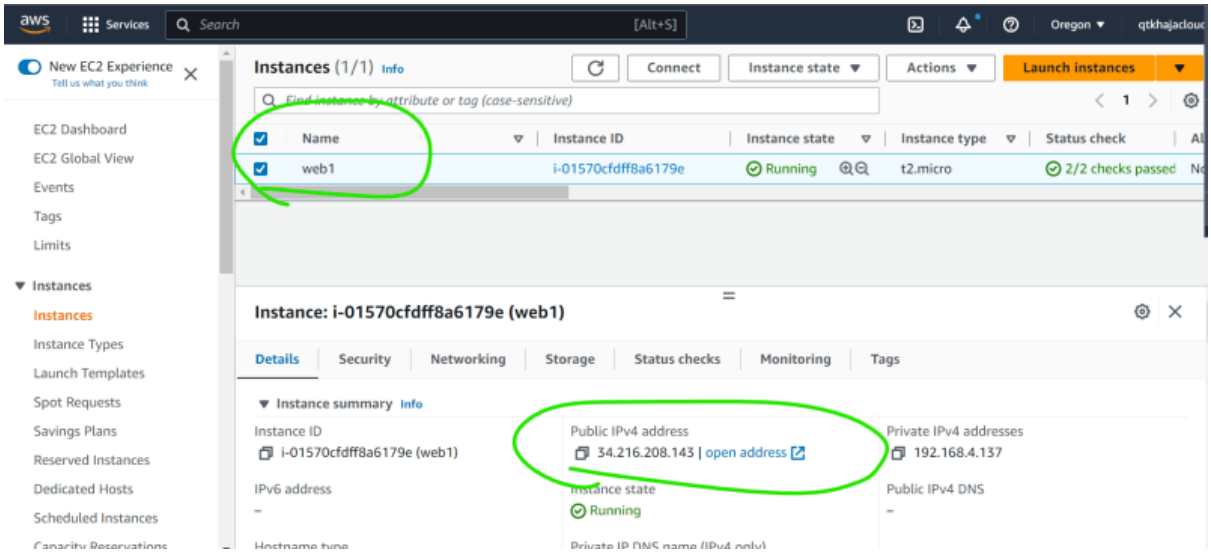


- Lets add database endpoint as output https://github.com/asquarezone/TerraformZone/commit/a2d123044700aca6f3e850439236a746f2ed3287

```
PS C:\khajaclassroom\devops\terraform\Mar23\ntier\aws> terraform output
db_endpoint = "terraform-20230326133836258500000001.ckwglbpocgwo.us-west-2.rds.amazonaws
com:3306"
PS C:\khajaclassroom\devops\terraform\Mar23\ntier\aws>
```

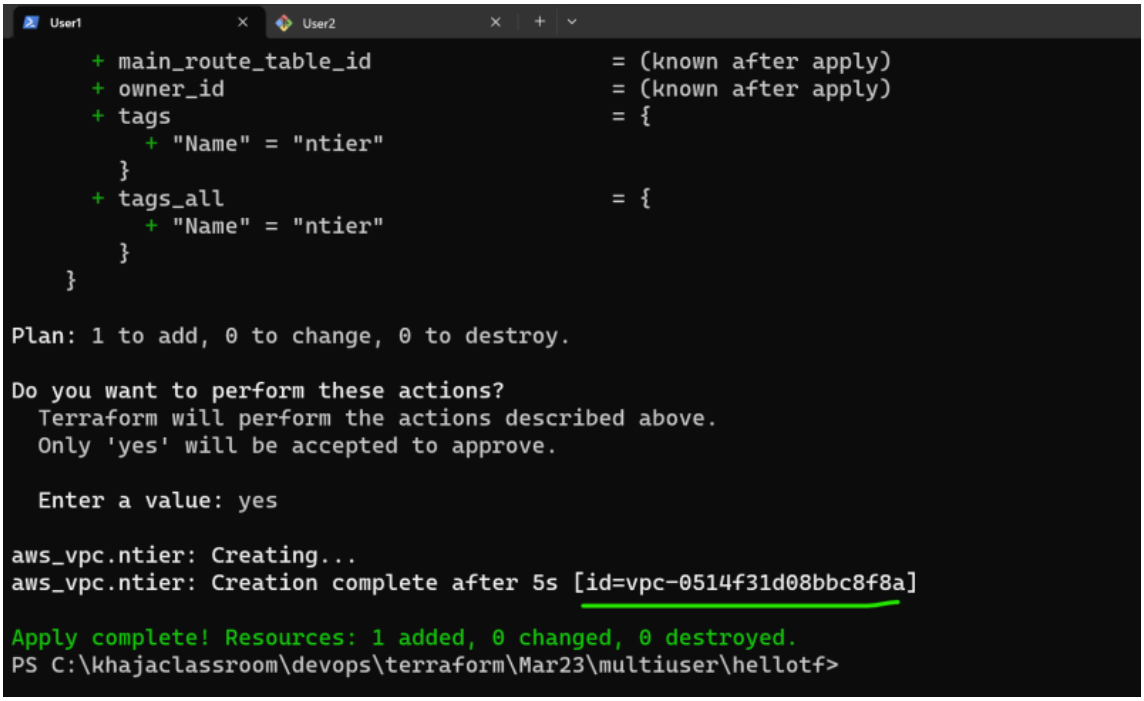
AWS EC2 from terraform

- Create an ec2 instance in web1 subnet
- Steps:
    - Create security group https://github.com/asquarezone/TerraformZone/commit/89d90acbf7b1bf1edc45e428349d92129fa4190e
- Create ec2 https://github.com/asquarezone/TerraformZone/commit/eec5edeed863dc2667df4e9a05b4a5f8891e2d3b for changes

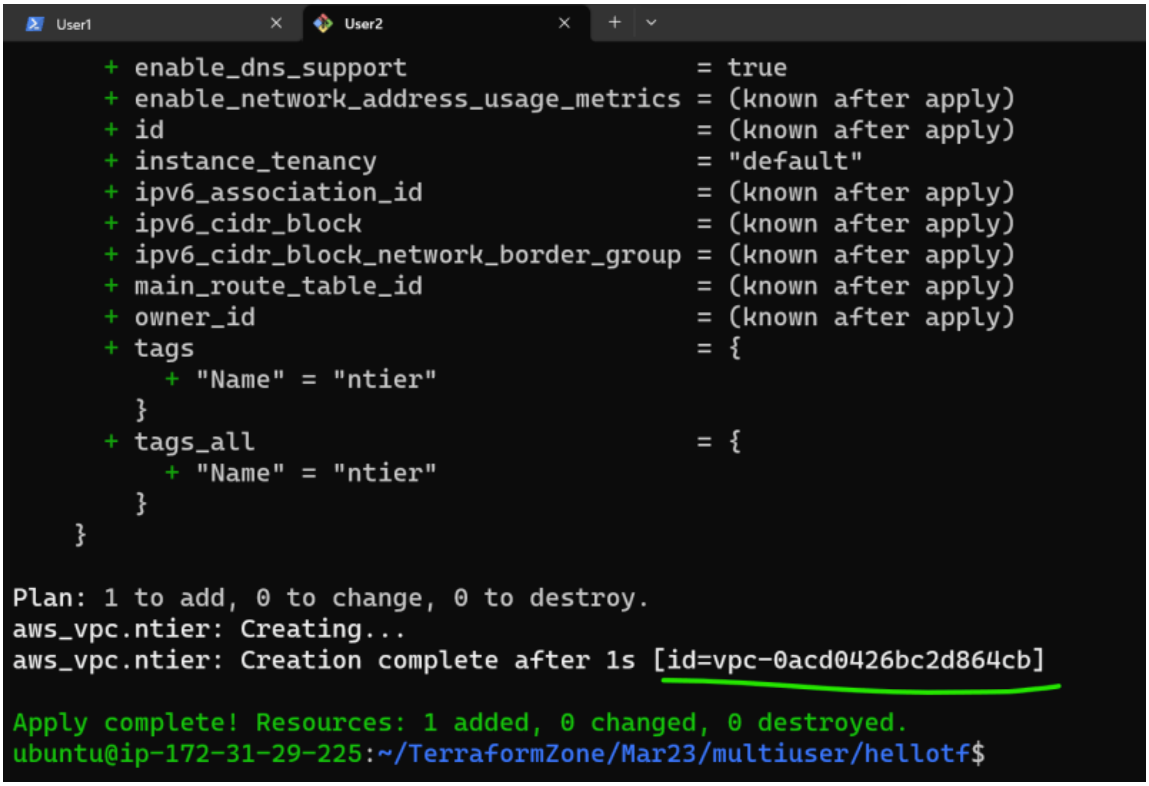

aws
Services
Search
[Alt+S]
Oregon
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Scheduled Instances
Instances (1/1) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
Status check
web1
i-01570cfdff8a6179e
Running
t2.micro
2/2 checks passed
Instance: i-01570cfdff8a6179e (web1)
Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Instance summary Info
Instance ID
i-01570cfdff8a6179e (web1)
Public IPv4 address
34.216.208.143 | open address
Private IPv4 addresses
192.168.4.137
IPv6 address
Instance state
Running
Public IPv4 DNS

Multi user setup in terraform

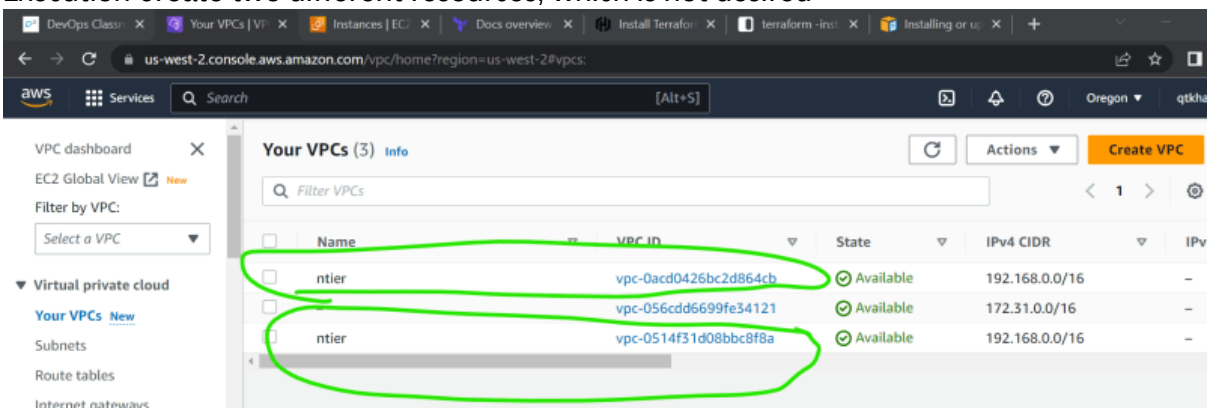
- Let's create a simple terraform template to create vpc https://github.com/asquarezone/TerraformZone/commit/0ebffe06aea8fbc28f8f99fd634e8795c1019fe7 for the changes
- Execute this from two different machines.
- Consider both of them are working for same purpose
- User 1:



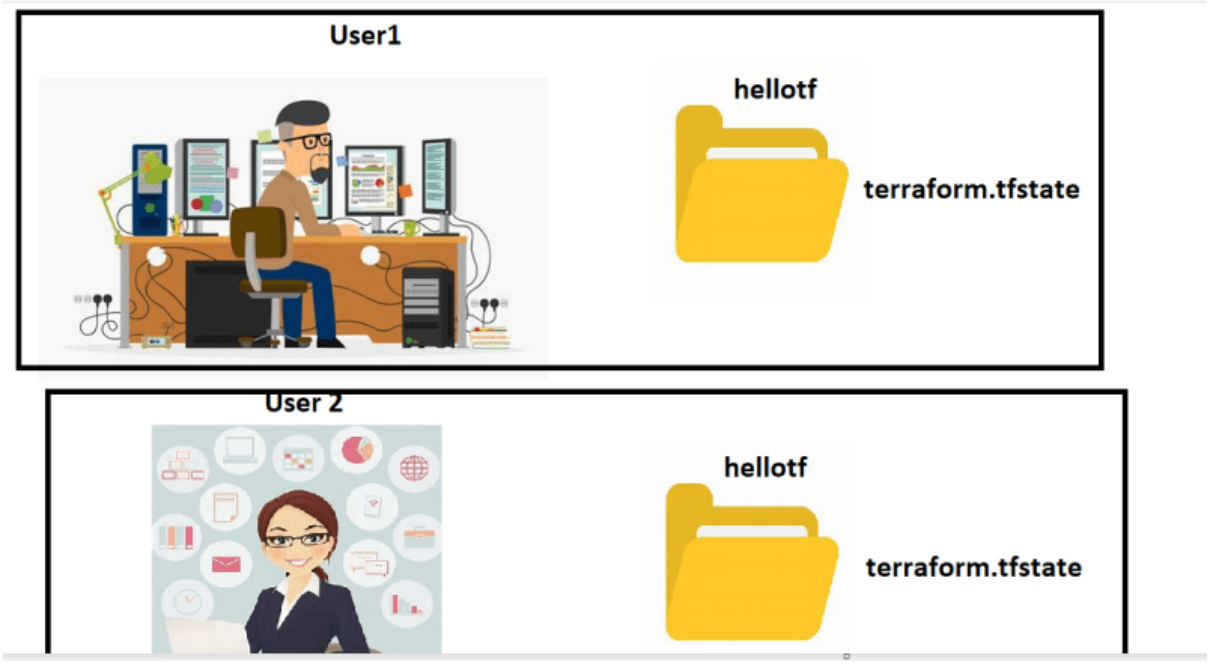
- User 2:



- Execution Create two different resources, which is not desired



- As of now state is stored locally i.e., when user1 executes it is stored in user1 system and same for user2



- To solve this problem, we need to store the state in some common place

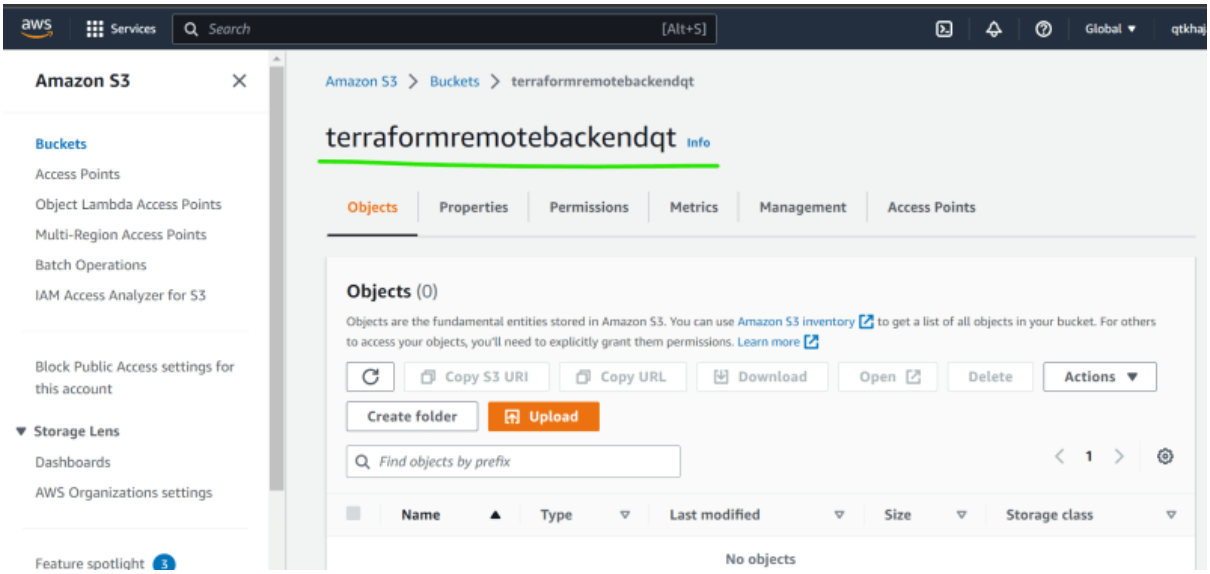
- The location of state file in terraform is defined by backend.

Terraform Backends

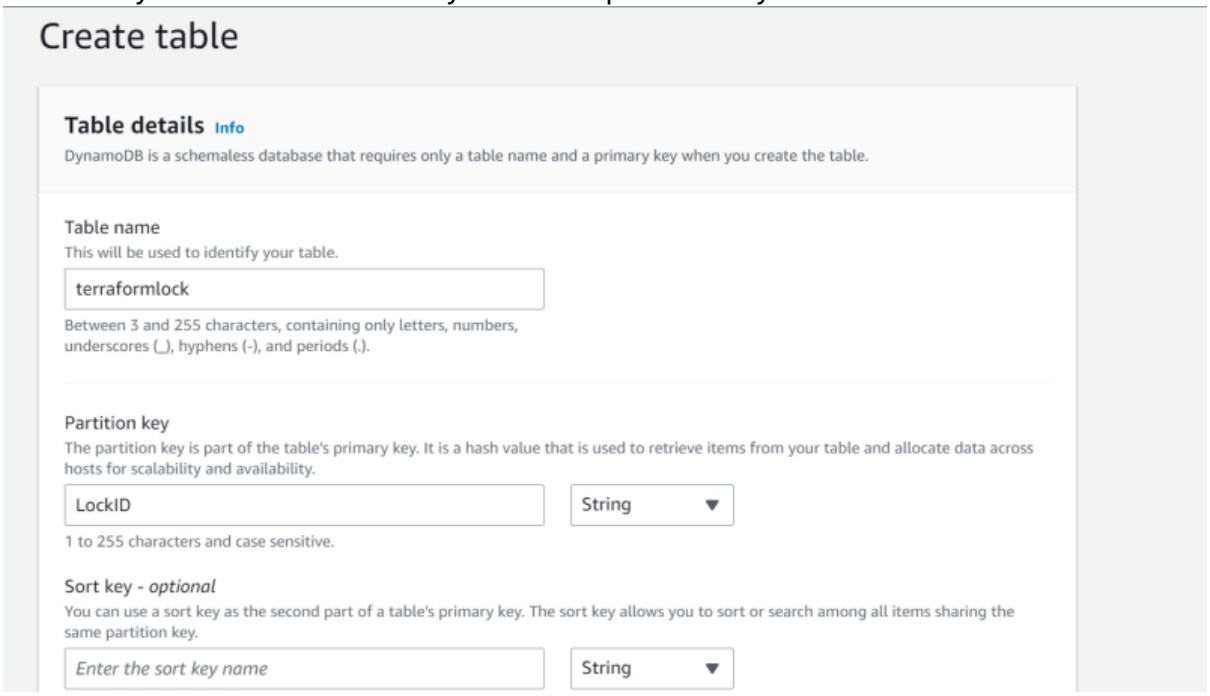
- Backend defines where the state has to be stored
- https://developer.hashicorp.com/terraform/language/state/backends for official docs and https://developer.hashicorp.com/terraform/language/settings/backends/configuration for configuring backend
- There are two types of backends
    - local-backend:
        - This is default backend
    - remote-backend
- https://developer.hashicorp.com/terraform/language/settings/backends/configuration#available-backends for available backends
- As common state for terraform for multiple users will have concurrency problem, Terraform backends need locking and unlocking
- S3 bucket can be used as terraform backend, S3 buckend doesnot support locking, if you need locking add dynamo db details

Remote Backends with S3

- Refer Here for official docs
- Create an s3 bucket



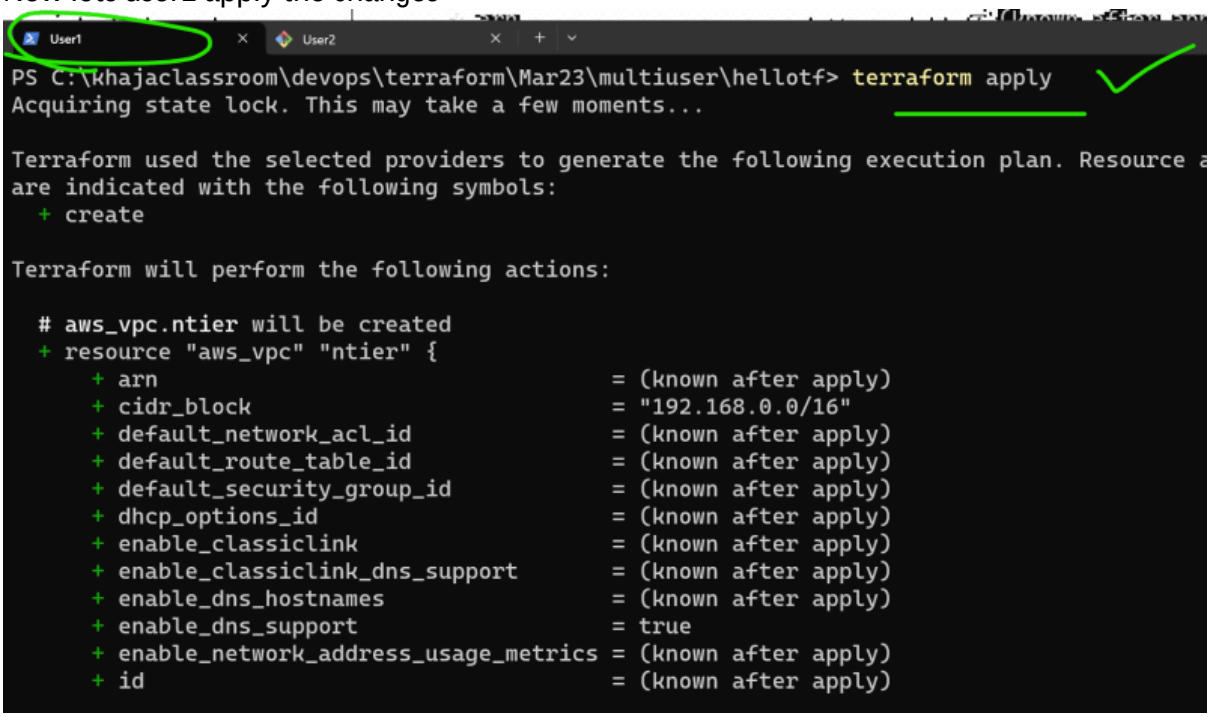
- Create a dynamo DB table with any name and partition key LockID



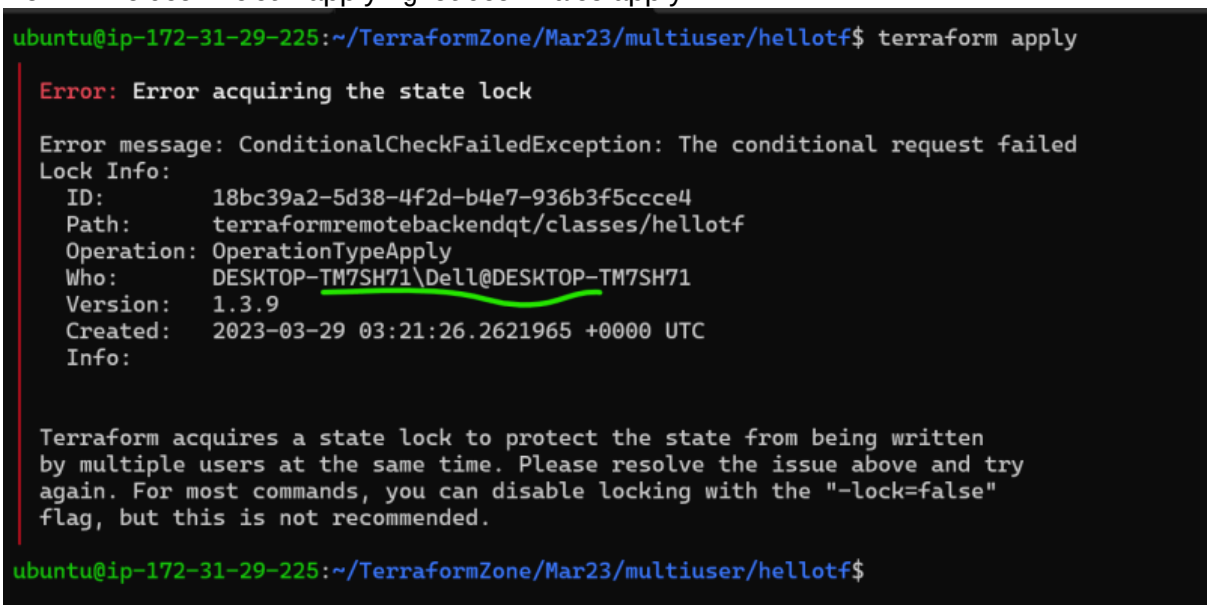
- https://github.com/asquarezone/TerraformZone/commit/84b6692c43a357dea049b69599993ef62b73f63e for the changes done to add s3 backend
- Perform init on both user machines

```
ubuntu@ip-172-31-29-225:~/TerraformZone/Mar23/multiuser/hellotf$ terraform init

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding hashicorp/aws versions matching "4.60.0"...
- Installing hashicorp/aws v4.60.0...
- Installed hashicorp/aws v4.60.0 (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
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.
ubuntu@ip-172-31-29-225:~/TerraformZone/Mar23/multiuser/hellotf$
```

- Now lets user1 apply the changes



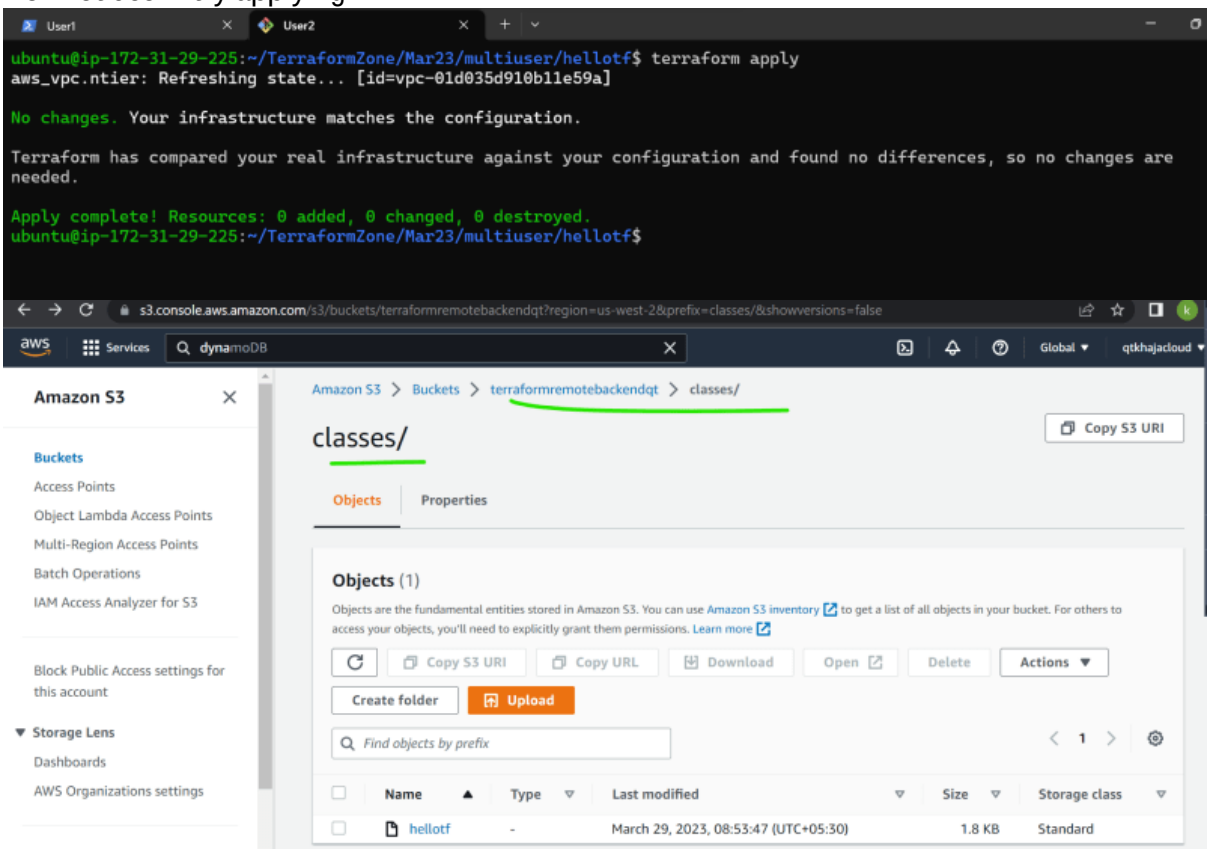
- Now while user1 is still applying let user 2 also apply



- Let user1 finish applying and create the resources

```
User1 | User2

      + owner_id                              = (known after apply)
      + tags                                  = {
          + "Name" = "ntier"
        }
      + tags_all                              = {
          + "Name" = "ntier"
        }
    }

Plan: 1 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_vpc.ntier: Creating...
aws_vpc.ntier: Creation complete after 5s [id=vpc-01d035d910b11e59a]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf>
```

- Now let user2 try applying



- Exercise: Configure Azurerm backend https://developer.hashicorp.com/terraform/language/settings/backends/azurerm which has inbuilt locking facility

Summary:

1) How to avoid duplicate resource creation: use statefile in remote backend.
2) Destroy delete all things from statefile as well.
3) When we are creating user it must have read/write access to S3 and DynmoDB.

Problem with Backends

- If we use the backends using same template for managing different environments will be tricky.
- When user 1 tries update Dev it should not stop user 2 to update QA environments

- The basic idea is to use the same template across environments
- Terraform has workspaces

Terraform workspaces: With the help of terraform workspace we can manage multiple environments with one configuration. [Somewhere similar to git branching].

- https://developer.hashicorp.com/terraform/language/state/workspaces for workspaces docs and https://developer.hashicorp.com/terraform/cli/commands/workspace for terraform workspaces cli
- Terraform by default assumes the current workspace is default

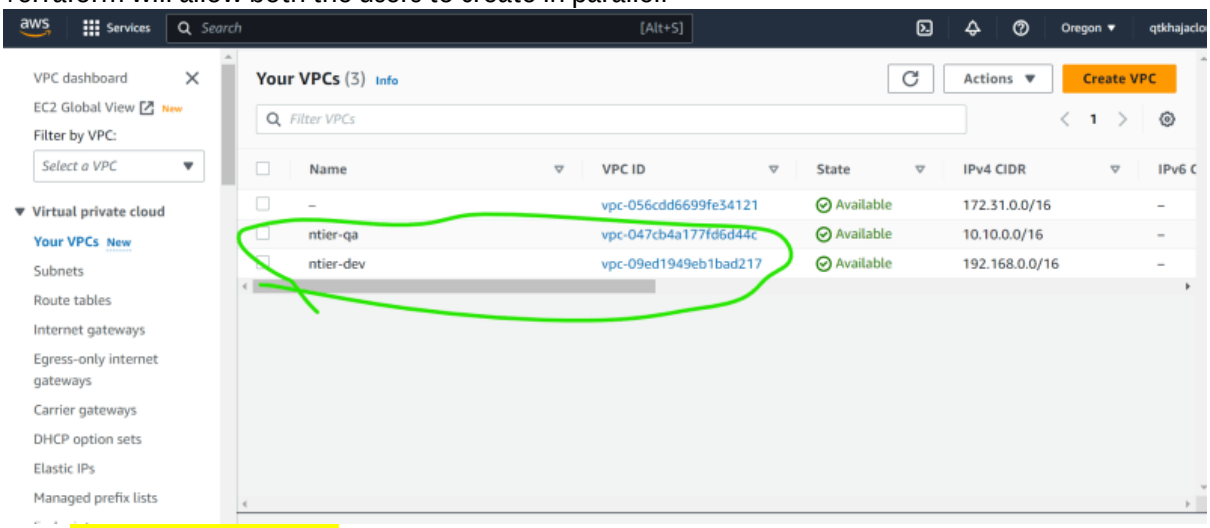
```
PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf> terraform workspace --help
Usage: terraform [global options] workspace

  new, list, show, select and delete Terraform workspaces.

Subcommands:
    delete    Delete a workspace
    list      List Workspaces
    new       Create a new workspace
    select    Select a workspace
    show      Show the name of the current workspace
PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf> terraform workspace list
* default

PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf>
```

- Let's create a workspace called as dev by user 1

```
PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf> terraform workspace new dev
Created and switched to workspace "dev"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf> terraform workspace list
  default
* dev

PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf>
```

- https://github.com/asquarezone/TerraformZone/commit/cc438f3ecfe71fd80a7de99f5165af36d83f8fcf for changes done

- Now let's create dev environment and apply the changes from user1 and created the qa environment and apply the changes from user 2.
- Terraform will allow both the users to create in parallel.



- Used conditional expression for creating a bastion subnet in qa https://github.com/asquarezone/TerraformZone/commit/82052226729ea4c070956a3169cec3ffad1108d6 for changes and https://developer.hashicorp.com/terraform/language/expressions/conditionals for conditional expressions in terraform
- Terraform allows multiple environments from one template with workspaces

Problem

- https://github.com/asquarezone/TerraformZone/commit/d88ce8fbdcdf56e405c9f09d4c24229a6f4896d8 for the change which used count to create three files
- Now let's make a small change https://github.com/asquarezone/TerraformZone/commit/d88ce8fbdcdf56e405c9f09d4c24229a6f4896d8 where the second item from list is removed
- now apply to see the plan

```
      - content              = "hello" -> null
      - content_base64sha256 = "LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmC
      - content_base64sha512 = "m3HSJL1i83hdltRq0+o9czGb+8KJDKra4t/3JRlnPK
ZjR1wuXDre9G9zvN7AQw==" -> null
      - content_md5          = "5d41402abc4b2a76b9719d911017c592" -> null
      - content_sha1         = "aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d"
      - content_sha256       = "2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1f
-> null
      - content_sha512       = "9b71d224bd62f3785d96d46ad3ea3d73319bfbc289
23c3d99ba5c11d7c7acc6e14b8c5da0c4663475c2e5c3adef46f73bcdec043" -> null
      - directory_permission = "0777" -> null
      - file_permission      = "0777" -> null
      - filename             = "test3.txt" -> null
      - id                   = "aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d"
    }

Plan: 1 to add, 0 to change, 2 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
```

- Terraform is deleting test2.txt and test3.txt also and then creates test3.txt again to maintain state
- To solve this problem, we can use for_each
  https://developer.hashicorp.com/terraform/language/meta-arguments/for_each
- Exercise: Try solving the above problem with for_each

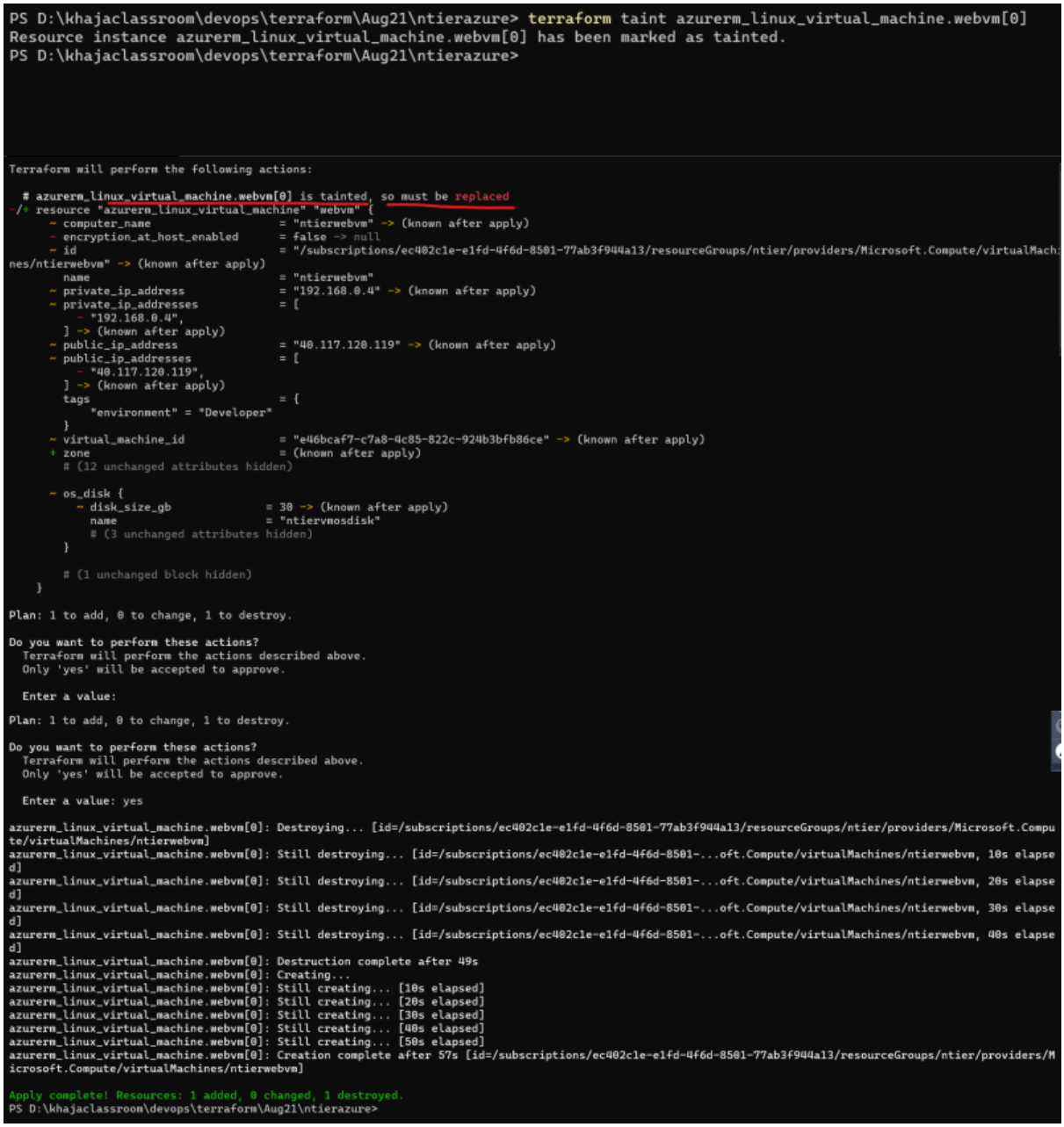
Interpolation in terraform

- Interpolation can be acheived by using format function or ${var} expresssion

```
PS C:\khajaclassroom\devops\terraform\Mar23\multiuser\hellotf> terraform console
Acquiring state lock. This may take a few moments...
> "ntier-${terraform.workspace}"
"ntier-dev"
> format("ntier-%s", terraform.workspace)
"ntier-dev"
>
```

Terraform conditionals

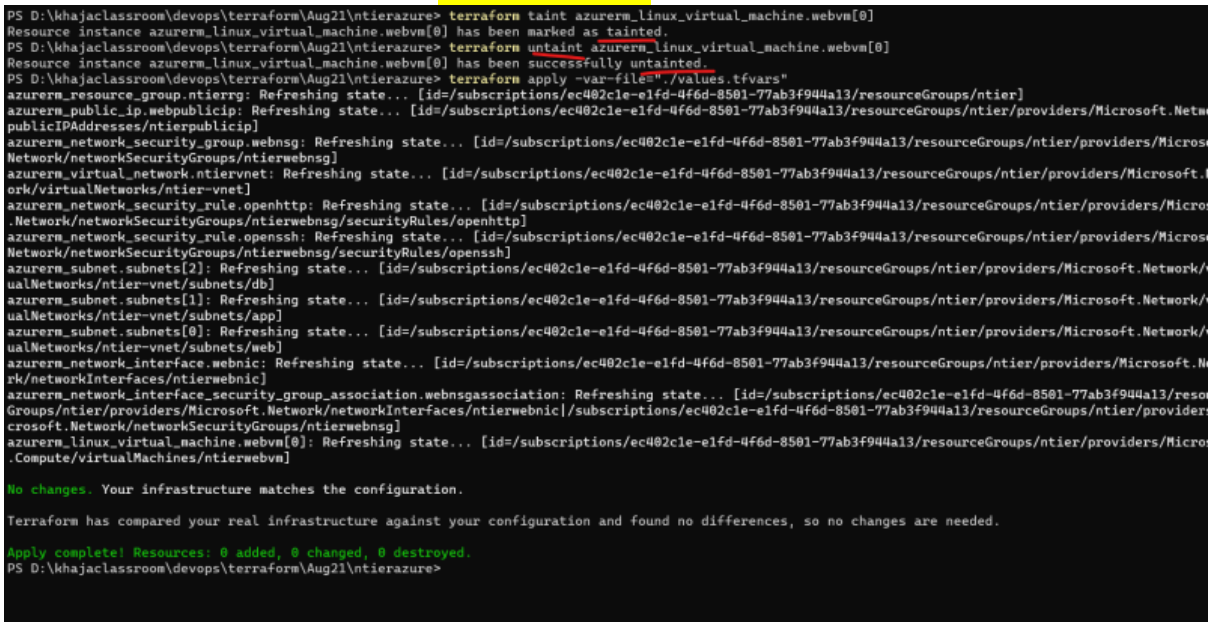
Terraform has a taint command that informs terraform that a particular object has become degraded or damaged and during the next execution of terraform replace it.



```
PS D:\khajaclassroom\devops\terraform\Aug21\ntierazure> terraform taint azurerm_linux_virtual_machine.webvm[0]
Resource instance azurerm_linux_virtual_machine.webvm[0] has been marked as tainted.
PS D:\khajaclassroom\devops\terraform\Aug21\ntierazure>
```

```
Terraform will perform the following actions:

  # azurerm_linux_virtual_machine.webvm[0] is tainted, so must be replaced
-/+ resource "azurerm_linux_virtual_machine" "webvm" {
      ~ computer_name                  = "ntierwebvm" -> (known after apply)
      - encryption_at_host_enabled     = false -> null
      ~ id                             = "/subscriptions/ec402c1e-e1fd-4f6d-8501-77ab3f944a13/resourceGroups/ntier/providers/Microsoft.Compute/virtualMach
nes/ntierwebvm" -> (known after apply)
        name                           = "ntierwebvm"
      ~ private_ip_address             = "192.168.0.4" -> (known after apply)
      ~ private_ip_addresses           = [
          - "192.168.0.4",
        ] -> (known after apply)
      ~ public_ip_address              = "40.117.120.119" -> (known after apply)
      ~ public_ip_addresses            = [
          - "40.117.120.119",
        ] -> (known after apply)
        tags                           = {
            "environment" = "Developer"
        }
      ~ virtual_machine_id             = "e46bcaf7-c7a8-4c85-822c-924b3bfb86ce" -> (known after apply)
      + zone                           = (known after apply)
        # (12 unchanged attributes hidden)

      ~ os_disk {
          ~ disk_size_gb             = 30 -> (known after apply)
            name                     = "ntiervmosdisk"
            # (3 unchanged attributes hidden)
        }

        # (1 unchanged block hidden)
    }

Plan: 1 to add, 0 to change, 1 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:

Plan: 1 to add, 0 to change, 1 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

azurerm_linux_virtual_machine.webvm[0]: Destroying... [id=/subscriptions/ec402c1e-e1fd-4f6d-8501-77ab3f944a13/resourceGroups/ntier/providers/Microsoft.Compu
te/virtualMachines/ntierwebvm]
azurerm_linux_virtual_machine.webvm[0]: Still destroying... [id=/subscriptions/ec402c1e-e1fd-4f6d-8501-...oft.Compute/virtualMachines/ntierwebvm, 10s elapse
d]
azurerm_linux_virtual_machine.webvm[0]: Still destroying... [id=/subscriptions/ec402c1e-e1fd-4f6d-8501-...oft.Compute/virtualMachines/ntierwebvm, 20s elapse
d]
azurerm_linux_virtual_machine.webvm[0]: Still destroying... [id=/subscriptions/ec402c1e-e1fd-4f6d-8501-...oft.Compute/virtualMachines/ntierwebvm, 30s elapse
d]
azurerm_linux_virtual_machine.webvm[0]: Still destroying... [id=/subscriptions/ec402c1e-e1fd-4f6d-8501-...oft.Compute/virtualMachines/ntierwebvm, 40s elapse
d]
azurerm_linux_virtual_machine.webvm[0]: Destruction complete after 49s
azurerm_linux_virtual_machine.webvm[0]: Creating...
azurerm_linux_virtual_machine.webvm[0]: Still creating... [10s elapsed]
azurerm_linux_virtual_machine.webvm[0]: Still creating... [20s elapsed]
azurerm_linux_virtual_machine.webvm[0]: Still creating... [30s elapsed]
azurerm_linux_virtual_machine.webvm[0]: Still creating... [40s elapsed]
azurerm_linux_virtual_machine.webvm[0]: Still creating... [50s elapsed]
azurerm_linux_virtual_machine.webvm[0]: Creation complete after 57s [id=/subscriptions/ec402c1e-e1fd-4f6d-8501-77ab3f944a13/resourceGroups/ntier/providers/M
icrosoft.Compute/virtualMachines/ntierwebvm]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
PS D:\khajaclassroom\devops\terraform\Aug21\ntierazure>
```

Terraform taint can be undone with untaint command



Every provider in terraform gives us data sources which we can use to pull information. The basic syntax of defining datasource is

```
data "type" "name" {
   arg1 = value1
   ...
   argn = valuen
}
```

Datasource: A data source is accessed via a special kind of resource known as a *data resource*, declared using a `data` block:

```
data "aws_ami" "example" {
  most_recent = true

  owners = ["self"]
  tags = {
    Name   = "app-server"
    Tested = "true"
  }
}
```

Terraform Modules

- A Terraform module is a reusable set of terraform configuration files.
- Terraform has lot of modules from community which can be reused

A module is a container for multiple resources that are used together. Every Terraform configuration has at least one module, known as its root module, which consists of the resources defined in the . tf files in the main working directory.

https://developer.hashicorp.com/terraform/language/modules/sources#github

How to create terraform module?

By defining below values in the main.tf file

```
module "webserver_cluster" {
  source = "../../../modules/services/webserver-cluster"  cluster_name          = "webservers-stage"
  db_remote_state_bucket = "(YOUR_BUCKET_NAME)"
  db_remote_state_key    = "stage/data-stores/mysql/terraform.tfstate"
}
```

Blog: https://blog.gruntwork.io/how-to-create-reusable-infrastructure-with-terraform-modules-25526d65f73d

- Module demo https://github.com/asquarezone/TerraformZone/commit/8a1e26f0e90e3d1018c607c6774069c4c2743390 for module usage with aws
- To create ec2 https://registry.terraform.io/modules/terraform-aws-modules/ec2-instance/aws/latest

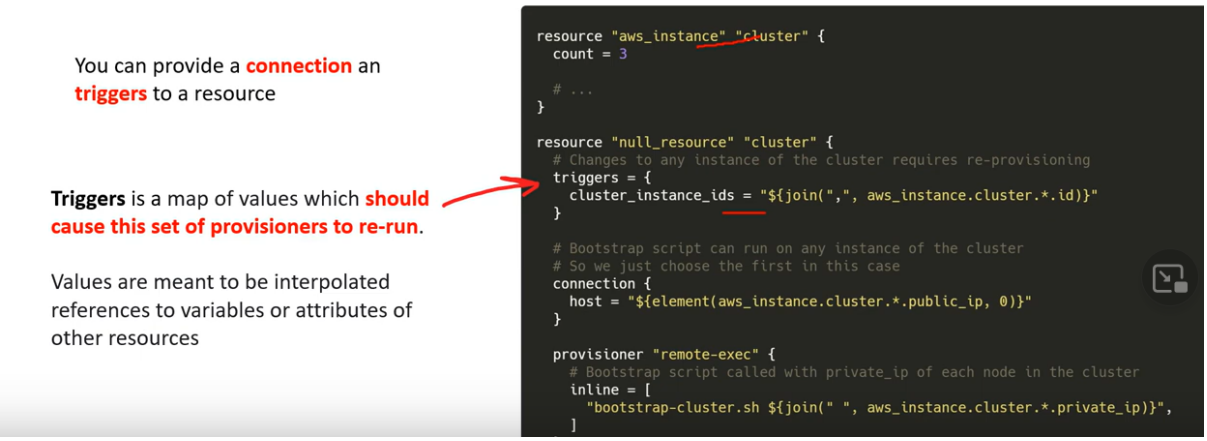
Terraform provisioners

- Provisioners are for performing activities in local or remote instance after resource creation
    - local https://developer.hashicorp.com/terraform/language/resources/provisioners/syntax#provisioners
    - remote https://developer.hashicorp.com/terraform/language/resources/provisioners/remote-exec

Terraform Provisioning

- Terraform has 3 basic provisioners
  - file: to copy the files from local to remote
  - local-exec: is used to execute commands in the machine where terraform is running
  - remote-exec: is used to executed commands on the machines created by terraform
- When we are working with remote system i.e. vm created by terraform we need to establish connection. Terraform has connection block which supports two kinds of connections
  - ssh
  - winrm

null_resource is a placeholder for resources that have no specific association to a provider resources.




```
resource "null_resource" "provision" {
  provisioner "local-exec" {
    command = "echo ${timestamp()} >> /tmp/now.tx
  }
}
```

Above, We are taking timestamp when we are apply command are getting executed, we can run Ansible playbook, and so many things with null resource.

Packer

- Every virtualization platform has its way of creating the reusable images.
- Packer is a tool which can automate this. [https://www.packer.io/] > Image as code
- To install packer on your system [https://developer.hashicorp.com/packer/downloads]
- Terminology:
    - Builder: Builder defines where the image has to be created
    - Provisioners: What provisioning can be done to create the iamge
    - Variables: this section defines the parameters that can be passed as user input.

Packer file:
https://github.com/asquarezone/CloudFormationTemplates/commit/9f75abf5a18be07d0117bc4e06425fd7794327c1

```
{
    "variables": {
        "aws_access_key": "",
        "aws_secret_key": ""
    },
    "builders": [
        {
            "type": "amazon-ebs",
            "ami_name": "apachefrompacker",
            "instance_type": "t2.micro",
            "source_ami": "ami-04bde106886a53080",
            "access_key": "{{user `aws_access_key` }}",
            "secret_key": "{{user `aws_secret_key`}}",
            "region": "ap-south-1",
            "ssh_username" : "ubuntu"
        }
    ],
    "provisioners": [
        {
            "type": "shell",
            "inline": ["sudo apt update", "sudo apt install apache2
-y"]
        }
    ]
}
```