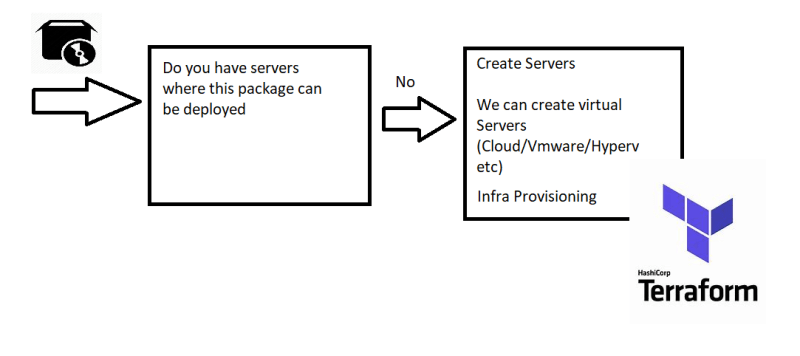
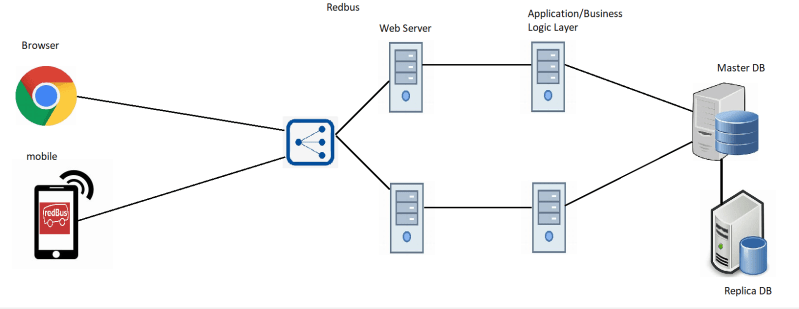
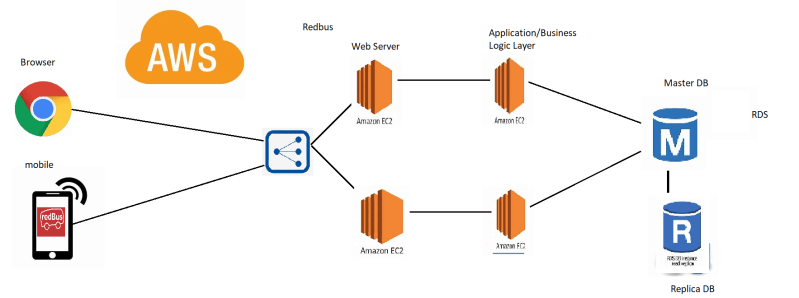
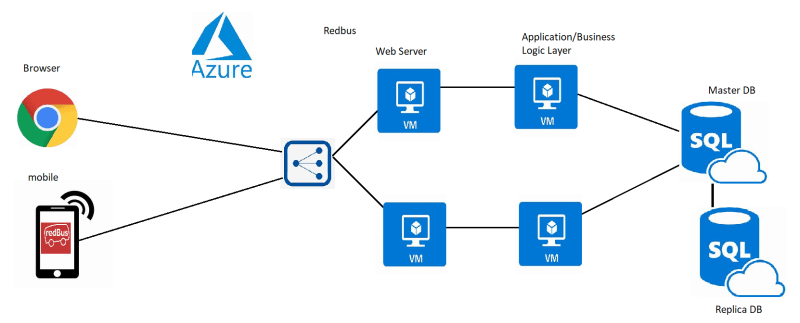
MAY 17, 2022

DevOps Classroomnotes 17/May/2022

**Terraform**

* Software isn’t done until you deliver it to the customers.
* What Terraform Does?  
  

**Infrastructure as code (IAC)**

* The idea behind IAC is that you write & execute the code to define, deploy, update and destroy your infrastructure
* Consider the following as the architecture of Redbus Application  
  
* Now lets assume RedBus is working on AWS  
  
* Now lets assume Redbus is working on Azure  
  

# AWS

resource "aws\_instance" "app" {

instance\_type = "t2.micro"

ami = "ami-892347389247"

availability\_zone = "us-west-2a"

}

# Azure

resource "azurerm\_vm" "app" {

size = "Standard\_B1s"

vmimage = "ubuntu2004LTS"

location = "eastus"

}

#vmware

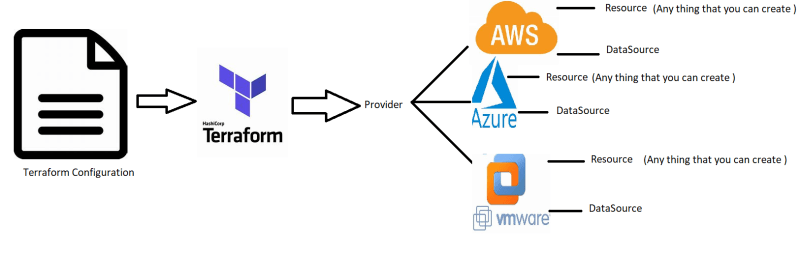
resource "vmware\_vm" "app" {

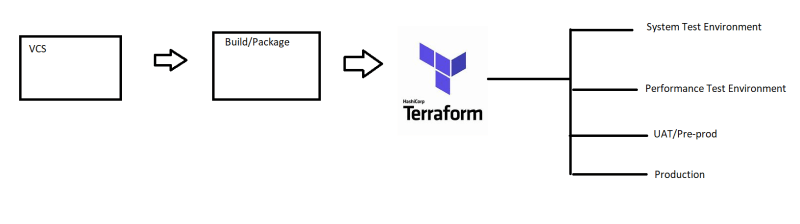
cpu = "2"

ram = "4GB"

image = "ubuntu"

}



* The major concepts of terraform are
  + Provider:
    - Where we can create infrastructure
  + Resource:
    - What has to be created
  + DataSource:
    - Query information about anything in the Provider
* Relevance of Terraform in CI/CD Pipeline  
  

**Machine Setup For DevOps**

* [Refer Here](https://www.youtube.com/watch?v=mRILfUNbsIo&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=14)
* Any Windows 10 users [Refer Here](https://www.youtube.com/watch?v=qLVn2EvPsYc&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=11)
* Mac:
  + Install Homebrew [Refer Here](https://brew.sh/)
  + Install git brew install git
  + visual studio code [Refer Here](https://formulae.brew.sh/cask/visual-studio-code) brew install --cask visual-studio-code

**Share this:**

Top of Form

Email Address

FOLLOW

Bottom of Form

## ADS

[https://secure.gravatar.com/avatar/6e3ce3931487d4f7575f04d7786e6fd4?s=48&d=identicon&r=g](https://directdevops.blog/author/shaikkhajaibrahim6e5af9224a/)

MAY 18, 2022

# DevOps Class room notes 18/May/2022

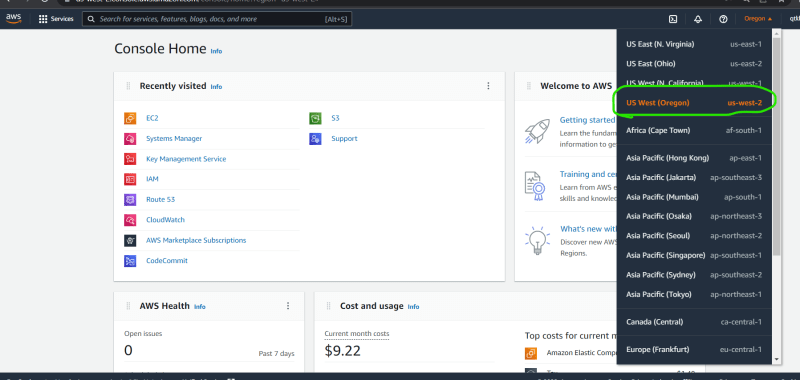
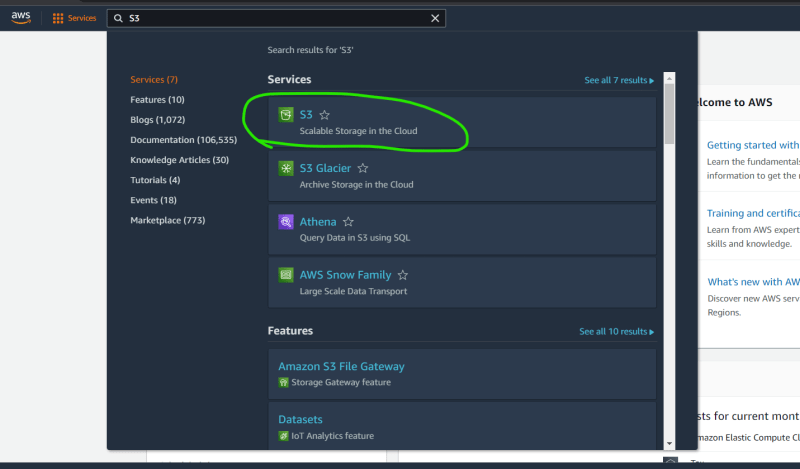
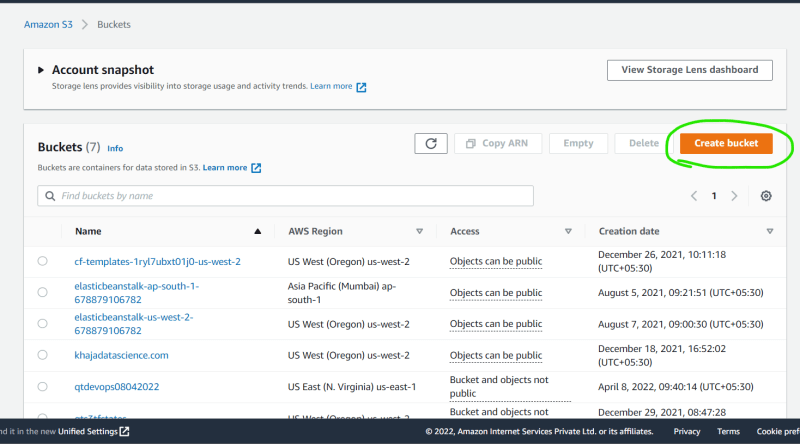
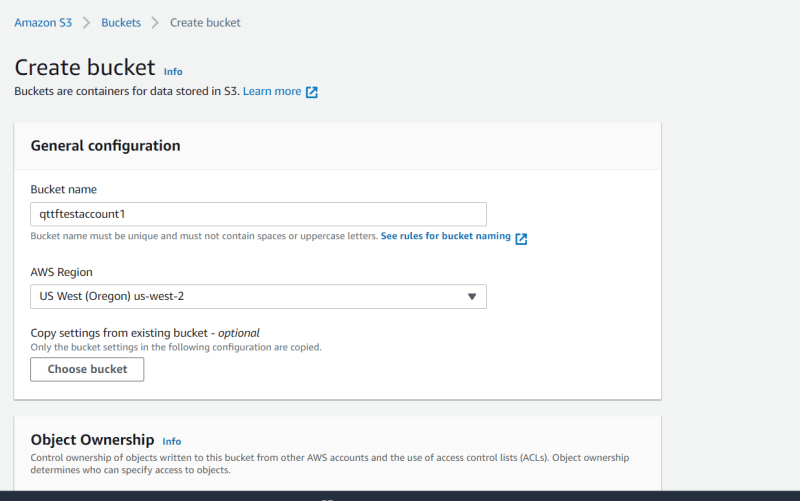
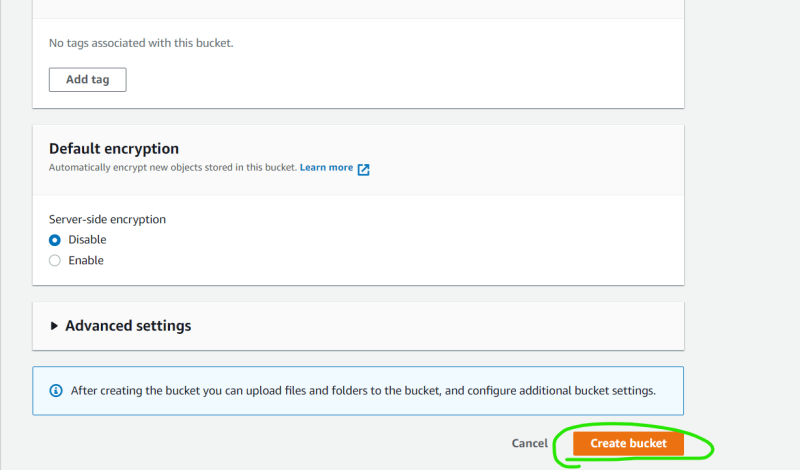
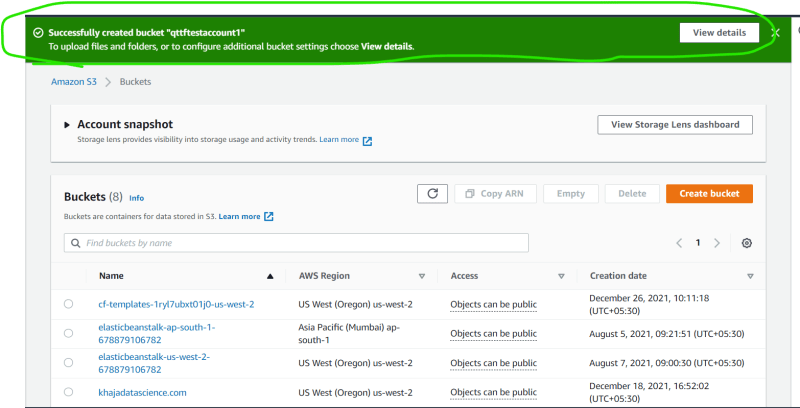
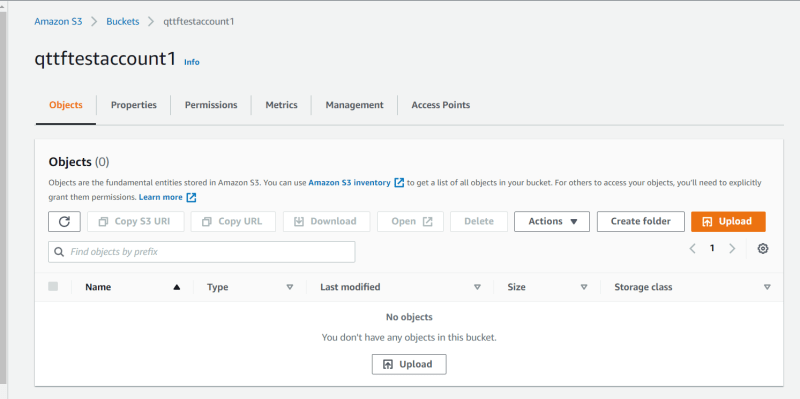
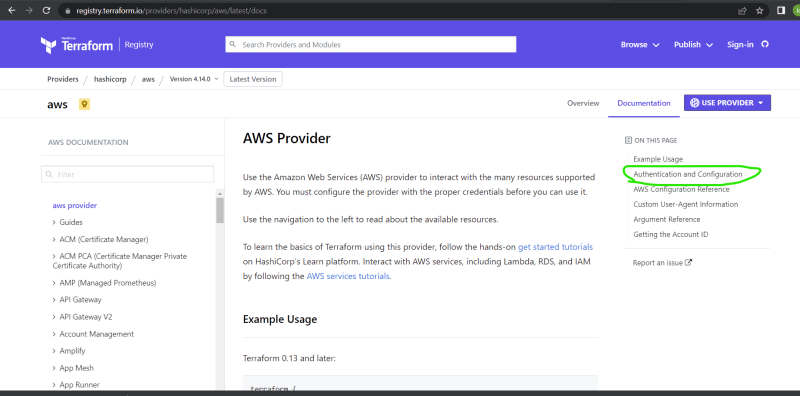
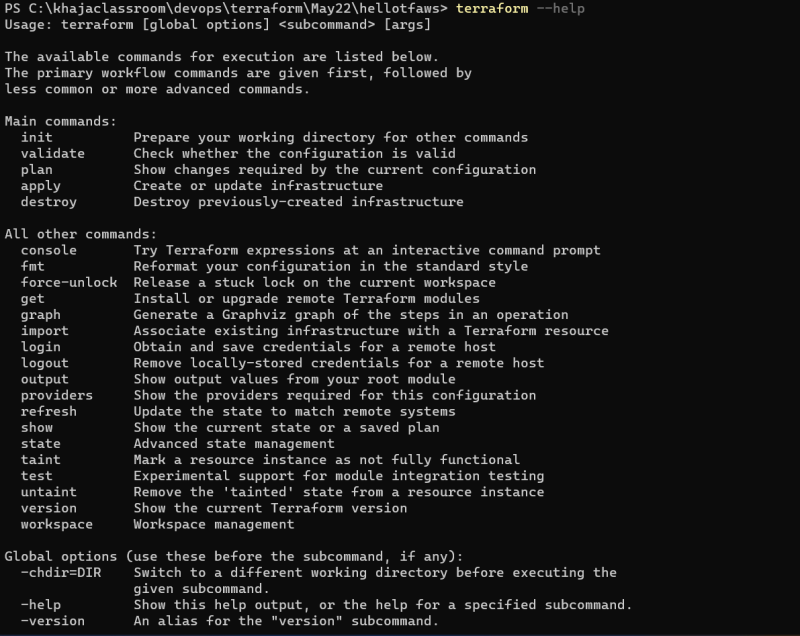
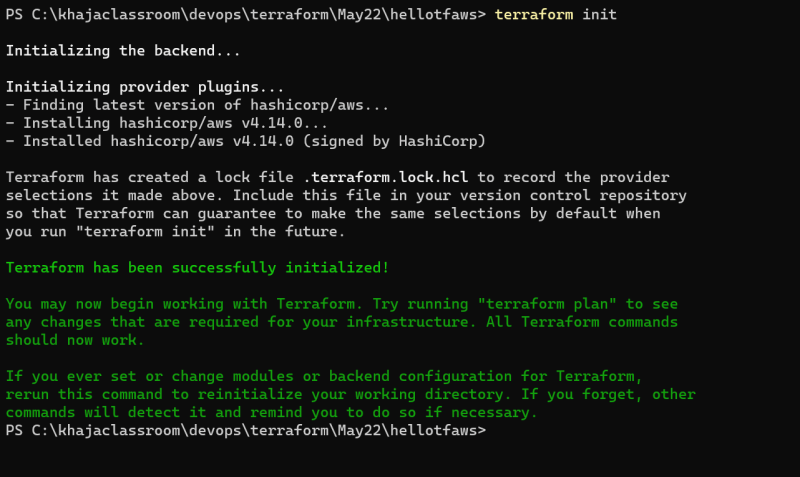
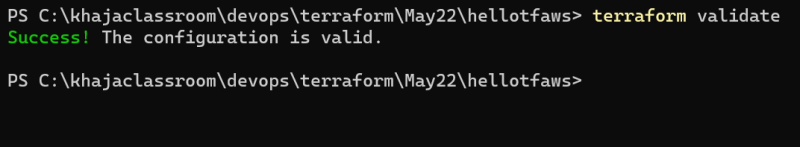
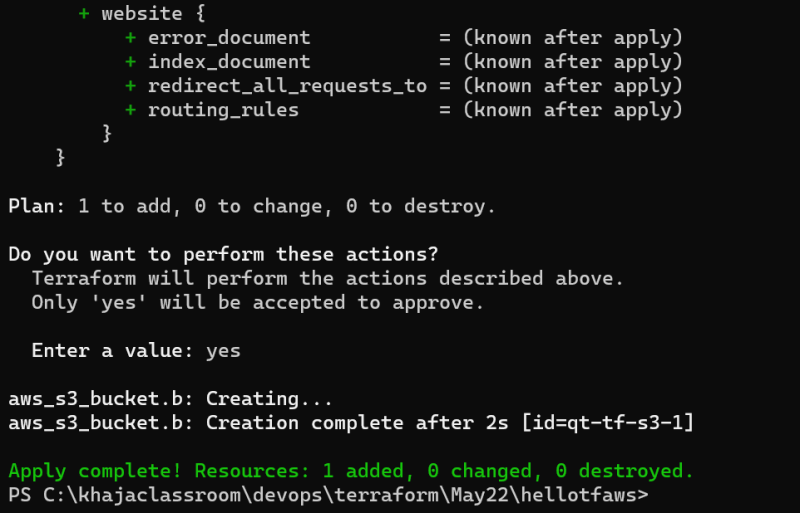
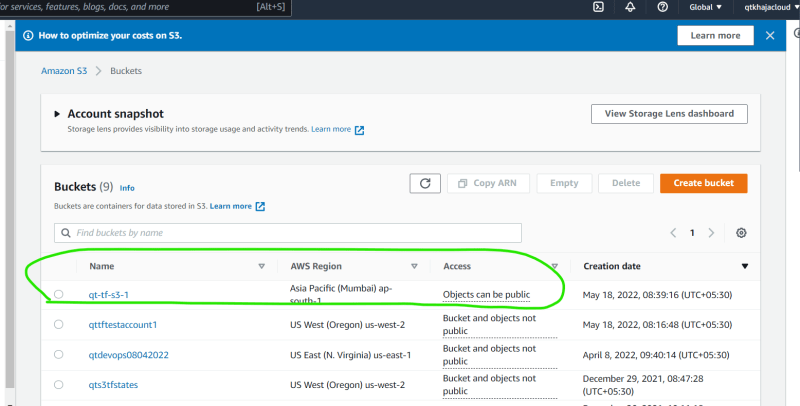
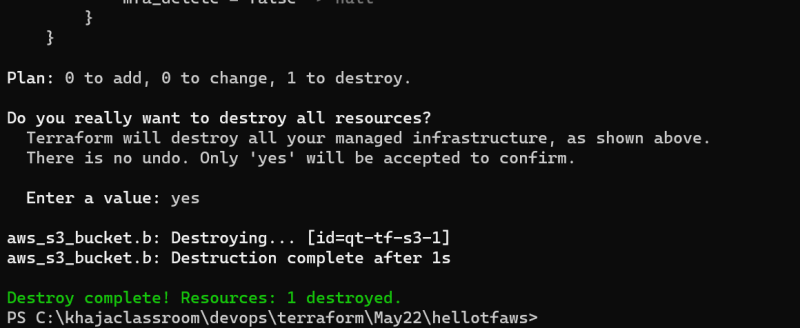
## Installing Terraform

* Windows: We have two options for installing terraform
  + Chocolatey:
    - Install chocolatey [Refer Here](https://chocolatey.org/install)
    - Use Chocolatey to install terraform choco install terraform
  + Manually configure terraform [Refer Here](https://www.terraform.io/downloads)
* Mac: [Refer Here](https://formulae.brew.sh/formula/terraform) brew install terraform

## Working with Terraform

* Identify the infrastructure you need to create/provision
* Identify the providers
* Configure the provider using HCL (Hashicorp Configuration Language)
* Configure the resources to be created using HCL

#### Activity 1: Create an S3 Bucket in AWS

* Manual Steps: Login to the AWS Account  
    
    
    
    
    
    
  
* Provider is where to create infra and Resource is what we are trying to create.
* In the above case
  + Provider: AWS
  + Resource: S3 Bucket
* In AWS to connect to aws account we had to login using email and password.
* Generally Provider configuration involves authentication and it depends on how provider allows it.
* [Refer Here](https://registry.terraform.io/browse/providers) for the terraform providers
* AWS Provider documentation  
  
* To work with AWS Provider we need access key and secret key, lets see how to generate it [Refer Here](https://serverless-stack.com/chapters/create-an-iam-user.html)
* [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket) for the documentation of the s3 bucket
* Now lets try to create infra from terraform  
  
* Initialize the directory  
  
* Validate the configuration  
  
* Now lets create the infrastructure terraform apply  
    
  
* Now to remove the infra terraform destroy  
  
* The configuration which we used

provider "aws" {

access\_key = "<your-access-key>"

secret\_key = "<your-secret-key>"

region = "ap-south-1"

}

resource "aws\_s3\_bucket" "b" {

bucket = "qt-tf-s3-1"

tags = {

Name = "My bucket"

Environment = "Dev"

}

}

### Terraform Language

* Terraform configuration can be expressed in
  + HCL/Terraform Language
  + JSON
* The two major elements in the Terraform language are
  + Arguments:
    - Assigns a value to a particular name bucket = "qt-tf-s3-1"
  + Blocks:
    - A block is container for other content
    - Three major blocks of terraform are
      * provider block
      * resource block
      * datasource block

#### Provider Block:

* Basic Syntax:

provider "<provider-name>" {

<argument-1>

<argument-2>

..

<argument-n>

}

* Example

provider "aws" {

access\_key = "LKJLKSKLJDALDJLKSADSLA"

secret\_key = "lksdfjdlkasfjlsadfjlksdafjlksdafjdallksafj"

region = "ap-south-1"

}

#### Resource Block

* Basic Syntax

resource "<PROVIDER>\_<TYPE>" "<NAME>" {

<argument-1>

<argument-2>

..

<argument-n>

}

* Example

resource "aws\_s3\_bucket" "b" {

bucket = "qt-tf-s3-1"

tags = {

Name = "My bucket"

Environment = "Dev"

}

}

#### Cloud Account Creations

* Azure: [Refer Here](https://www.youtube.com/watch?v=MdDOc9OPVDA&list=PLuVH8Jaq3mLuqXuGs6aeqxhuvCYSzB1kT) for the free account creation
* AWS: [Refer Here](https://www.youtube.com/watch?v=z95MhW1gAcA&list=PLuVH8Jaq3mLszrC7lv68a0VcrDripW-HK&index=1)

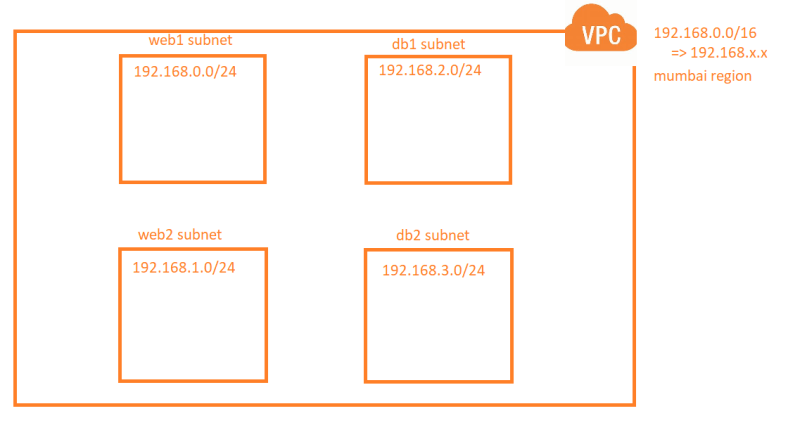
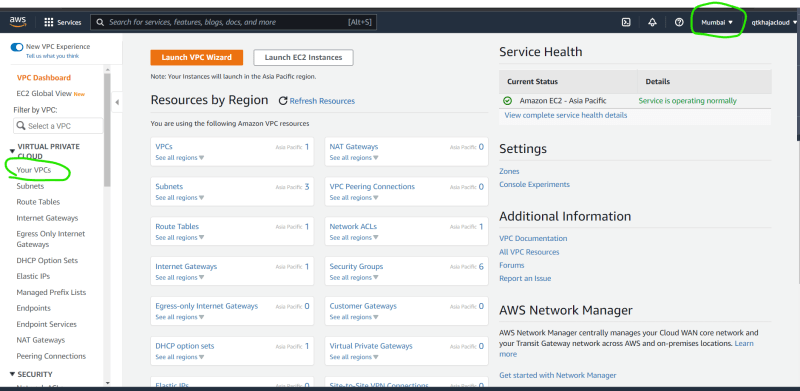
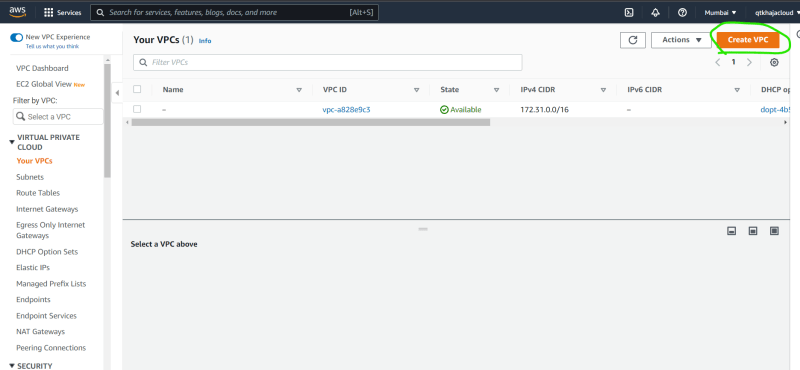
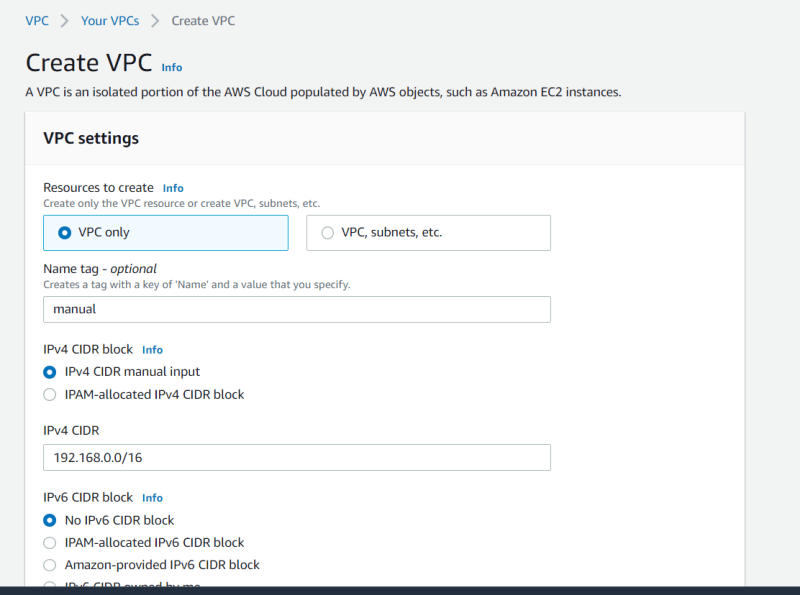
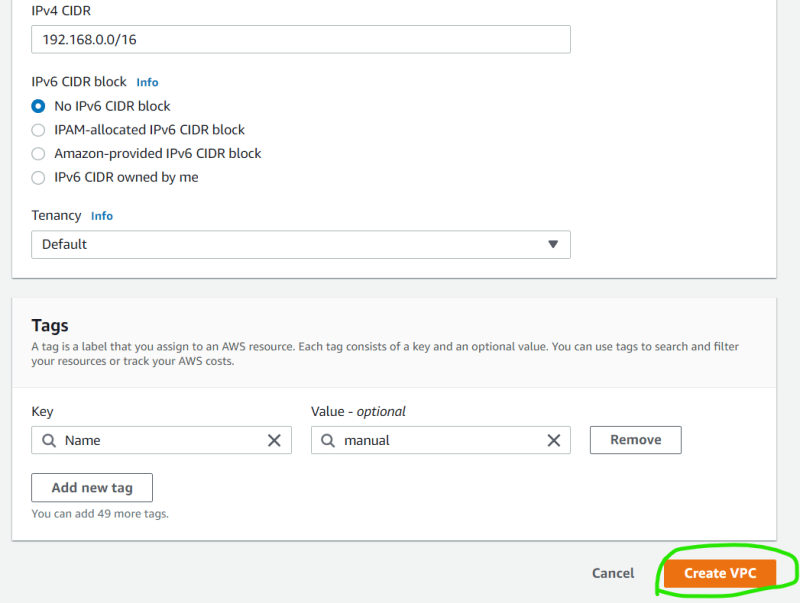
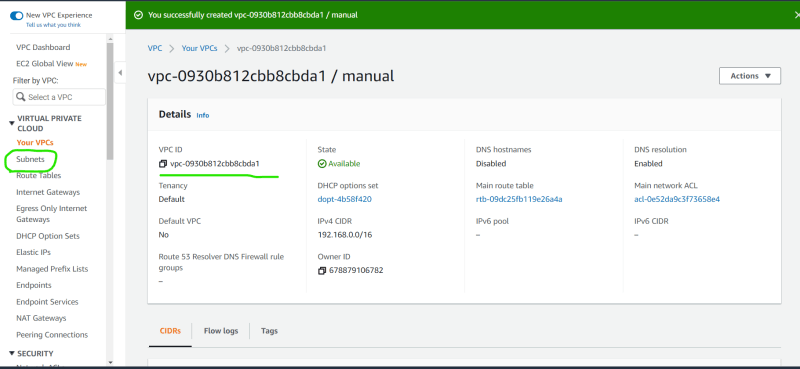
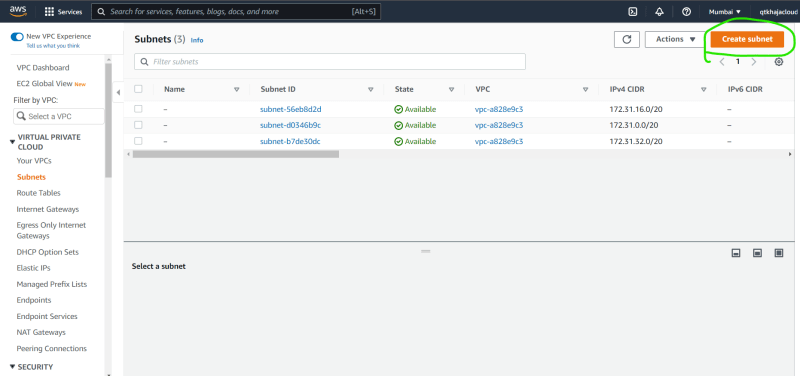
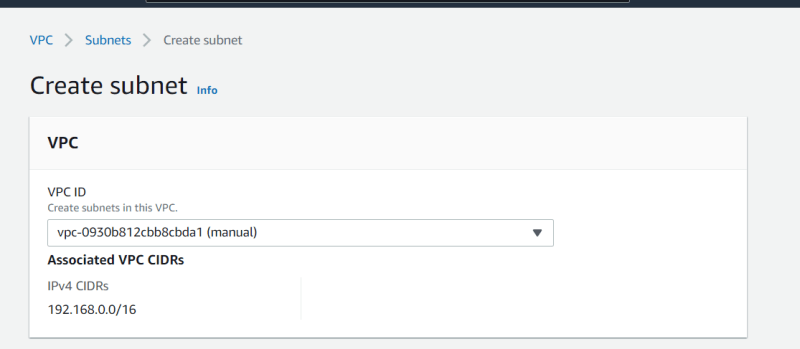
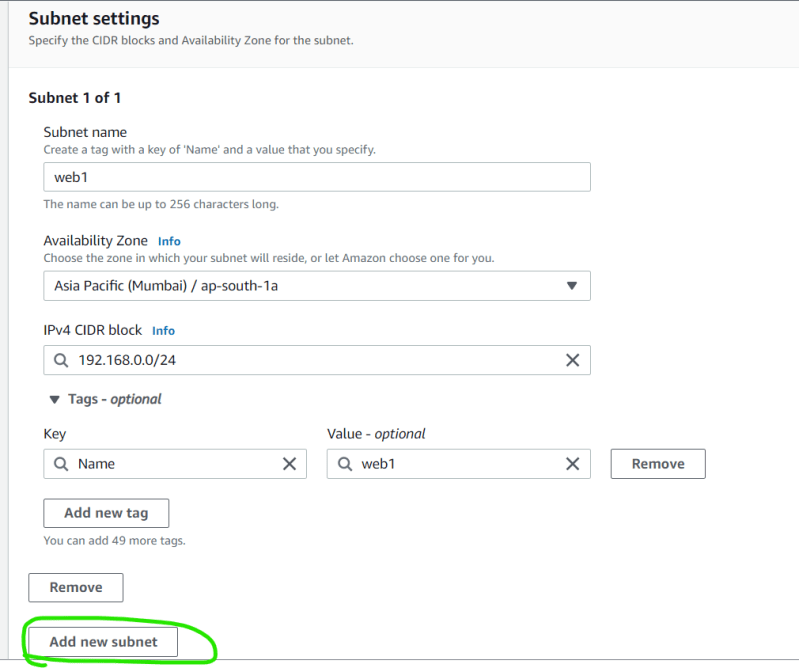
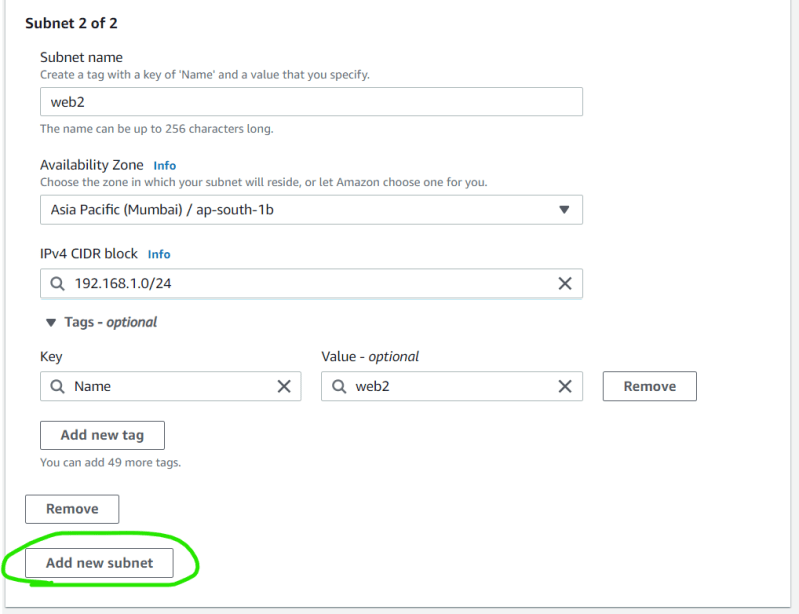
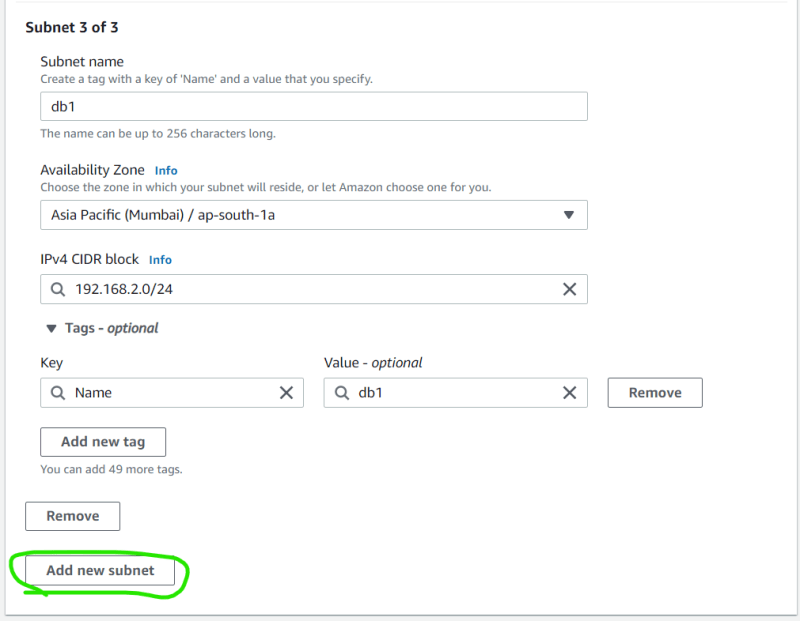
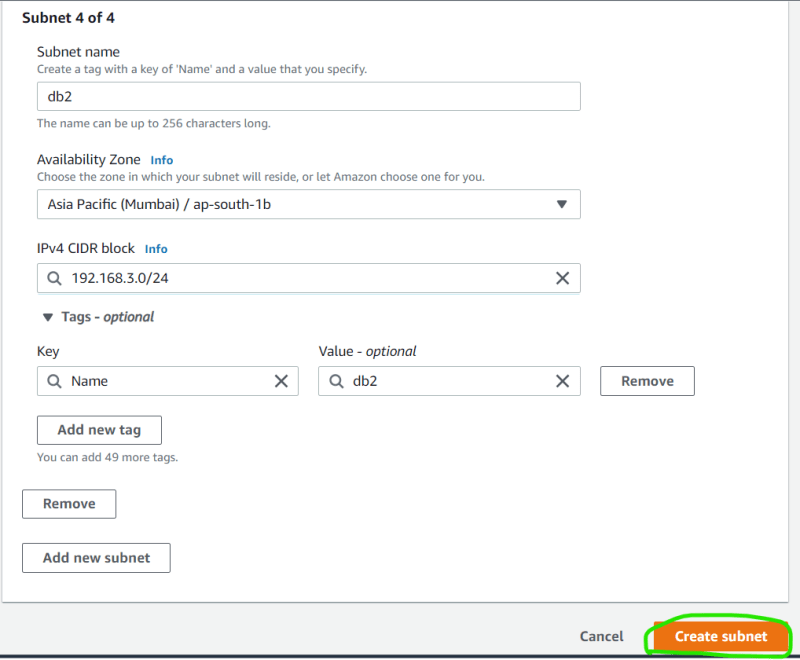
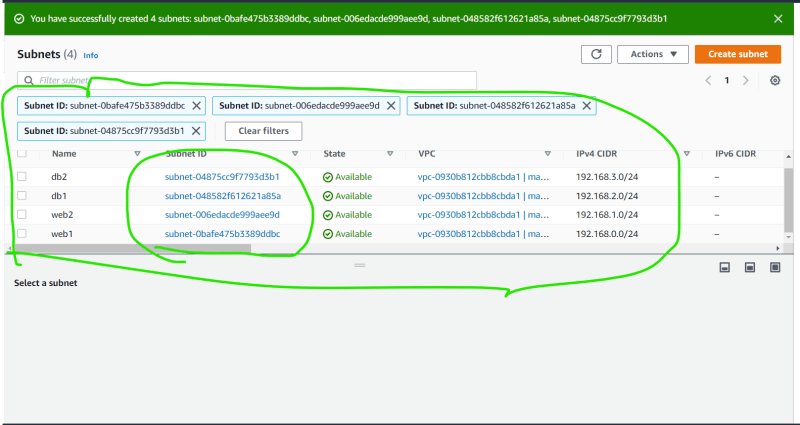
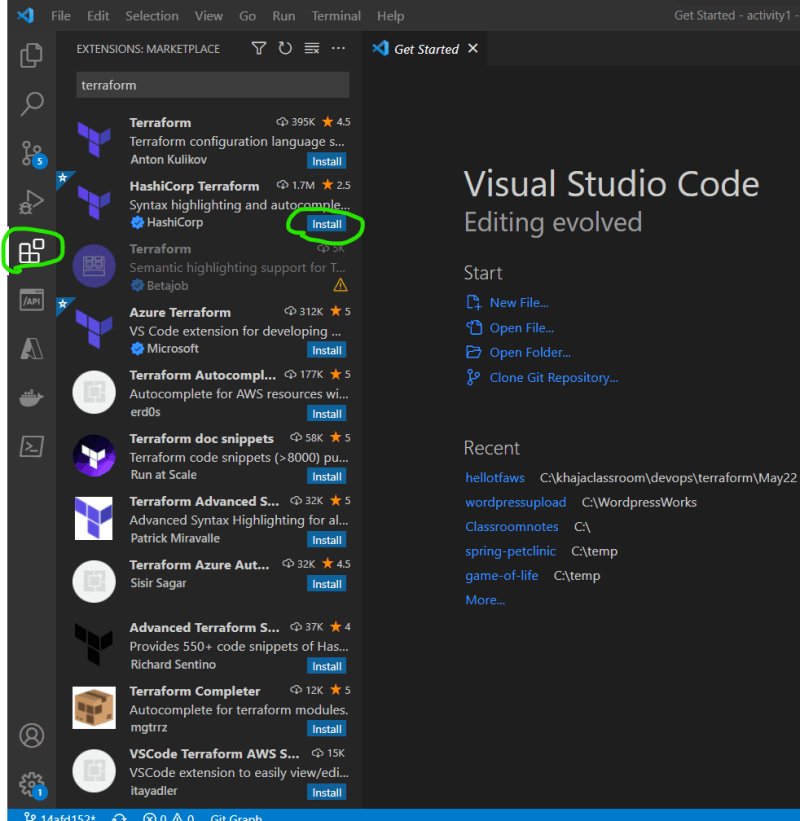
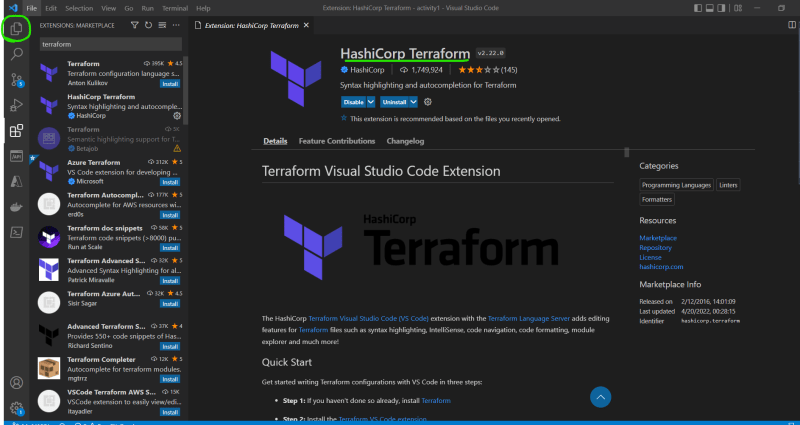
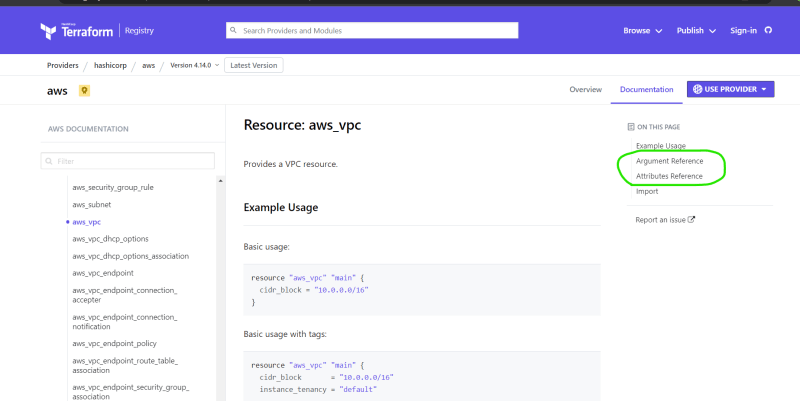
MAY 19, 2022

# DevOps Classroom notes 19/May/2022

## Terraform contd

* The inputs provided to the Resources/Data sources is called as arguments and the outputs are referred as attributes

#### Activity 1: AWS Infrastructure Simple

* Let’s try to create the below infrastructure using terraform  
  
* Manual Steps  
    
    
    
    
    
    
    
    
    
    
    
  
* Provider = aws
* Resource = vpc, subnet
* Lets configure visual studio code to help us with terraform  
    
  
* When terraform commands are executed generally they scan all the .tf files in the directory and execute the configuration.
* [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/vpc) for the vpc documentation  
  
* Lets define the provider and vpc definition

provider "aws" {

access\_key = "LKJLKSKLJDALDJLKSADSLA"

secret\_key = "lksdfjdlkasfjlsadfjlksdafjlksdafjdallksafj"

region = "ap-south-1"

}

# lets try to define the resource for the vpc

resource "aws\_vpc" "myvpc" {

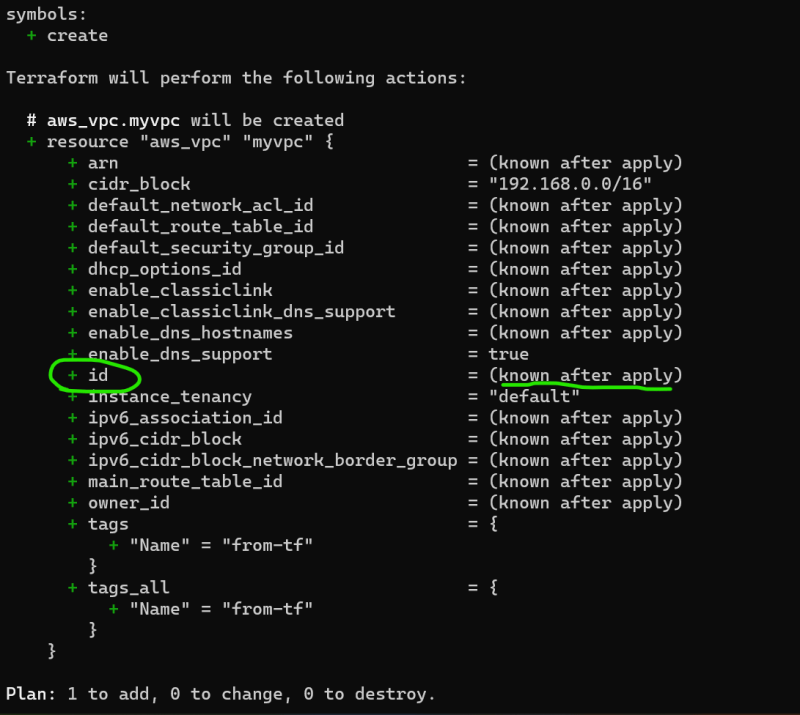
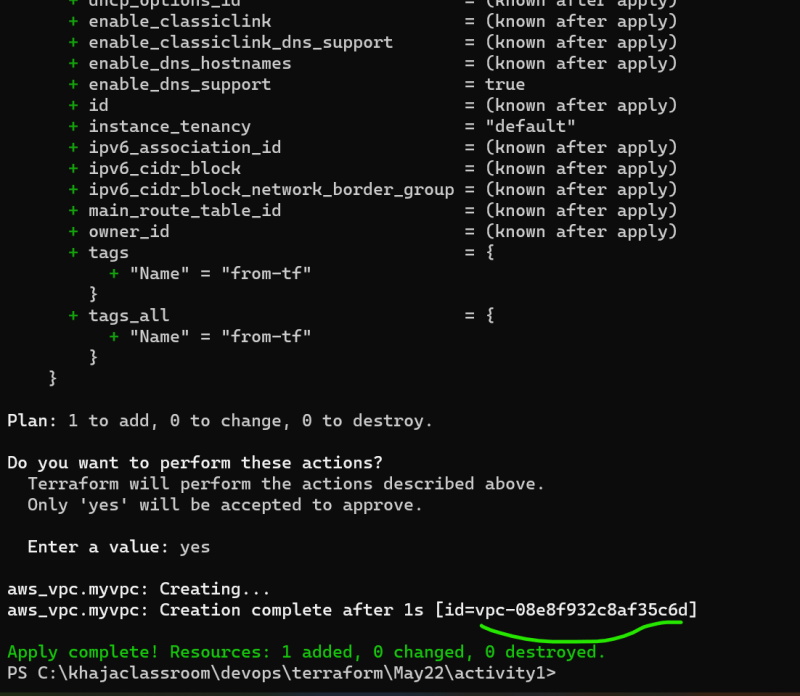
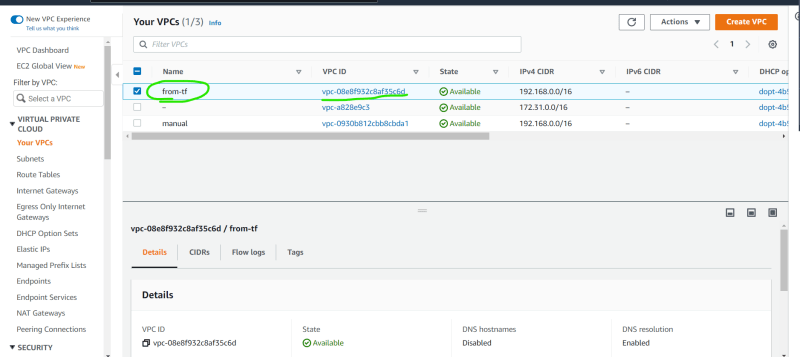
cidr\_block = "192.168.0.0/16"

tags = {

"Name" = "from-tf"

}

}

* Now perform init and validate. Now lets apply terraform to create infra terraform apply  
    
    
  
* Now lets try to add web1 and web2 subnets [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/subnet) for the subnet resource documentation
* The vpc id is the attribute of the resource block to access attributes the syntax is <PROVIDER>\_<TYPE>.<NAME>.<ATTRIBUTE-NAME>
* We have added the following for the subnets

# lets create web1 subnet

resource "aws\_subnet" "web1" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "192.168.0.0/24"

availability\_zone = "ap-south-1a"

tags = {

"Name" = "web1-tf"

}

}

# lets create web2 subnet

resource "aws\_subnet" "web2" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "192.168.1.0/24"

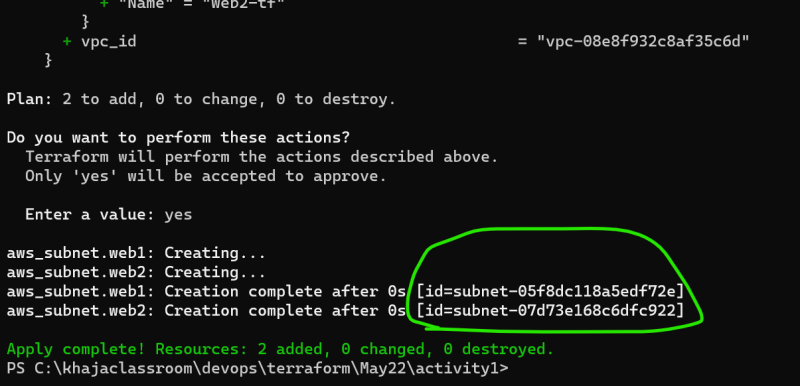
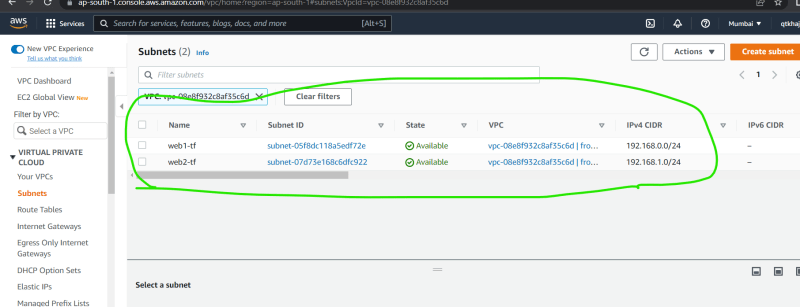
availability\_zone = "ap-south-1b"

tags = {

"Name" = "web2-tf"

}

}

* Now validate and apply  
    
  
* Now lets create db1 and db2 subnet

# lets create db1 subnet

resource "aws\_subnet" "db1" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "192.168.2.0/24"

availability\_zone = "ap-south-1a"

tags = {

"Name" = "db1-tf"

}

}

# lets create db2 subnet

resource "aws\_subnet" "db2" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "192.168.3.0/24"

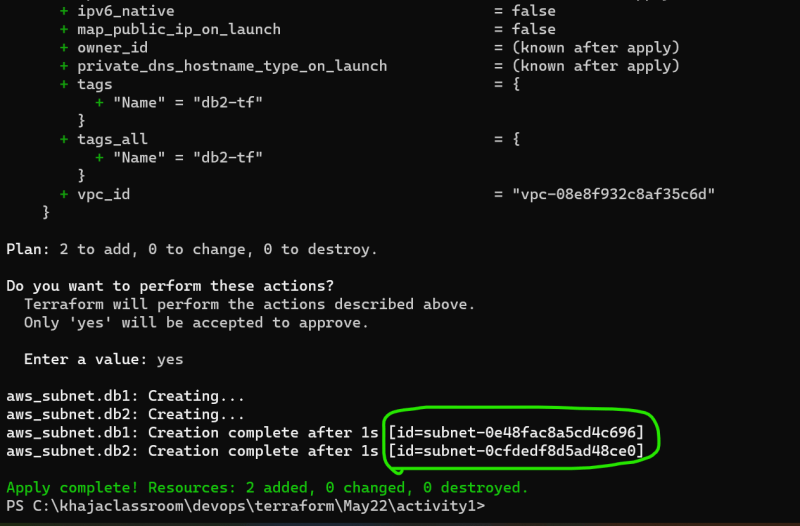
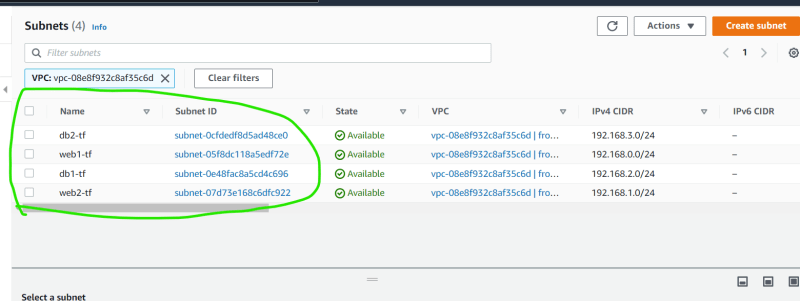
availability\_zone = "ap-south-1b"

tags = {

"Name" = "db2-tf"

}

}

* Now validate and apply  
    
  
* Next steps:
  + Let us improve the terraform configuration, which we have used to create the activity 1.

### Share this:

MAY 20, 2022

DevOps Class room notes 20/May/2022

**Terraform Contd…**

* The two major problems with the terraform configuration code which we have written are
  + secrets are directly written (access key and secret key)
  + for creating subnets we have similar resource definition used multiple times
* Then the next set of problems are, reason for writing terraform templates are to make the infrastructure as code and reusable. But our template always
  + creates the infra in mumbai region
  + the cidr ranges are fixed
  + In my account
* Terraform aws provider configuration:
  + [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs#authentication-and-configuration)
  + Lets try to install aws cli [Refer Here](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)
  + Now execute aws configure and enter access key secret access key and region ap-south-1
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/d370f71f4e1244c4ed54547d66b7b36f2070a48f) for the terraform script used to create the vpc.
* To solve the problem of using similar resource definition multiple times for subnet, first we need to make the terraform script reusable.
* In terraform to enable the parametrization of the argument values, we can use input variables [Refer Here](https://www.terraform.io/language/values/variables)

**Input Variables**

* Syntax:

variable "<var-name>" {

type = "<type of the variable>"

default = "<default value>"

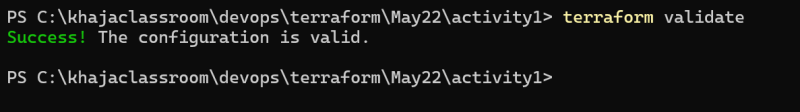
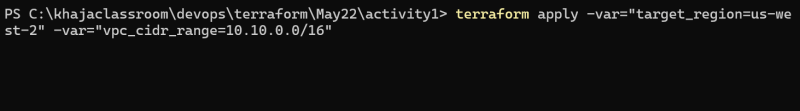
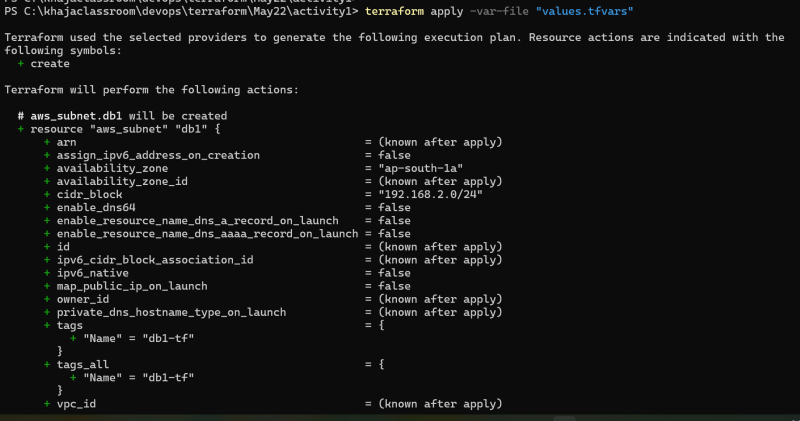
description = "description of the variable"

validation = "A block to validate the values"

sensitive = "Sensitive information"

nullable = "specifies if the variable can have null value"

}

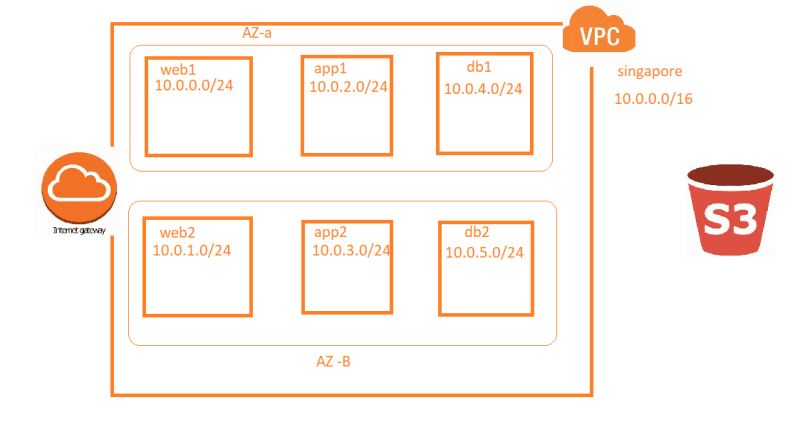
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/707c9d568f7ea219c5e2c8a3270f6306b27606ca) for the variable defintions added
* We need to use the variables the syntax is var.<variable-name> [Refer Here](https://github.com/asquarezone/TerraformZone/commit/0c421d281094afd51927e94a43111b4029ef362c) for the changes
* Now lets try to validate the terraform script  
  
* To pass the variables from command line [Refer Here](https://www.terraform.io/language/values/variables#variables-on-the-command-line)  
  
* If we have lot of variables to pass then variable defintions file is the option [Refer Here](https://www.terraform.io/language/values/variables#variable-definitions-tfvars-files).
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/3903b4a6fdd923ba051f9fa0194ae3688bbda3d1)  
  
* Lets add more variables and [Refer Here](https://github.com/asquarezone/TerraformZone/commit/64b47431d6a07f84666277225e21cc9fced7fbbc) for the changes
* Now execute terraform apply -var-file "default.tfvars"

**Share this:**

MAY 23, 2022

DevOps Classroomnotes 23/May/2022

**Exercise: Create the following using terraform**

* Create the vpc, s3 bucket  
  
* Create the internet gateway and attach to the vpc
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/538f2f4085409fdb7624ffe232be08a69e4c9b10) for the solution created in the class.

**Share this:**

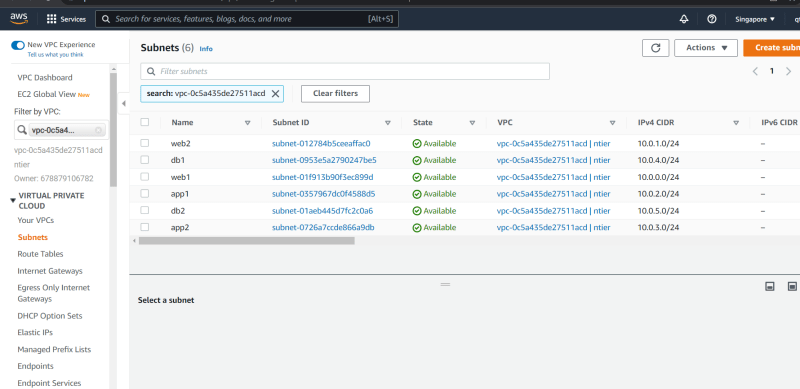
* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2022/05/23/devops-classroomnotes-23-may-2022/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2022/05/23/devops-classroomnotes-23-may-2022/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2022/05/23/devops-classroomnotes-23-may-2022/?share=jetpack-whatsapp&nb=1)

**Like this:**

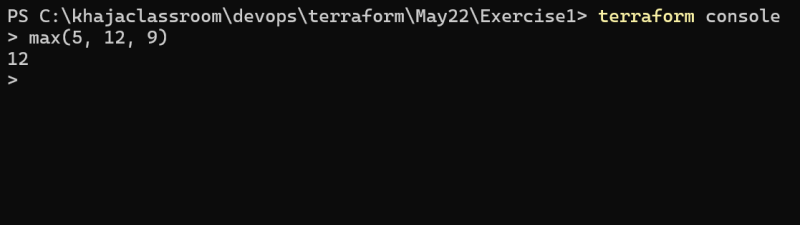
MAY 24, 2022

DevOps Classroomnotes 24/May/2022

**Terraform contd**

* Variable types: variable types supported by terraform are
  + string
  + number
  + bool
  + list
  + map
  + set
  + object
  + any
* Terraform offers loops
  + loops with count paramter
  + loops with for\_each expression
  + loops with for expression
* Loops with count parameter: [Refer Here](https://github.com/asquarezone/TerraformZone/commit/4cba883dccbebcfe6ecb9ea97a227cc3aa851331?diff=split) for the usage of count parameter  
  

**Terraform Functions**

* Terraform has lot of built in functions to help us with some logics/functionality [Refer Here](https://www.terraform.io/language/functions)
* To understand functions, explore them using terraform console  
  
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/d8c65ab90edc58fc1343c6247c935775846a6038?diff=split) for the usage of length function
* Exercise: What does coalesce and coalescelist functions do?
* Used cidrsubnet function to dynamically calculate the cidr ranges [Refer Here](https://github.com/asquarezone/TerraformZone/commit/88d3df4445615726511f19a015dafc4a7c2e8546?diff=split)

**Terraform Expressions**

* Conditional Expressions [Refer Here](https://www.terraform.io/language/expressions/conditionals)
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/8fe03a4d0a97f74d26adb1b0e7a3fd8eae9f852e?diff=split) for the usage of conditional expression.

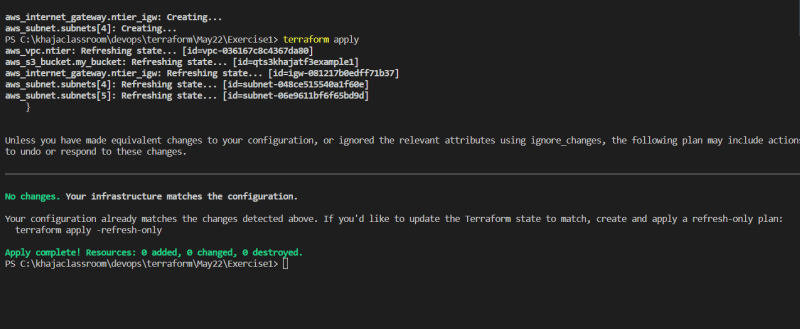
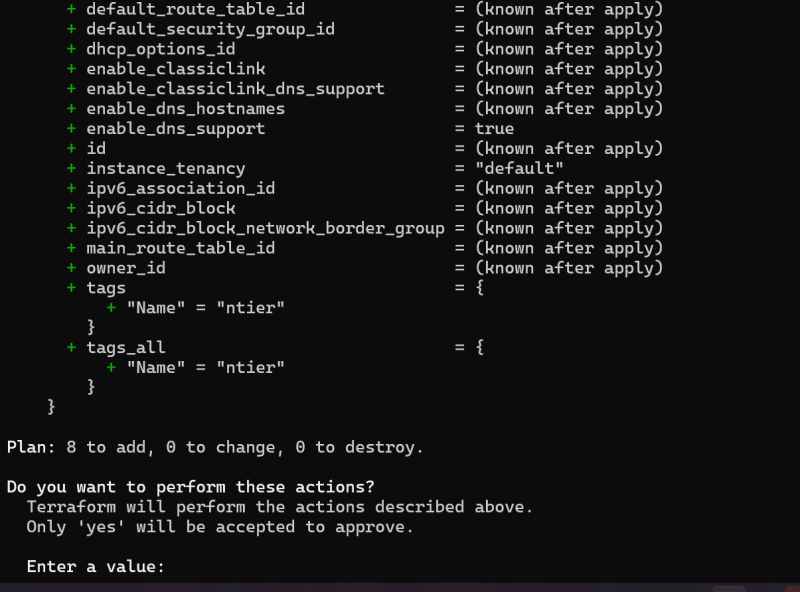
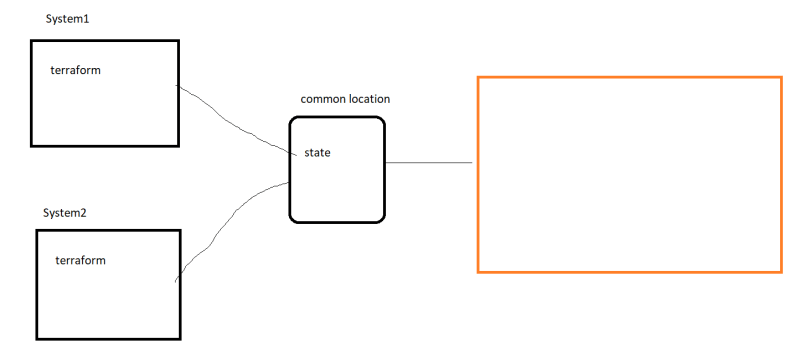
Exercise: try to understand all the types of terraforms with examples [Refer Here](https://www.terraform.io/language/expressions/types)

**Share this:**

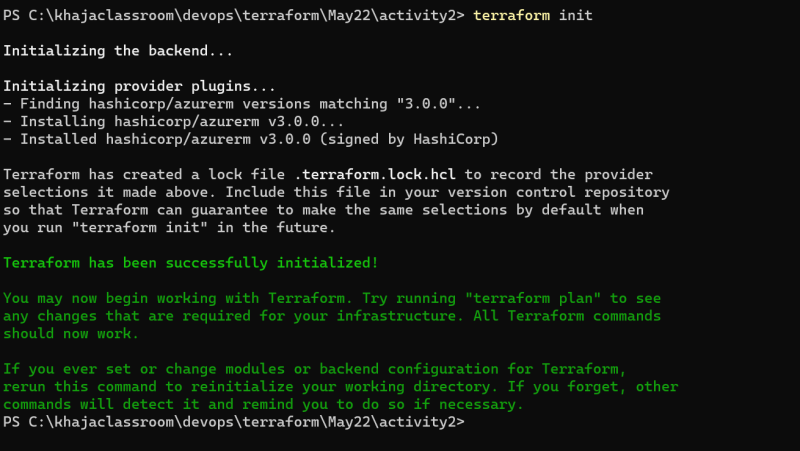
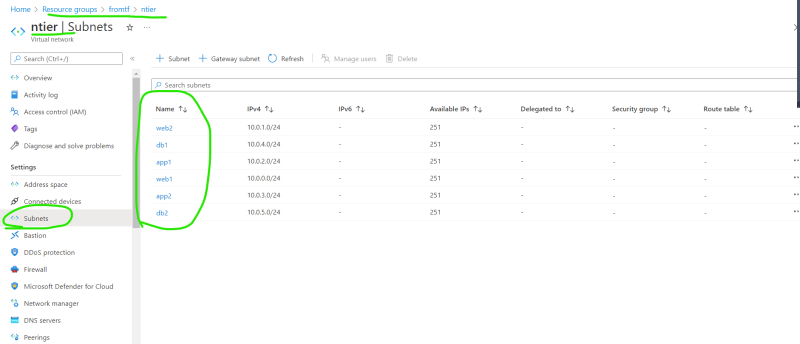
MAY 26, 2022

DevOps Classroomnotes 26/May/2022

**Terraform State**

* Terraform maintains the information of resources created in the state file.
* This file contains a custom JSON format that records a mapping from the Terraform resources in your configuration files to a representation of those resources in the real world (Provider).
* When we execute terraform apply, terraform first fetches the information from provider about the resources in state file and compares that to the state file, if there any changes required to match what you want (expressed) in your configuration file, terraform executes them
* terraform state matches with existing resources  
  
* Now Delete the resources manually in AWS/Azure and execute terraform apply will try to create the resources again  
  
* Ideally terraform state should be stored in common location for multiple users to execute the terraform apply leading to same infra, we will be learning this very soon as part of terraform backends.  
  

**Activity 2: Try to create a network with six subnets in Azure and a storage account**

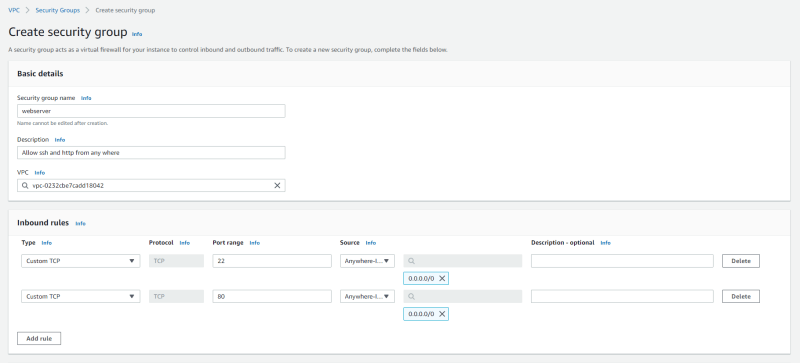
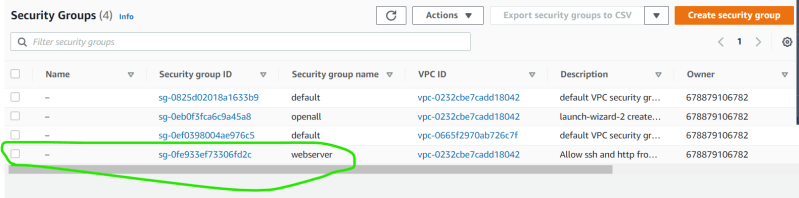
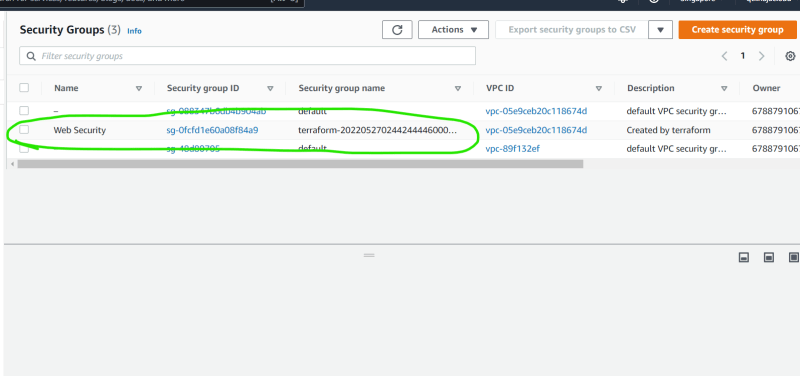
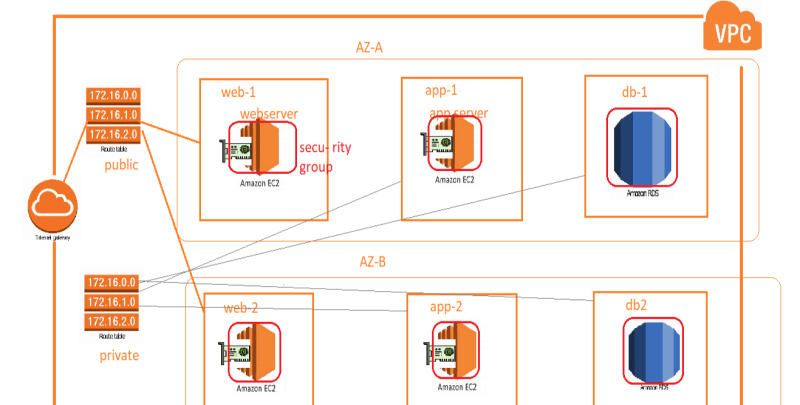
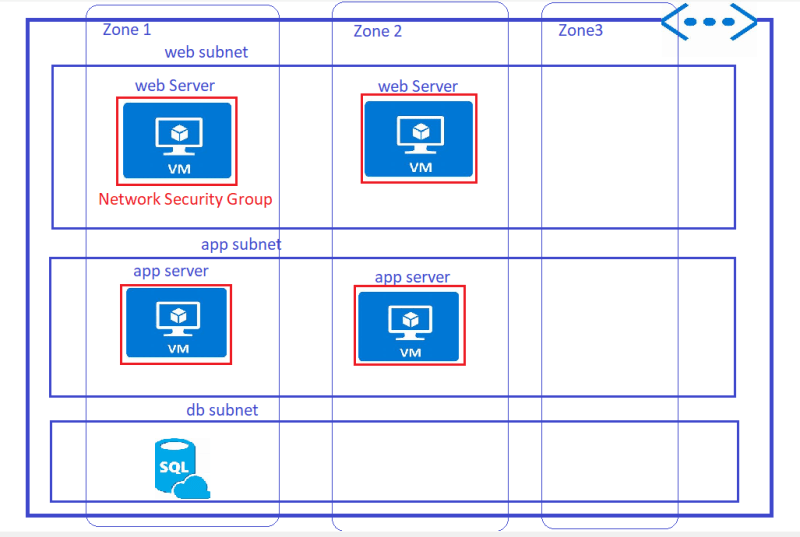
* In Azure to create any resources we need to first create a resource group
* Watch the classroom video for steps
* Now we need to configure provider first
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs) for the documentation
* Lets configure the authentication with Azure CLI, Lets install Azure CLI [Refer Here](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli)
* Once azure cli is installed. Relaunch powershell az login and login into azure account.
* Configure provider [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/azure_cli)  
  
* Lets configure resource group after provider configuration [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/resource_group)
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/virtual_network) for the vnet configuration
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/subnet) for subnet creation
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/b8326aed9faa8f37f22ac19a50375449378dc1e1) for the changes  
  

**Share this:**

MAY 27, 2022

DevOps Classroomnotes 27/May/2022

**Terraform contd..**

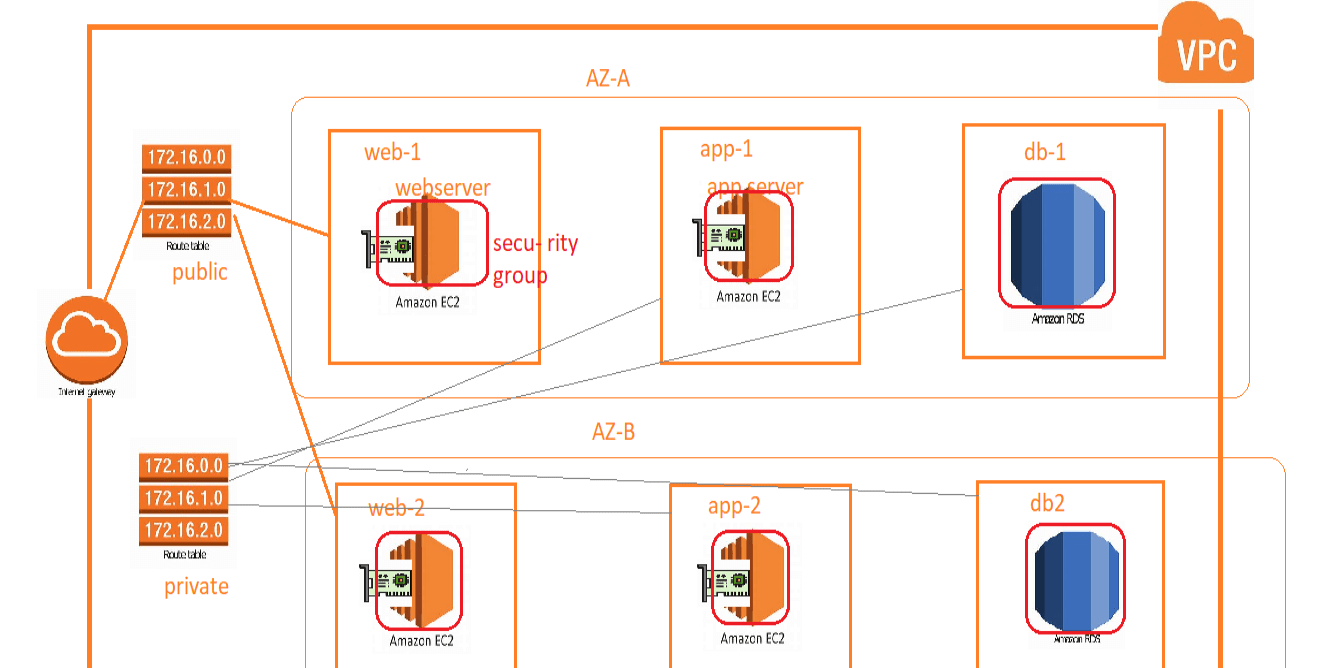
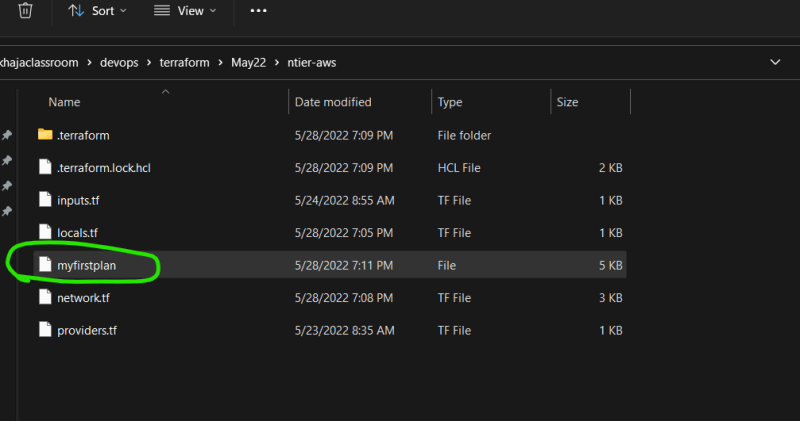
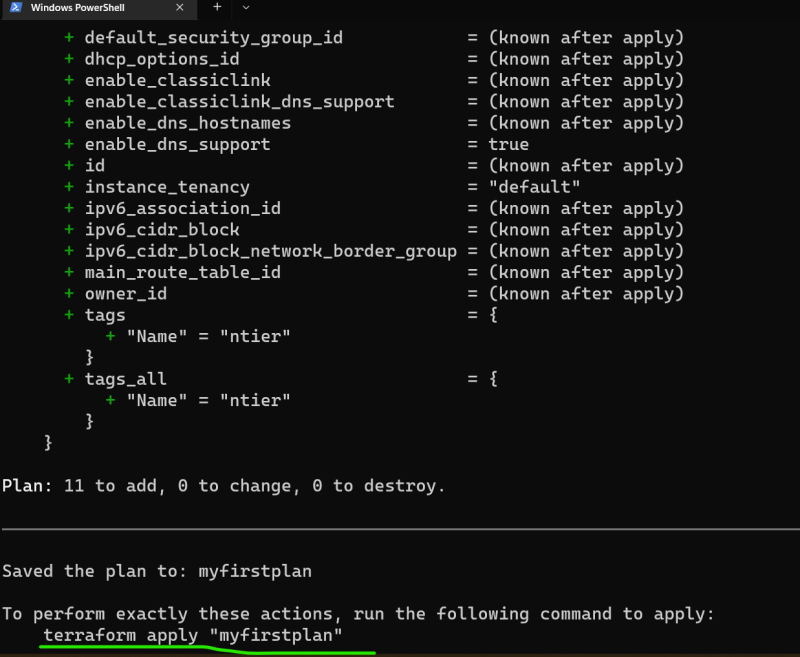
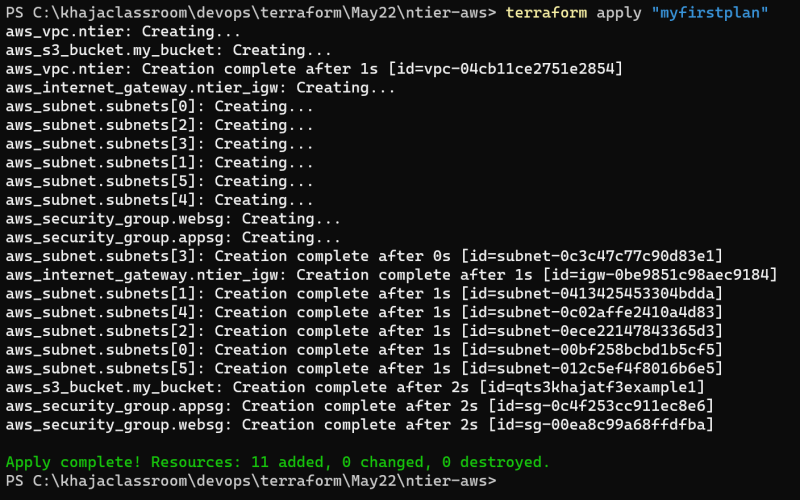
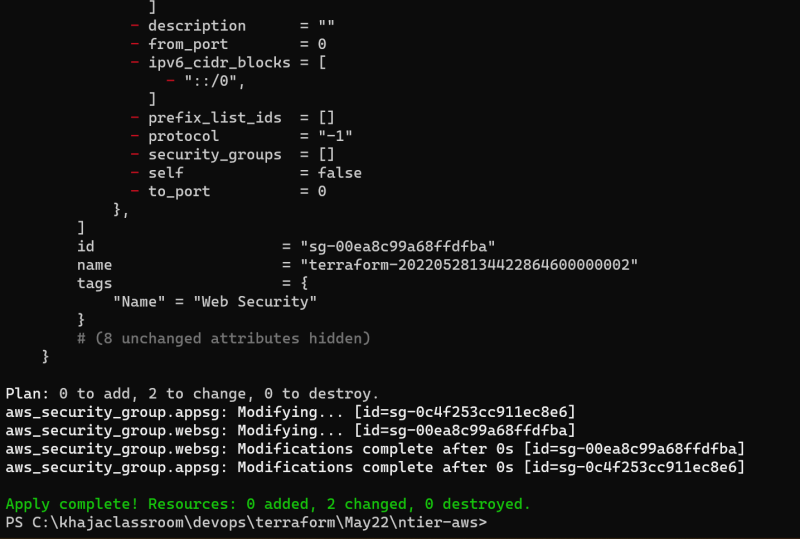
* Creating a Security Group in AWS, which is used as firewall for the virtual machines.
  + We need to open 22 port for ssh communication from anywhere (0.0.0.0/0)
  + We need to open 80 port as we will be using this for web servers
  + you can give the name of the security group as webserver  
      
    
* Write the terraform resource to create the same security group in the vpc created by terraform
* [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group) for the official docs
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/7b3d74f4df8d3086560a34f8d1db81b4c8ff0016?diff=split) for the changes done  
  
* Now lets assume in our application architecture we use
  + web servers => apache which runs on linux machine and can be accessed over port 80
  + application servers => tomcat which runs on some linux machine and can be accessed over port 8080
  + databases servers => mysql which runs on port 3306
* We need to create two more security groups
* AWS Architecture for ntier application  
  
* AZure Architecture for ntier application  
  

**Share this:**

MAY 28, 2022

DevOps Classroomnotes 28/May/2022

**AWS Infra with Terraform**

* We would like to realize the following architecture  
  
* [Refer Here](https://www.terraform.io/language/values/locals) for the Terraform local variables.
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/15736d40c7349370e17faa399f13c79da0e56b73) for the changeset containing local variables defined and their usage
* when we execute terraform apply a plan gets created depending on what has to be created. Generally when we apply terraform apply the plan gets created internally, we can create the plan externally as well.
* Now execute `terraform plan -out “myfirstplan”  
    
  
* Now execute terrform apply “myfirstplan”  
  
* Now we have a small fix for security egress rules [Refer Here](https://github.com/asquarezone/TerraformZone/commit/ba753651c8b2896e8ff029398aa3cdc062c23c22?diff=split) for the changes. Execute the application by following command  
  terraform apply -auto-approve  
  
* Now lets add public and private route tables [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table) for the resource documentation.
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/1c0089894350ef56a2d7bfdcf085866958ba1ba6?diff=split) for public and private routetables added into terraform
* Now we need to associate route tables with subnets. [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table_association) for the resource documentation
* One approach is to use this

resource "aws\_route\_table\_association" "associations" {

count = length(aws\_subnet.subnets)

subnet\_id = aws\_subnet.subnets[count.index].id

route\_table\_id = count.index<2 ? aws\_route\_table.publicrt.id : aws\_route\_table.privatert.id

}

* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/036ae3083aa8011b49e71de3a5e6718733d895c6?diff=split) for the usage of condition with variable with configurable public subnets, by default subnets not defined in public\_subnets variable are private [Refer Here](https://github.com/asquarezone/TerraformZone/commit/036ae3083aa8011b49e71de3a5e6718733d895c6?diff=split) for the changes done.
* Lets add db security group as well [Refer Here](https://github.com/asquarezone/TerraformZone/commit/c21a20817d945ce1d26e390a3ee301f0756ad621) for the changes
* To Create explicit dependecies, we use depends on [Refer Here](https://www.terraform.io/language/meta-arguments/depends_on). [Refer Here](https://github.com/asquarezone/TerraformZone/commit/3858feeedce9010fb54c7f8bb7eb189560e5cbc6?diff=split) for the changes done

**AWS RDS Creation**

* Manual Steps: refer class room video
* As tried we need to have the following before we create rds
  + Security group
  + Subnet Group
* We already have the Security Group, lets create the db subnet group
* If we need to pull information from aws regarding any resource we can use data source which is like query to a provider [Refer Here](https://www.terraform.io/language/data-sources)
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/34ff8b2d34a01af5e7d2e716059838b3a7480e39?diff=split) for the usage of datasources and creating db subnet group
* Note: from now on i will not be creating variables/locals until and unless it is important. I would expect you to change that while practicing
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/b01fb873732a531737330be5e34cd9f3d5398ad5?diff=split) for the changeset to rds instance in our vpc

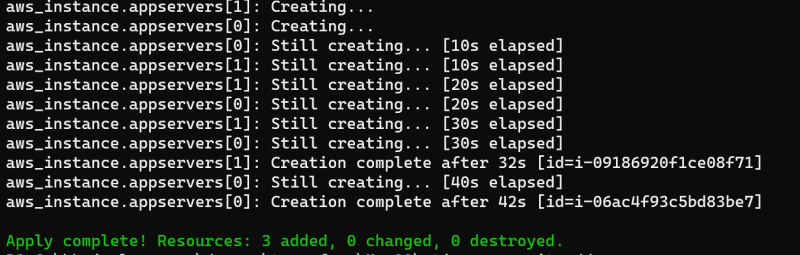
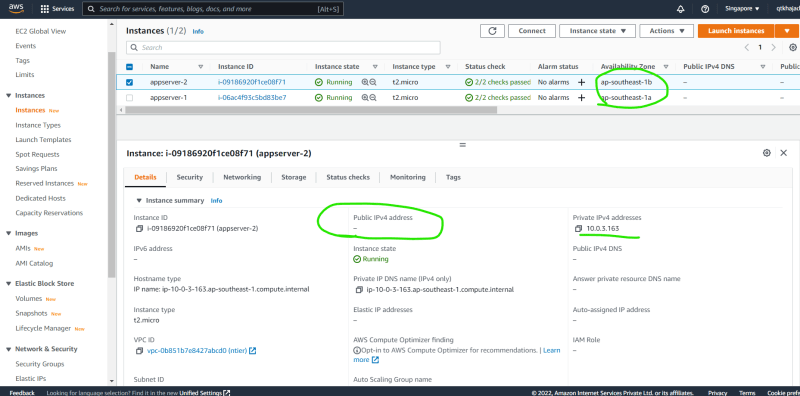
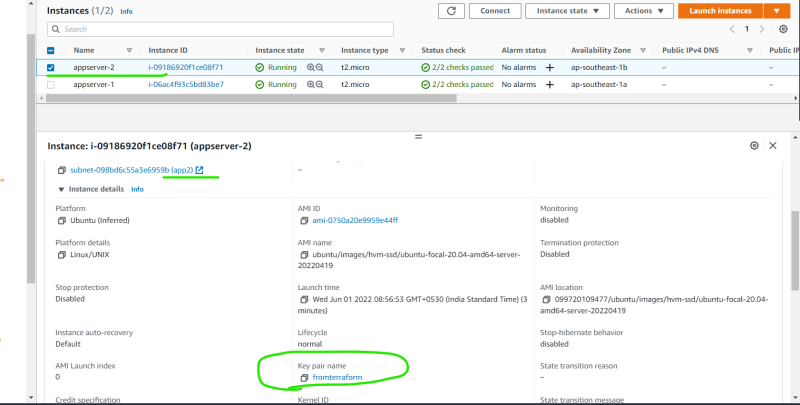
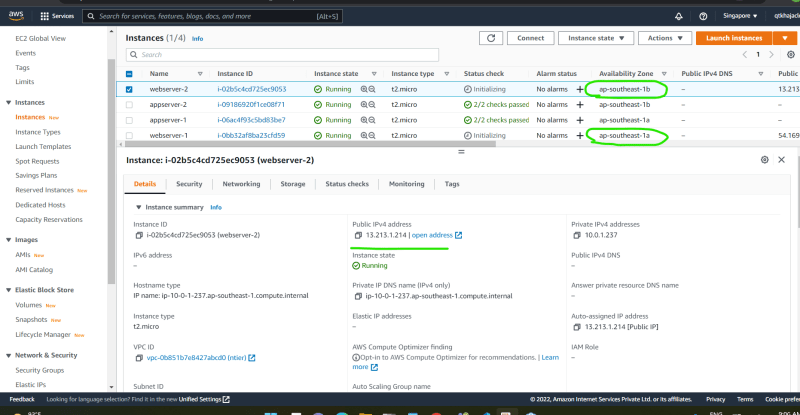
**Share this:**

* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2022/05/28/devops-classroomnotes-28-may-2022/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2022/05/28/devops-classroomnotes-28-may-2022/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2022/05/28/devops-classroomnotes-28-may-2022/?share=jetpack-whatsapp&nb=1)

JUNE 1, 2022

DevOps Classroomnotes 01/Jun/2022

**Terraform with AWS Contd**

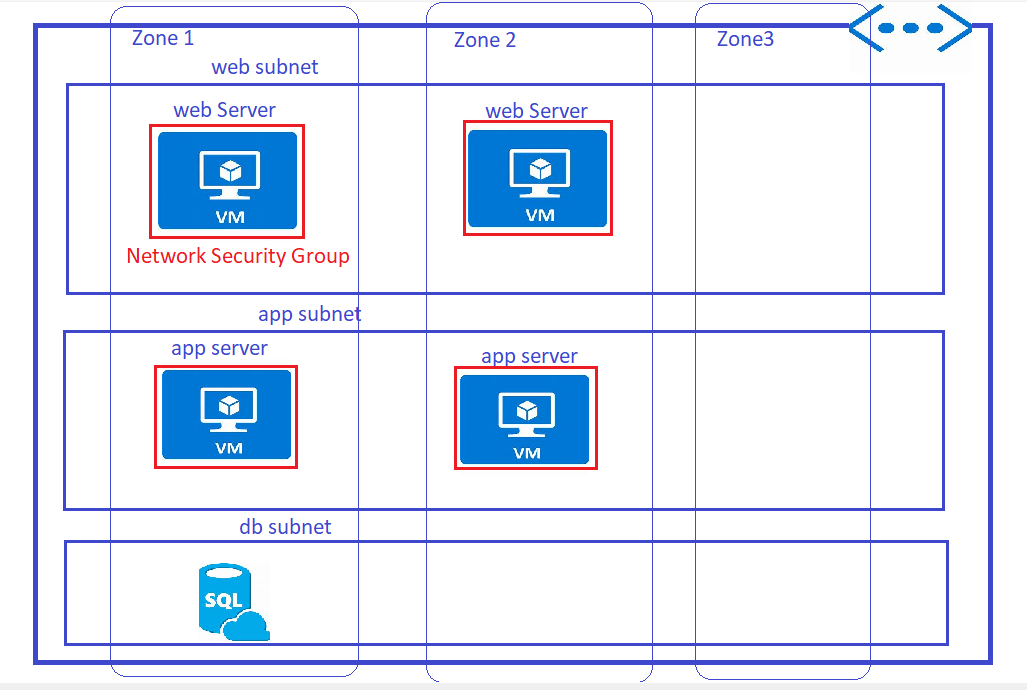
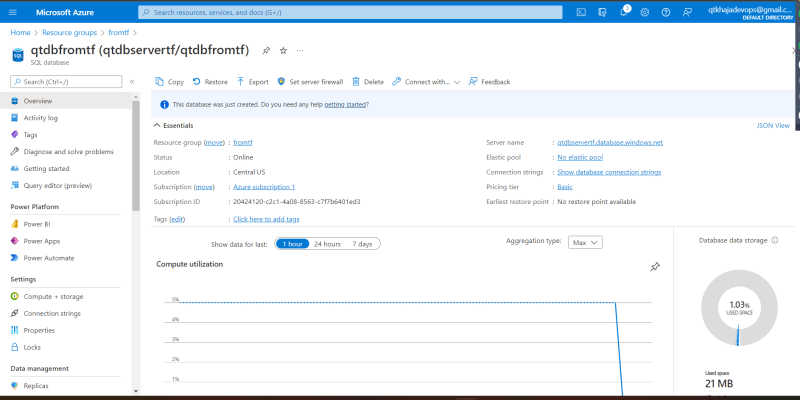
* The necessary inputs for creating an ec2 instance are
  + AMI id
  + instance type
  + security group
  + subnet-id
  + name tag
  + key pair
* Lets get started with key pair.
* Ensure you have a key pair created with ssh-keygen
* Used a object to declare the necessary information of the variable for server creation and add the ec2 instance [Refer Here](https://github.com/asquarezone/TerraformZone/commit/fb81fdfc033cd763ac19be8bb67de1426e807137?diff=split) for the changeset and execute terraform apply  
    
    
  
* Now lets create 2 web servers [Refer Here](https://github.com/asquarezone/TerraformZone/commit/e38c97958c2656b48d70e66f6a841d14375f1836) for the changes  
  

**Share this:**

JUNE 2, 2022

DevOps Classroomnotes 02/Jun/2022

**Terraform with Azure**

* Lets try to realize the below architecture  
  
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/8c5c9121f15d5032c07e459ac2a174db808dc963) for existing template
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_security_group) for the documentation of nsg
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/aeb5b0ce9b681308a3d9568da68173ea52b1cc52) for the changeset to add nsgs.
* We need to create a database (refer to class room video for manual creation)
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_database) for the terraform documentation on sql database and [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/mssql_server) for sql server
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/98625edfb39d61c7814304dff827dcd5adb70158) for the changes done to add a public database.  
  
* Exercise: refer the changeset above and add necessary local and input variables.

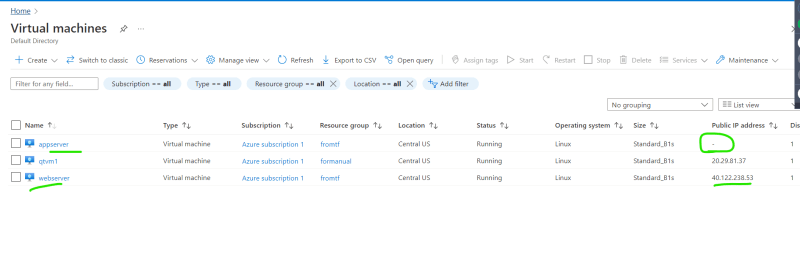
**Share this:**

* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2022/06/02/devops-classroomnotes-02-jun-2022/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2022/06/02/devops-classroomnotes-02-jun-2022/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2022/06/02/devops-classroomnotes-02-jun-2022/?share=jetpack-whatsapp&nb=1)

JUNE 3, 2022

DevOps Classroomnotes 03/Jun/2022

**Terraform with Azure Contd..**

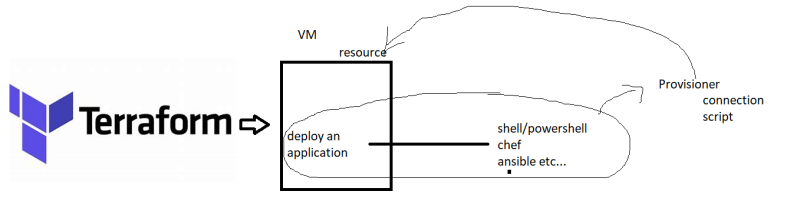
* Lets create virtual machines in azure
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux_virtual_machine) for the resource documentation.
* We need to have the following to create vm
  + subnet
  + nsg
  + nic
* So lets try to create nic [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_interface) for the documentation
* For creating public ip, [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/public_ip)
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/e9c7a50999b642cd52170d628388aad7518ff5d3) for the changeset containing all the info for creating the vm  
  
* Note: please refer to class room video for manual steps
* Next Steps:
  + Installing sofwares in vm’s based on existing automation
  + Making multiple users work with terraform configuration
  + creating multiple environments with the same terraform configuration
  + taint’s etc

**Share this:**

JUNE 4, 2022

# DevOps Classroomnotes 04/Jun/2022

## Terraform Provisioning

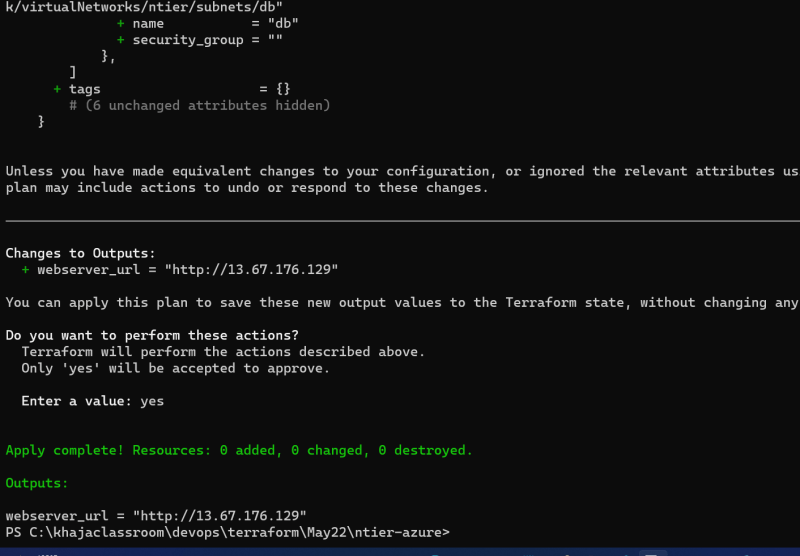
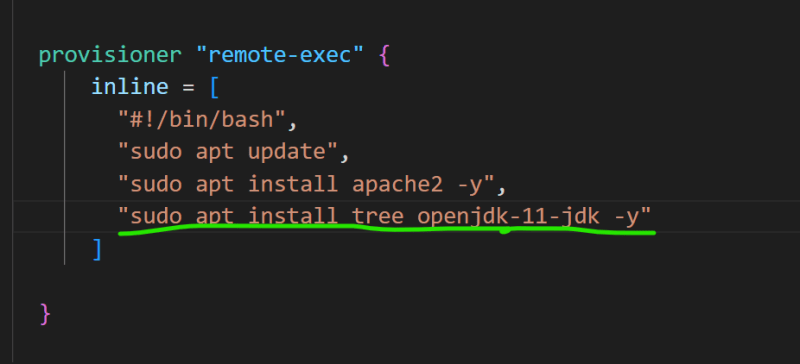
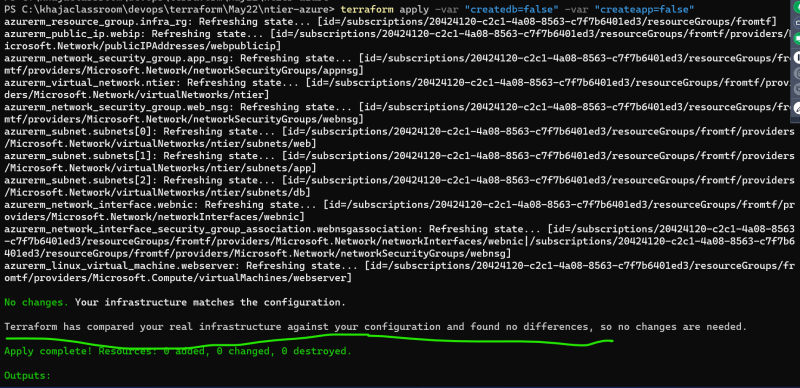
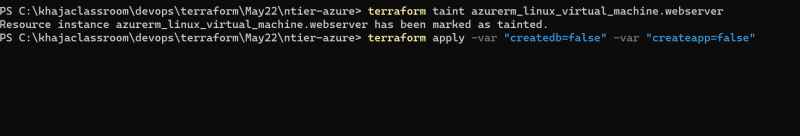
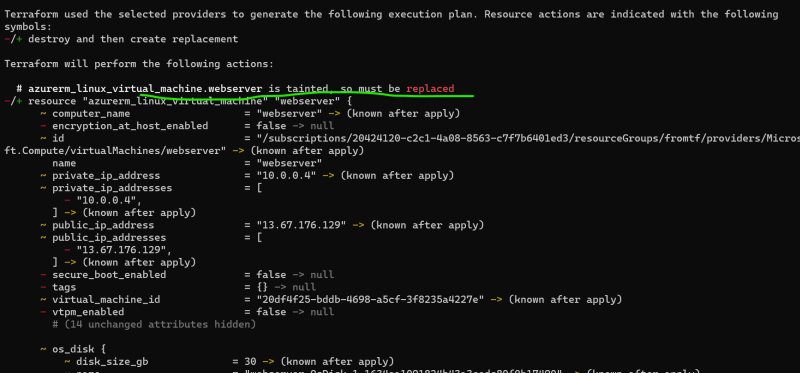
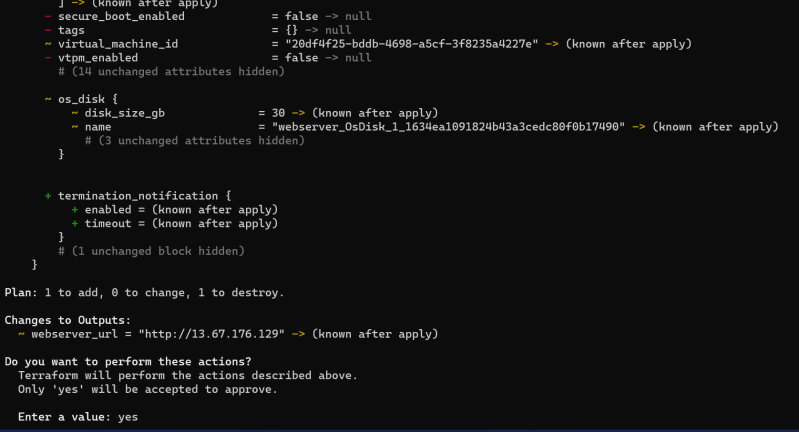
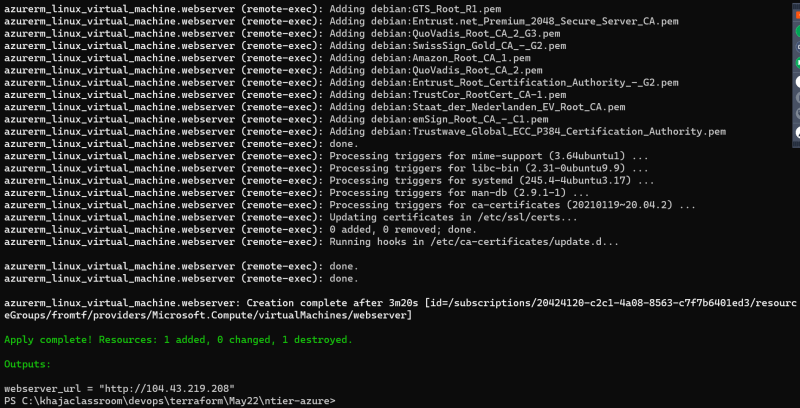
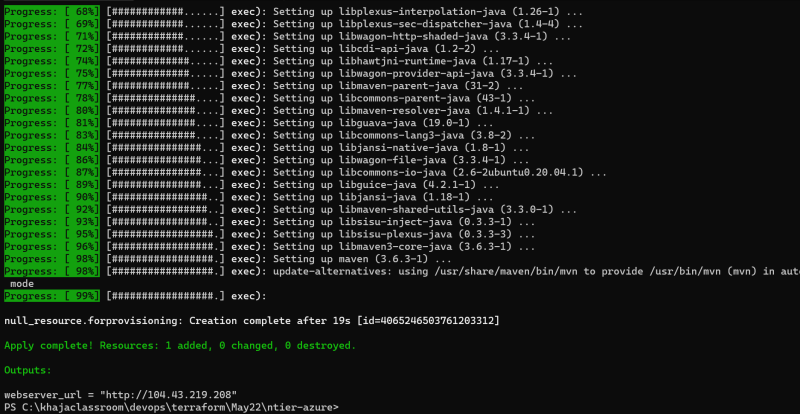
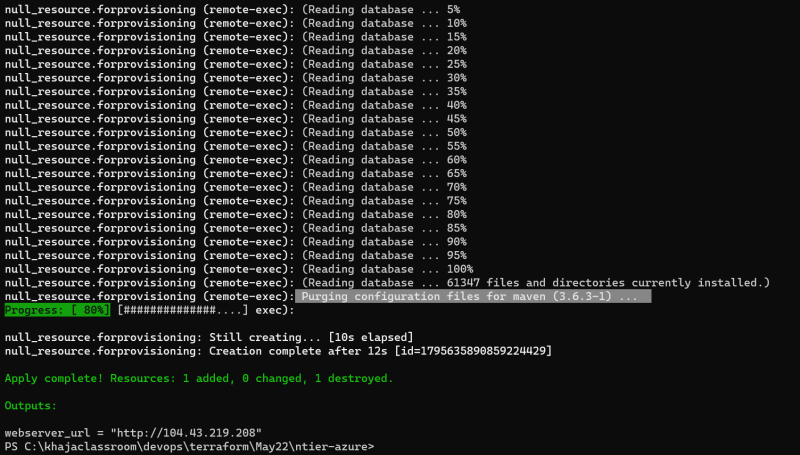
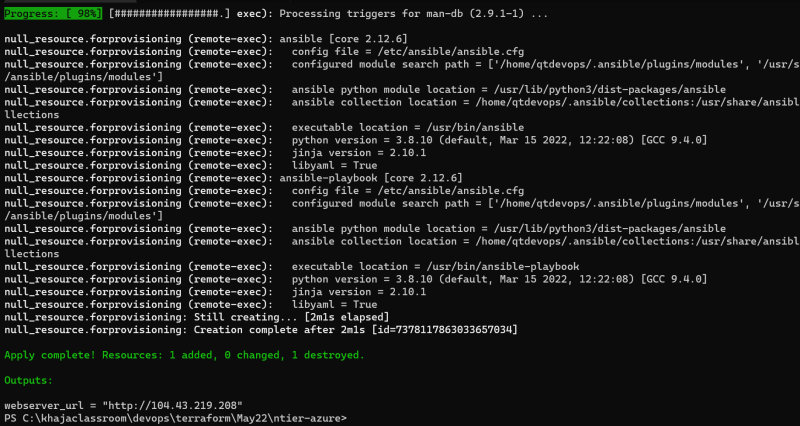
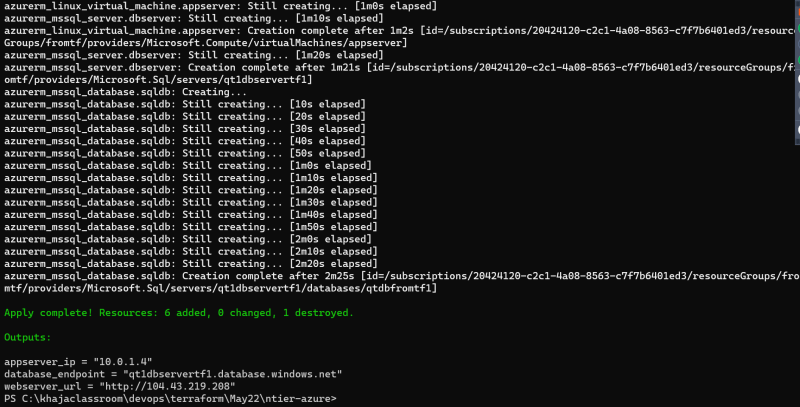
* Provisioners can be used to model specific actions on the local machine or on a remote machine in order to prepare server or other infrastructure objects at scale.
* Concept of provisioning  
  
* Provisioner types:
  + file
  + local-exec
  + remote-exec
  + chef
  + salt
  + puppet
* [Refer Here](https://www.terraform.io/language/resources/provisioners/syntax)

### Activity: Lets install the following on the web vm

* Lets install apache server

sudo apt update

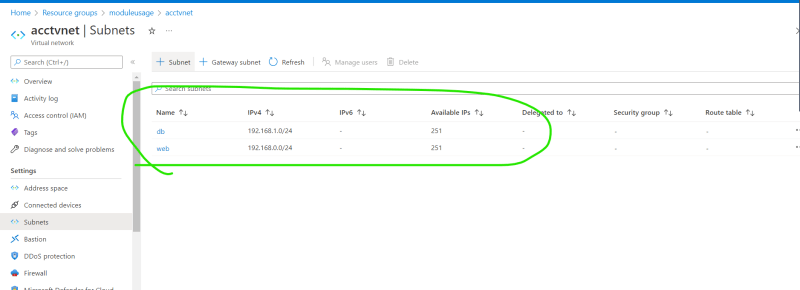
sudo apt install apache2 -y

* Lets do this in Azure (/AWS)
* Lets create connection
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/5d4a069c88e4371dfdbef6049c8c3adbb1a54a11) for the changes done to install apache2 on web vm and conditionally create app and db servers
* Situation1: To get the public ip from vm, we had to go to azure portal. [Refer Here](https://www.terraform.io/language/values/outputs) for the output documentation. [Refer Here](https://github.com/asquarezone/TerraformZone/commit/63db20fbdc0f0a109e64e916fb66d811f20bf033) for the changes done  
  
* Situation 2: We need to change the script and execute the installation of some other software
  + The VM is already created
  + Lets change the script and rexecute  
    
  + The change in script of provisioning is not considered as change  
    
* Solution 1 to above problem:
  + Whenever you change any thing in script, recreate the virtual machine
  + We need to mark virtual machine for deletion during next apply this is referred as taint in terraform.
  + Lets taint the vm and execute apply  
      
      
      
    
* Solution 2: Execute provisioning all the times when you execute apply
  + But terraform executes provisioner only when resource is created.
  + Terraform has a resource called as null resource which it tries to create during apply and we run the provisioner over there  
    
  + One option is to mark the null resource for taint before apply (or when the provisioining needs to happen)
  + other option is to use triggers  
      
    
  + [Refer Here](https://github.com/asquarezone/TerraformZone/commit/2b00cc1d20aee0ef87adf15ba4267060412fdf3a) for the changes done.
* Exercise:
  + For the people aware of ansible try to install apache server from ansible playbook in terraform provisioning
  + Create the provisioner to install tomcat on appserver
* Made some subtle changes and we have the following output, [Refer Here](https://github.com/asquarezone/TerraformZone/commit/d2f408dfb56e7aabcad3107a83adb1a15942f6d2) for the changes  
  

### Activity: Create a reusable terraform configuration to create ntier

* Terraform modules can help in creating reusable configuration [Refer Here](https://www.terraform.io/language/modules/syntax)

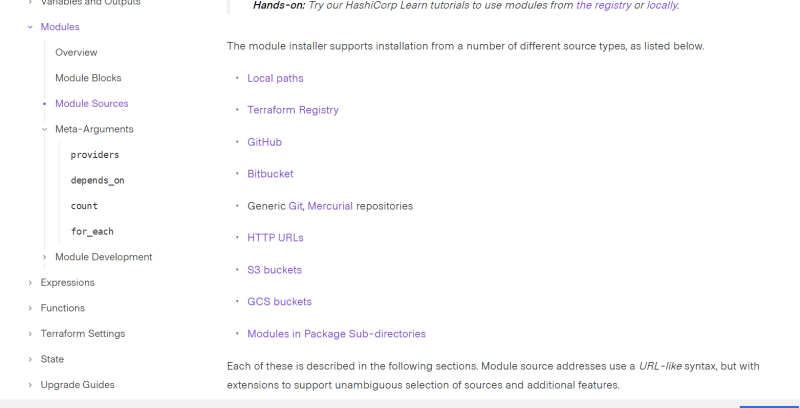
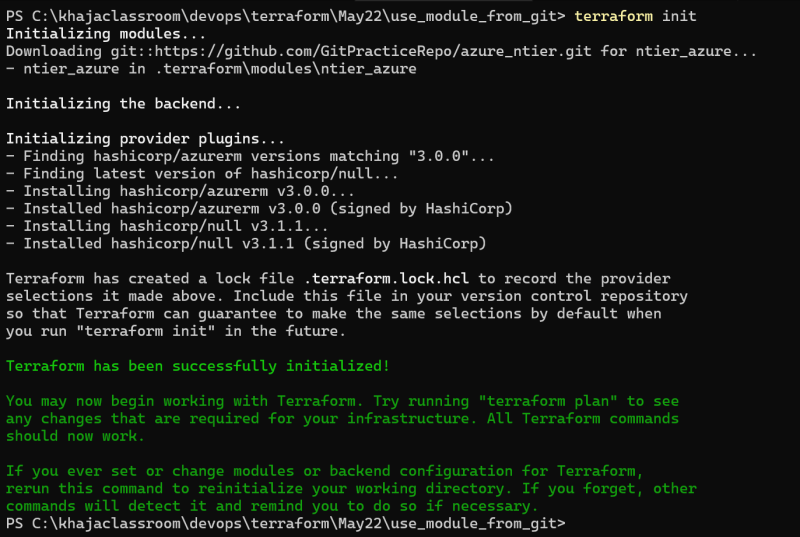
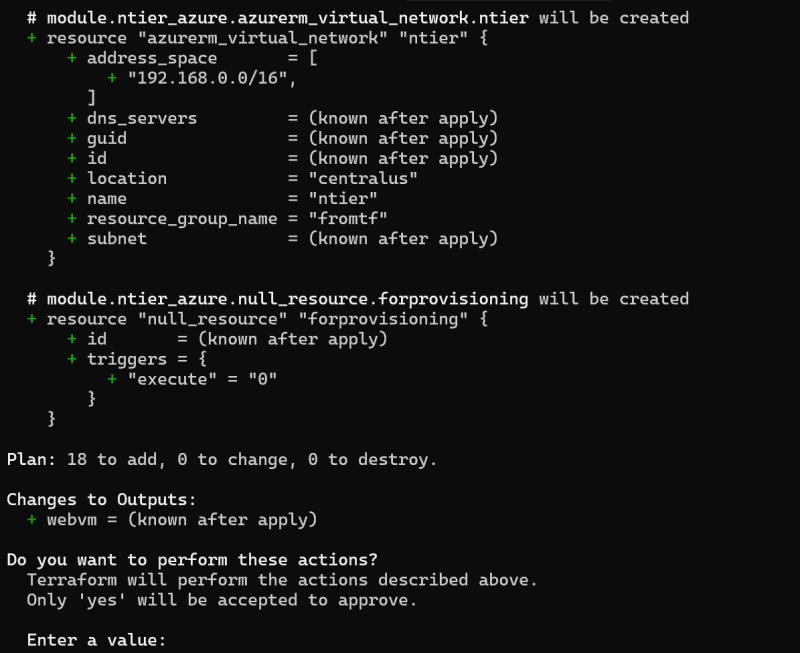
#### Situation 1: Lets try to use existing modules

* A lot of opensource contributers create reusable terraform configuration as modules and host them on various sources.
* Terraform registry is the popular one
* Lets try to use a module created by someone to create a network in Azure [Refer Here](https://registry.terraform.io/modules/Azure/network/azurerm/latest)for the network module
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/69458f6772d6356891a257f2cf2f2b04859f8ffd) for the changeset  
  
* The source code of the module which we have used is [Refer Here](https://github.com/Azure/terraform-azurerm-network)
* It is exactly similar what we have been developing
* A Module is a collection of terraform files
  + input variables become arguments
  + output variables become attributes of the module

#### Situation 2: Lets try to create module from our configuration

* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/9fc2e015a8a35aab4ecc88a5b99f0343e2a514b4) for the module which we added
* Structure recommendations of module from terraform [Refer Here](https://www.terraform.io/language/modules/develop/structure)
* We have used the source which is present in the same machine and we can use this source to access any drive (local/network) to gain access to reusable modules

#### Situation 3: How to make modules to be used by all of our team members

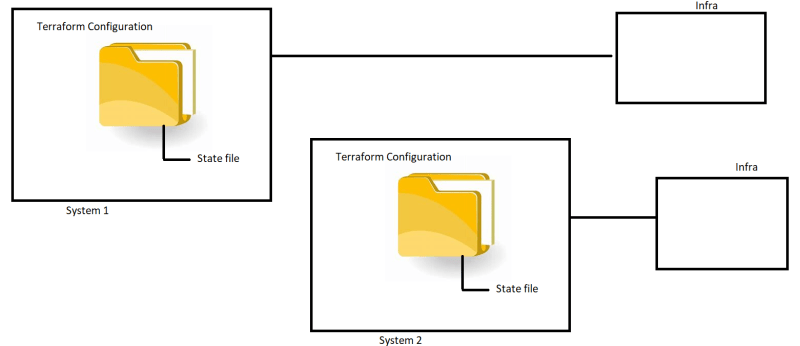
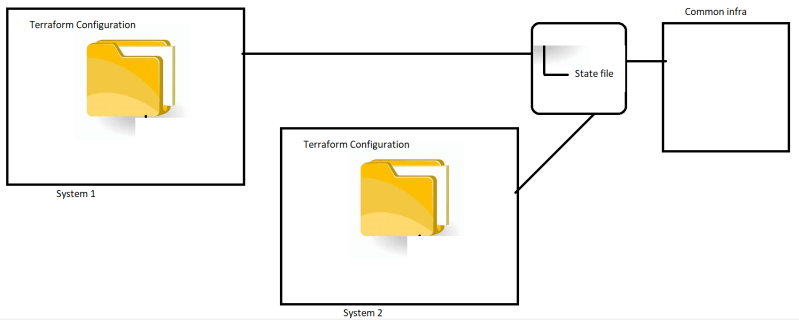
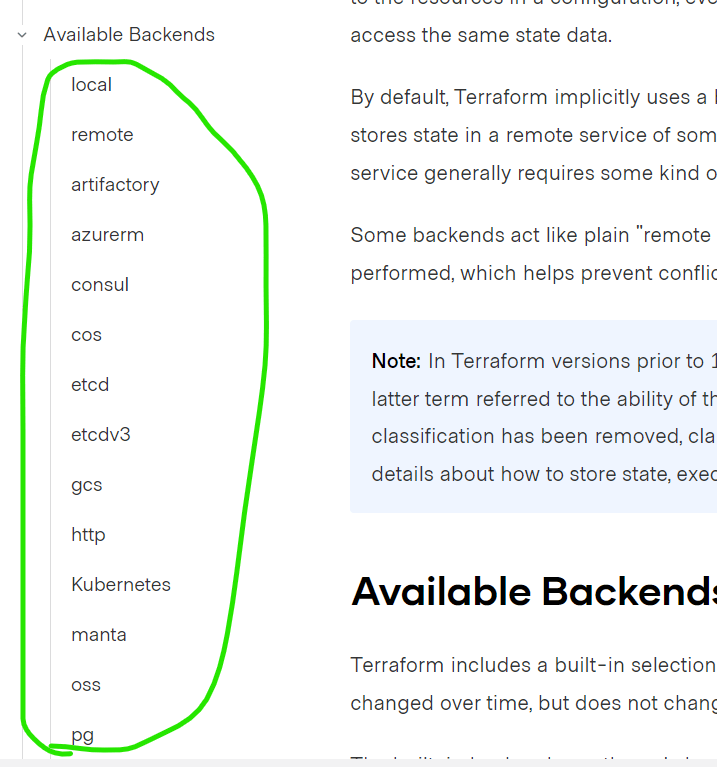
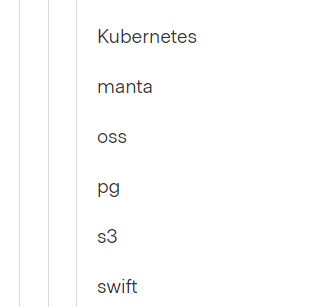
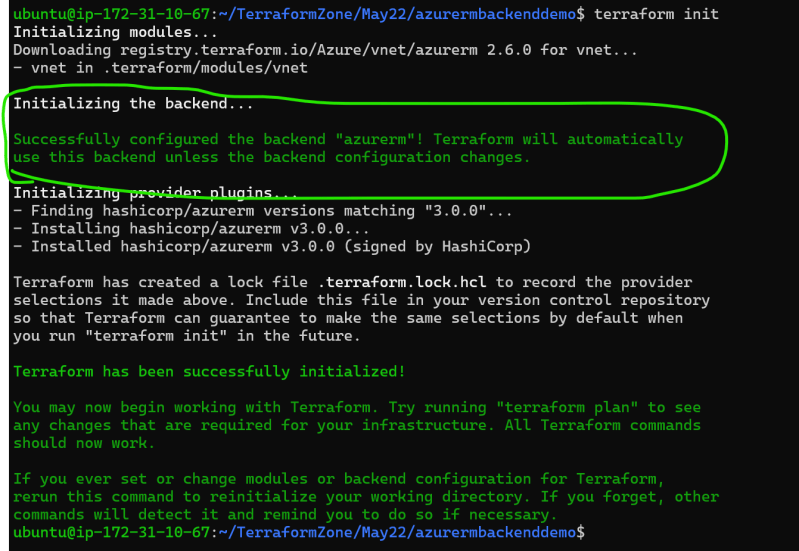
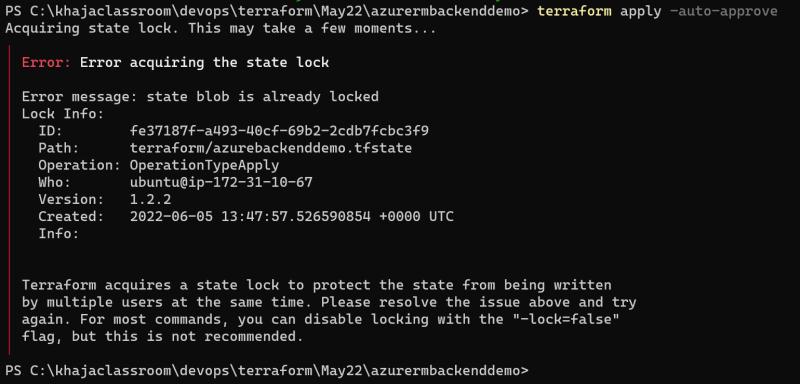
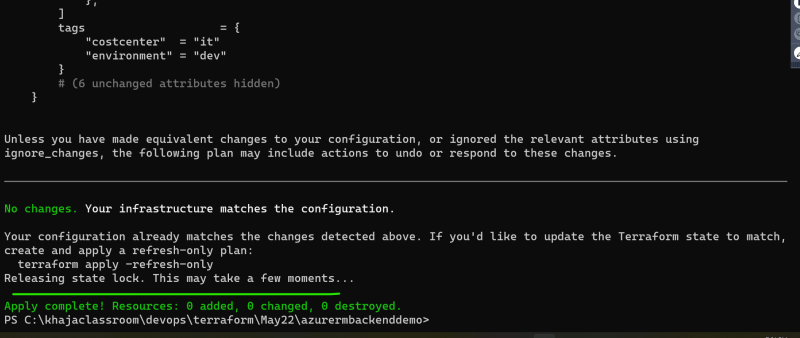
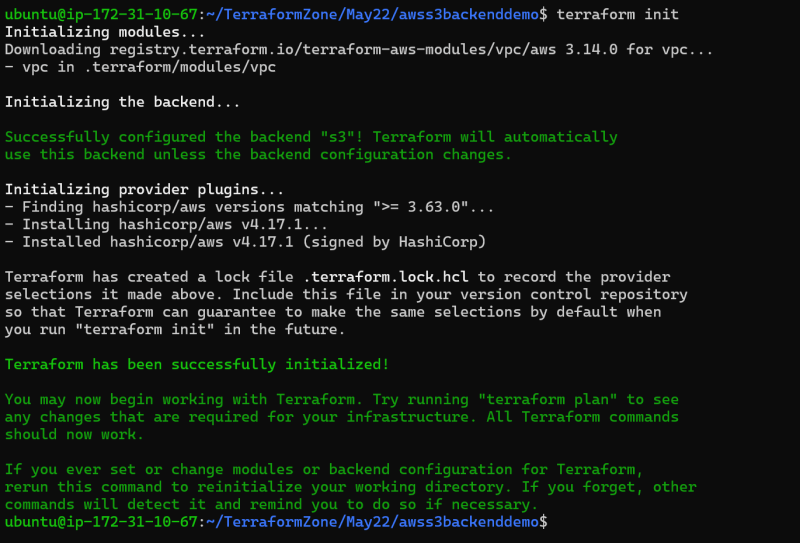
* One way is make them available on network file shares and access them
* We can use any of the module sources [Refer Here](https://www.terraform.io/language/modules/sources)  
  
* Create a git repository for every module which devops team creates [Refer Here](https://github.com/GitPracticeRepo/azure_ntier)
* for using module from git we have created the following changeset [Refer Here](https://github.com/asquarezone/TerraformZone/commit/ff50cdd2cfae2b3c13641f1bd8a71e4a30c1e8a5)
* Now execute init and apply  
    
  

### Share this:

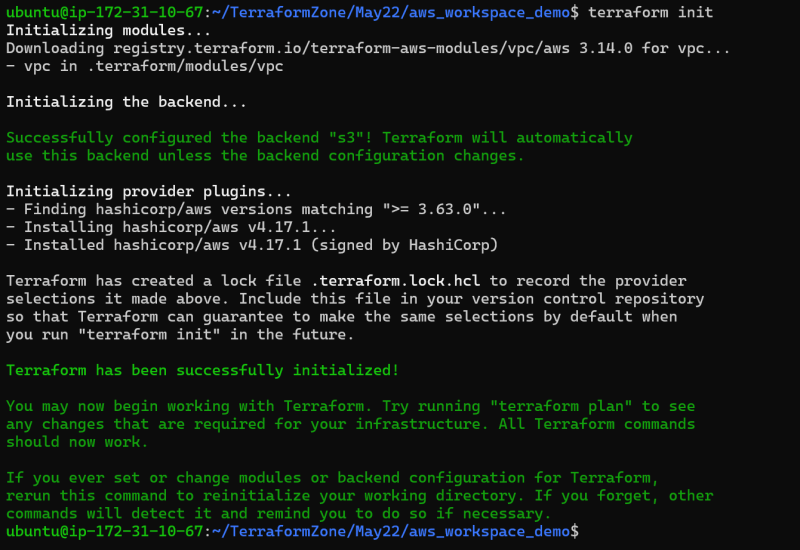
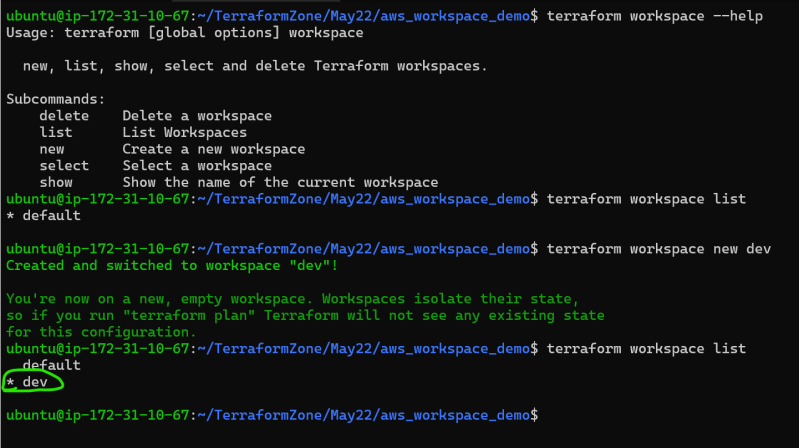
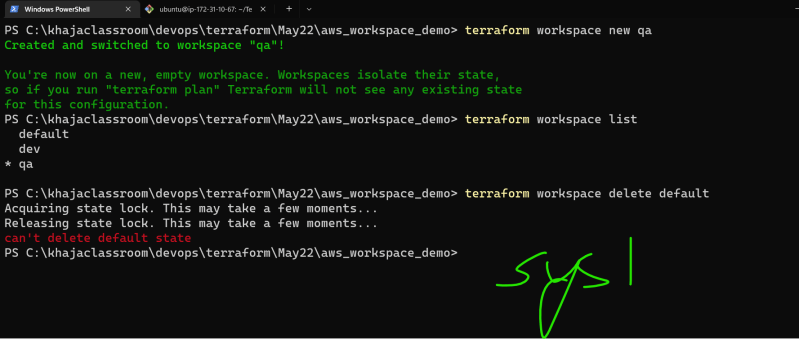
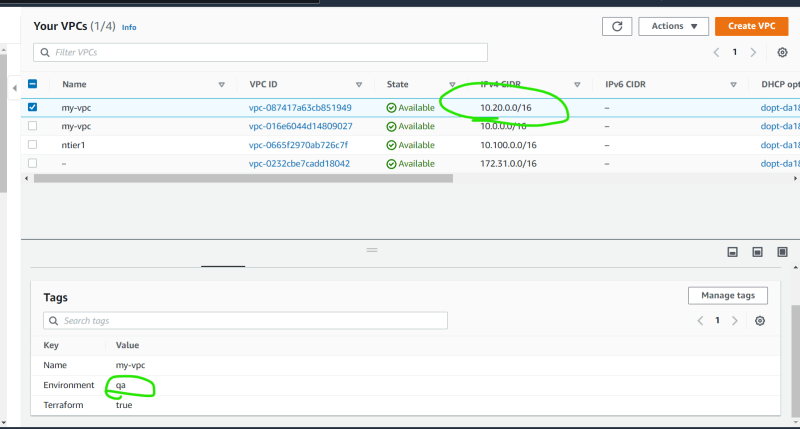
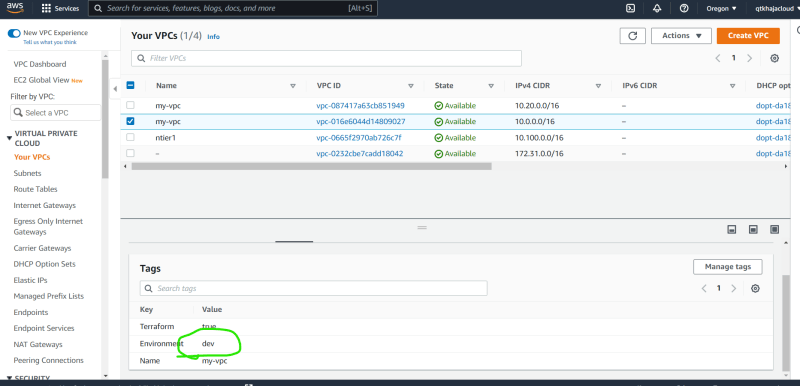
JUNE 5, 2022

DevOps Classroomnotes 05/Jun/2022

**Terraform backends**

* When two engineers execute the terraform configuration with a perspective to create infrastructure for an application, it creates two different infra.  
  
* What if these two engineers want to create the same environment, in this case the state should not be on the local system, it should be present on common location.
* This approach is called as backend. The default backend is local folder  
  
* [Refer Here](https://www.terraform.io/language/settings/backends) for the terraform backends  
    
  
* Lets create a common infra for multiple users using Azure RM backend  
  [Refer Here](https://github.com/asquarezone/TerraformZone/commit/0300e39e536f86f967bb9924ec2e444d81e43f8d) for the changes done
* For simulating multiple users i will be creating a linux instance, installing terraform and creating infra
* Now execute initialization  
  
* Now apply from system 2 and execute apply from system 1 as well, on system you should get an error message of lock acquired by some one else  
  
* After successful apply from system 2 now execute apply from system 1  
  
* AWS S3 Backend: We can use s3 backend, but for locking we need to use dynamo db. [Refer Here](https://www.terraform.io/language/settings/backends/s3)
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/a6da940c80082e65ffad5f57127a17221ed7ab5c) for the s3 backend changeset
* Initialize the backend and do the rest as done for azurerm backend  
  

**Workspaces**

* Using Terraform configuration to create multiple environmetns from the same configuration
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/aa3be76b05a36b5c6126355333f687d678ca8d19) for the changeset containing workspace demo features
* Lets initialize  
  
* Lets check for the workspaces available. create the dev on the system 2  
    
  
* Lets create dev from system 2 and qa from system 1 parallely
* Dev from system 2  
    
  

**Lifecycle**

* [Refer Here](https://www.terraform.io/language/meta-arguments/lifecycle) for the terraform lifecycle
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/41680e894d567f84962931911e0c5a74cbd7ab71) for the changeset

**Packer**

* Packer is a free and opensource tool to create images i.e we can create azure vm images, aws amis
* [Refer Here](https://www.packer.io/docs)
* [Refer Here](https://learn.hashicorp.com/tutorials/packer/get-started-install-cli) for installing packer
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/2b75156ca9489d261e348b4173be37599213d2fd) for the example to create ami with packer in aws
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/460836d584a4602b6169b917a17e96d8ac214b11) for the changes to create azure vm image

**Share this:**