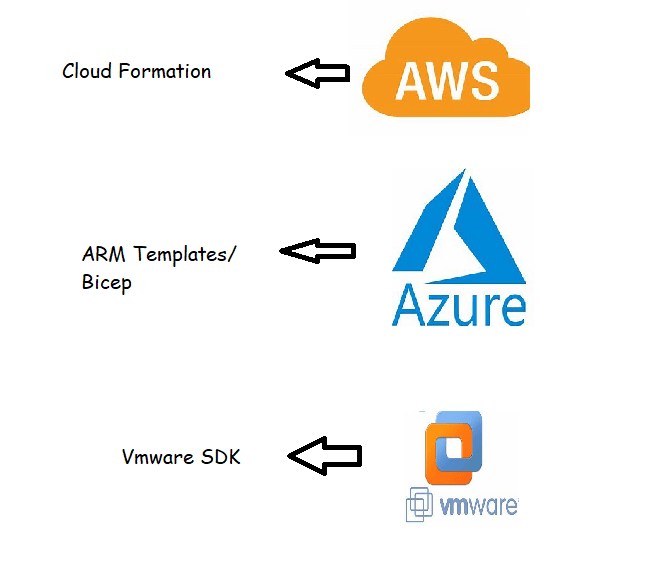
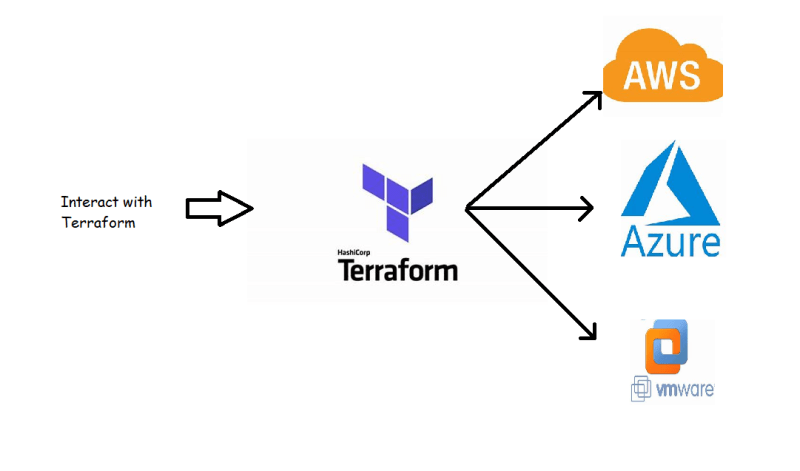
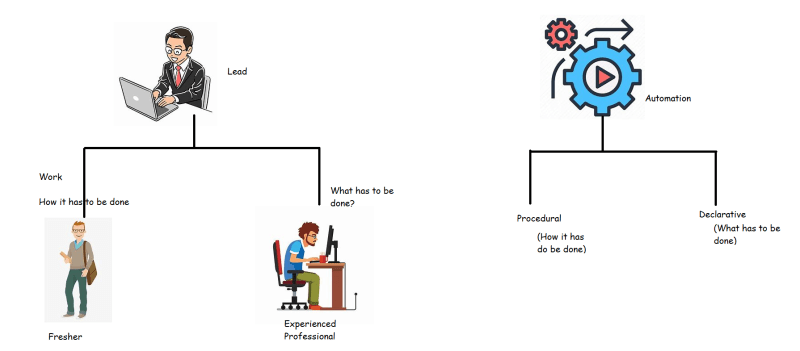
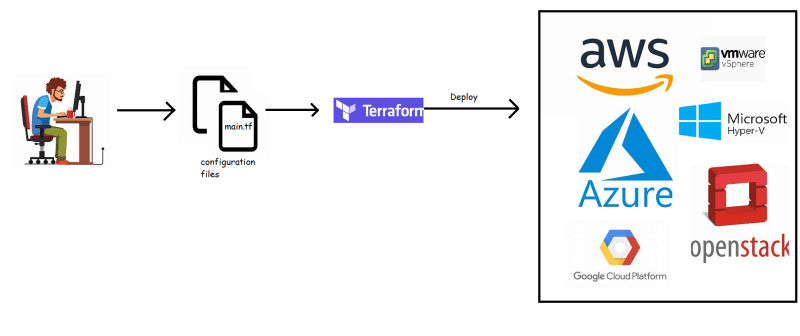
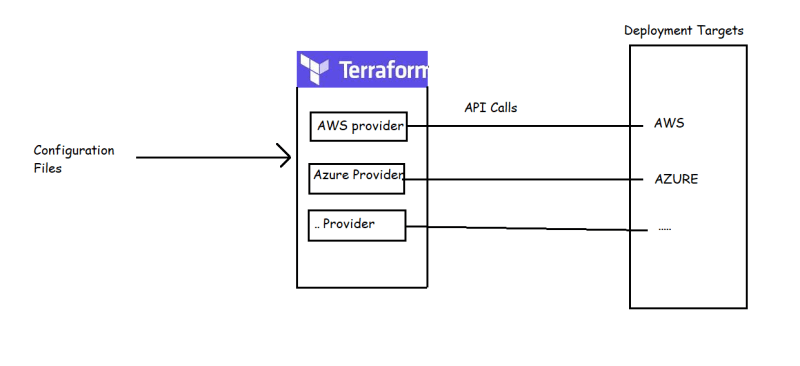
**Terraform By Khaja**

DECEMBER 14, 2021

DevOps Classroom Series – 14/Dec/2021

**Need for Infrastructure Provisioning**

* In CI/CD pipelines, we need to create various test environments according to organizational QA policy.
* To create test environments, we might also need to create virtual infrastructure on the cloud / Hypervisor (VMWare, Hyper-V).
* To automate this infrastructure creation/updation we need some kind of a tool and these tools are called as Infra Provisioning Tools
* Depending on the infrastructure used in the organization we will have different tools which can help us in automating infrastructure creation 
* To solve this problem an organization called as Hashicorp has came up with a tool called as Terraform which can work with multiple infra providers 
* Procedural vs Declarative 
* Using Terraform we would use declarative approach to specify what are our infrastructure needs => Infrastructure as Code (IaC)
* Workflow: 
* Terraform is Cloud-agnostic because deploy to any cloud or virtual infra provider 
* Terraform integrates with different clouds through Terrraform providers

**Lab Setup**

1. Azure Free Trail Account: [Refer Here](https://www.youtube.com/watch?v=MdDOc9OPVDA&list=PLuVH8Jaq3mLuqXuGs6aeqxhuvCYSzB1kT)
2. Aws Free tier account [Refer Here](https://www.youtube.com/watch?v=z95MhW1gAcA&list=PLuVH8Jaq3mLszrC7lv68a0VcrDripW-HK)
3. Windows Users: [Refer Here](https://www.youtube.com/watch?v=mRILfUNbsIo&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=14)
4. Windows 10+ Users: Install Windows Terminal [Refer Here](https://www.youtube.com/watch?v=qLVn2EvPsYc&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=11)
5. Mac:
   1. Install Homebrew: [Refer Here](https://docs.brew.sh/Installation)
   2. Git: brew install git
   3. Visual Studio Code: brew install --cask visual-studio-code [Refer Here](https://formulae.brew.sh/cask/visual-studio-code)

**Share this:**

DECEMBER 15, 2021

DevOps Classroom Series – 15/Dec/2021

**Installing Terraform**

* Terraform is an open-source project developed in Google’s GO language
* Installing Terraform is much like downloading an exe/executable on your system adding it to the PATH variable and using it
* Mac users => Install Homebrew => [Refer Here](https://docs.brew.sh/Installation)

# Launch terminal

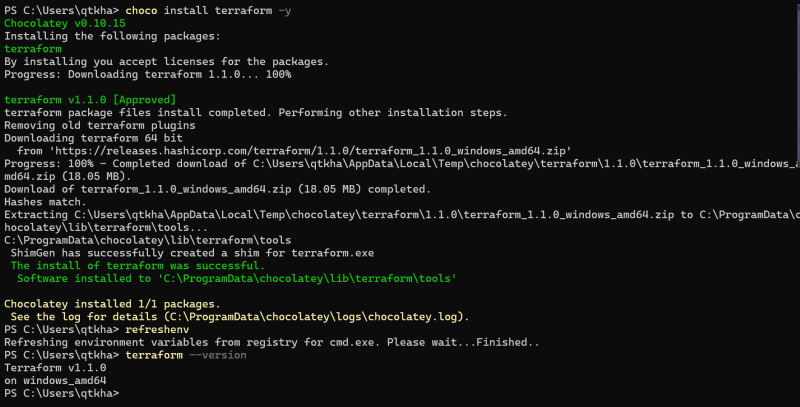
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

* Windows Users => Install Chocolatey => [Refer Here](https://chocolatey.org/install)

# Launch Powershell as admin

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))

Close your terminal/Powershell and relaunch as admin

* Install Terraform
  + Windows => choco => choco install terraform -y 
  + Mac => brew => brew install terraform [Refer Here](https://formulae.brew.sh/formula/terraform)

**Terraform Terminologies**

* Provider: This specifies terraform where to create infrastructure
* Resource: Resources are the core elements of your infrastructure.
  + The inputs which we specify about the resource are called as arguments
  + When we create resources using terraform they specify some outputs of the resource which are called as attributes
* Data Source: This helps us in querying information (about resources) from terraform configuration.

**Writing a Terraform Configuration file**

* We need to specify the provider
  + Syntax:
* provider "<name-of-provider>" {
* arg1 = "value1"
* ...
* ..
* argn = "value2"
* }
* We need to find a right resource from the provider

resource "<resource-type>" "<resource-name>" {

arg1 = "value1"

...

..

argn = "value2"

}

* Hello World Configuration

provider "aws" {

region = "us-west-2"

}

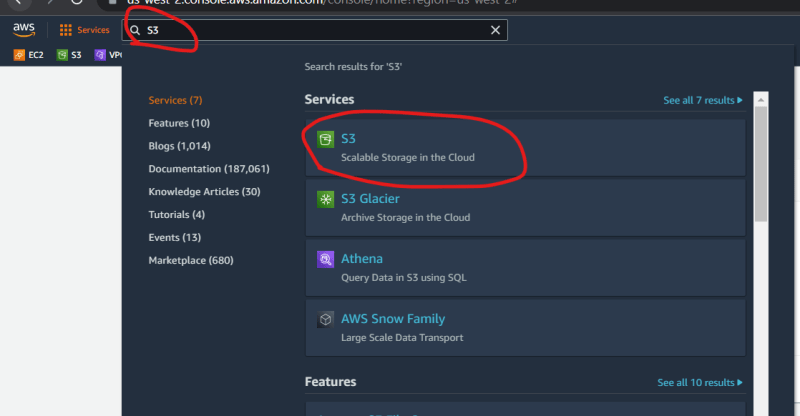
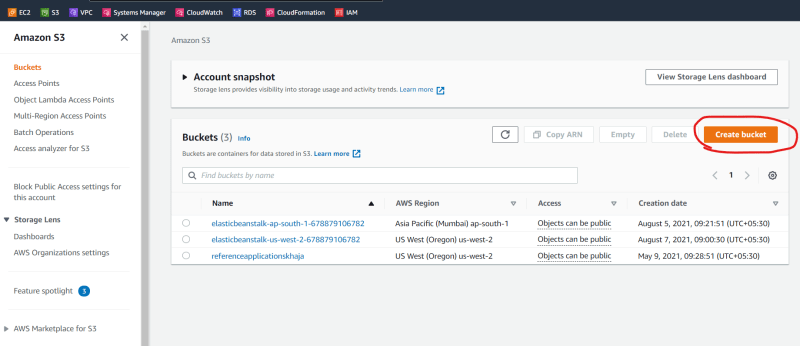
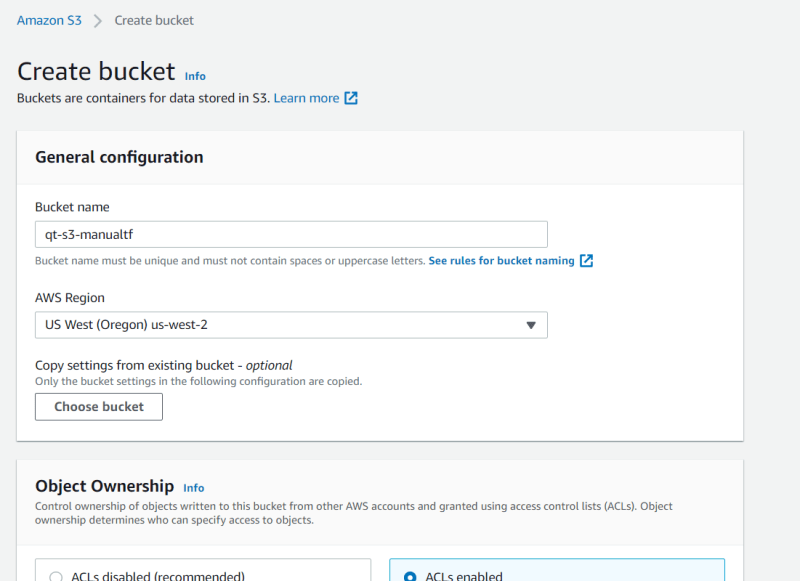
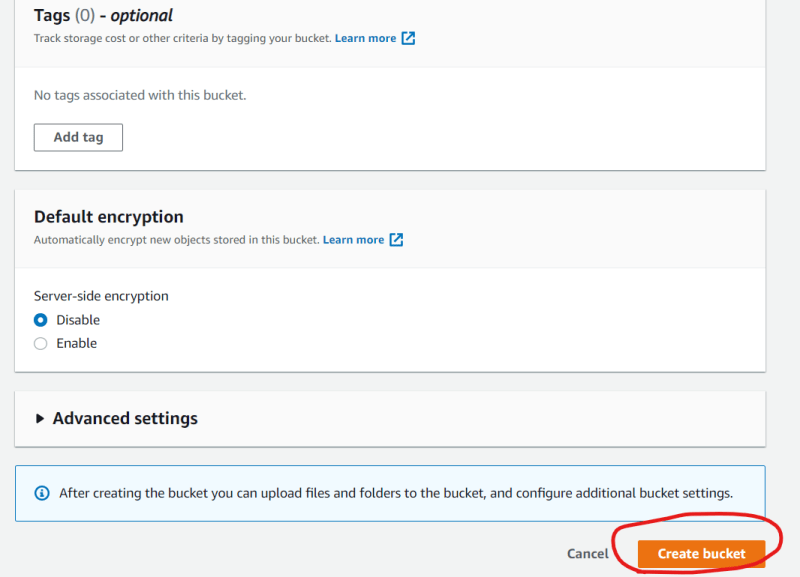
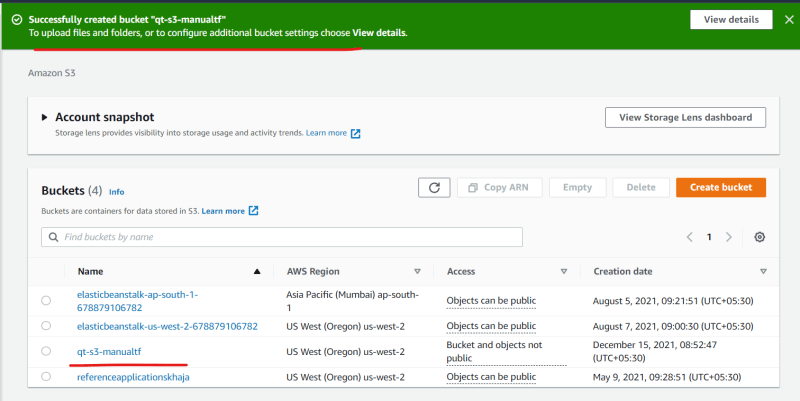
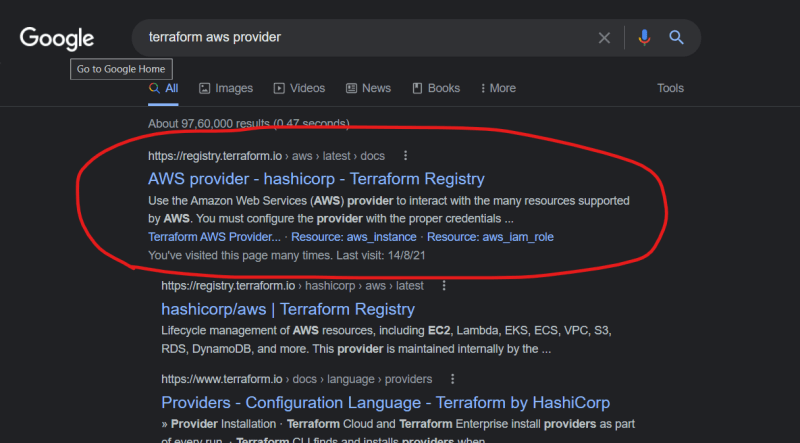
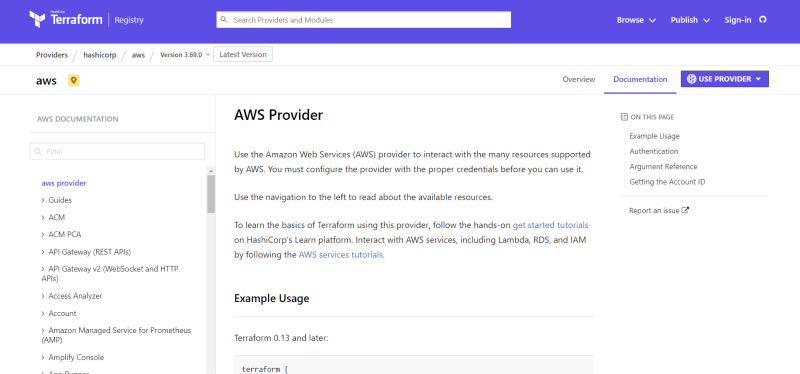
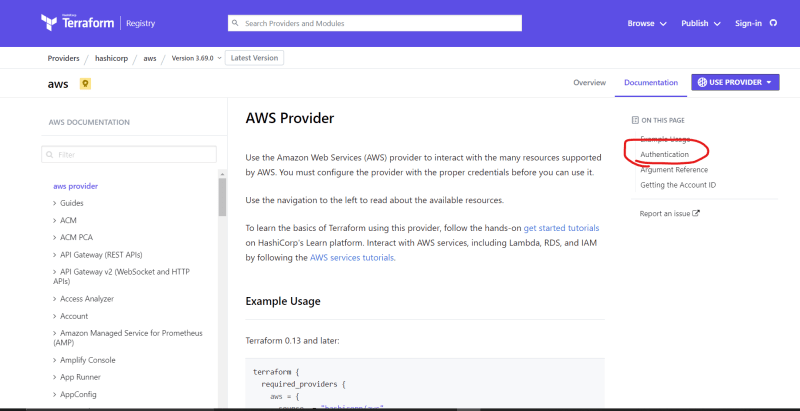
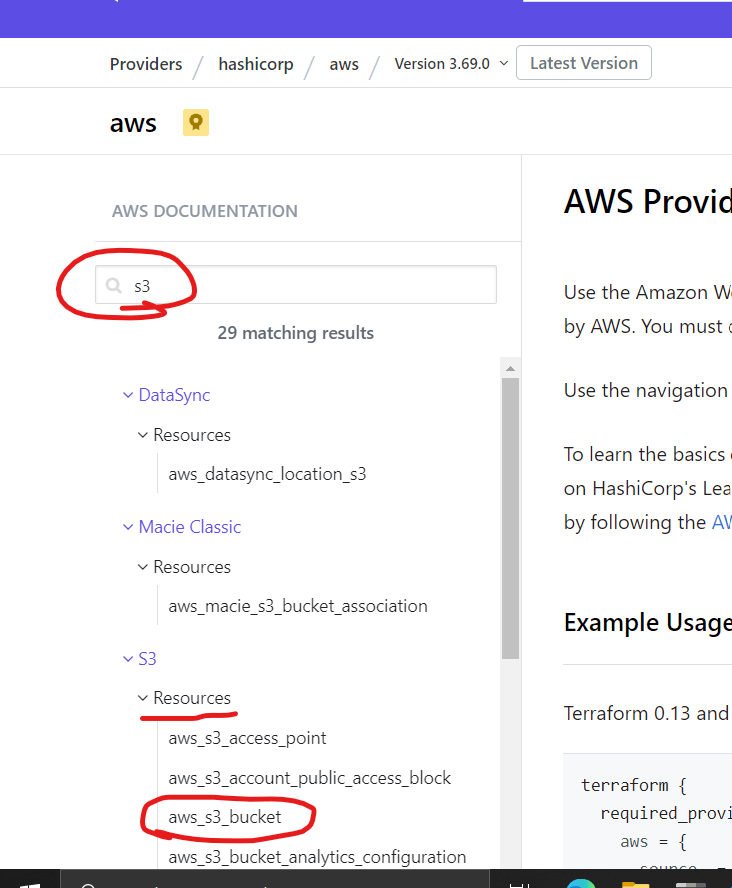
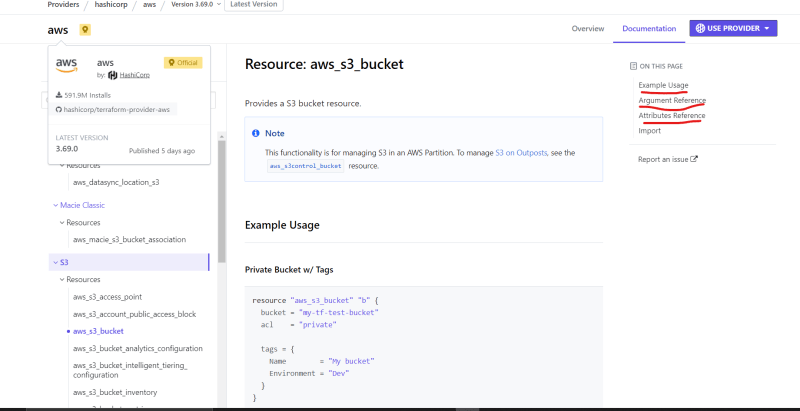
resource "aws\_s3\_bucket" "my\_bucket" {

bucket = "my-tf-qt-s3-bucket"

acl = "private"

}

**Activity: From Terraform Create an S3 bucket in AWS Cloud**

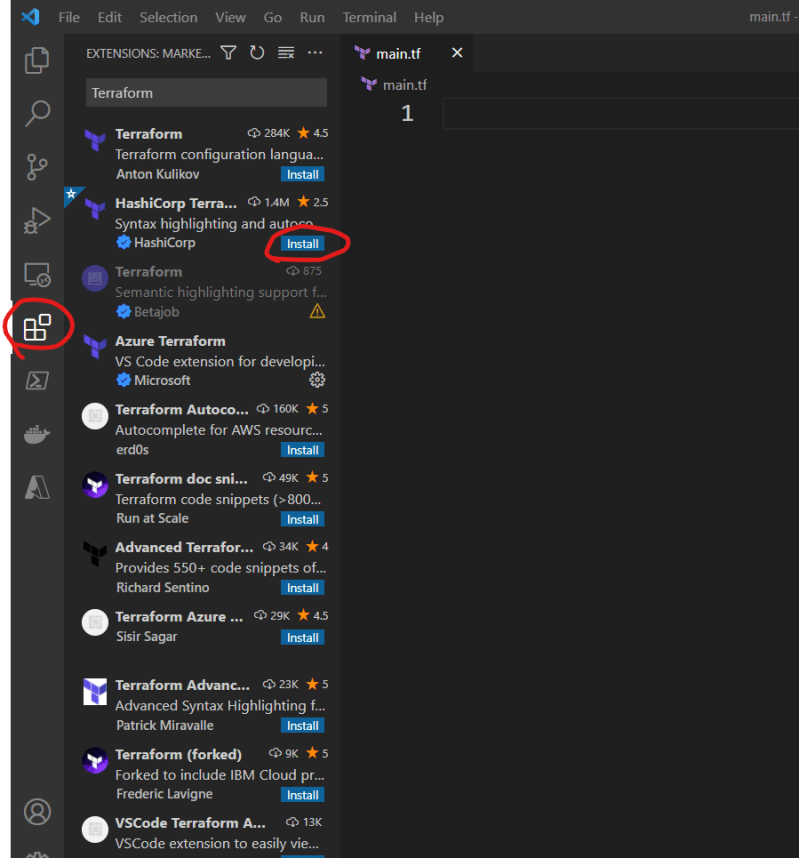
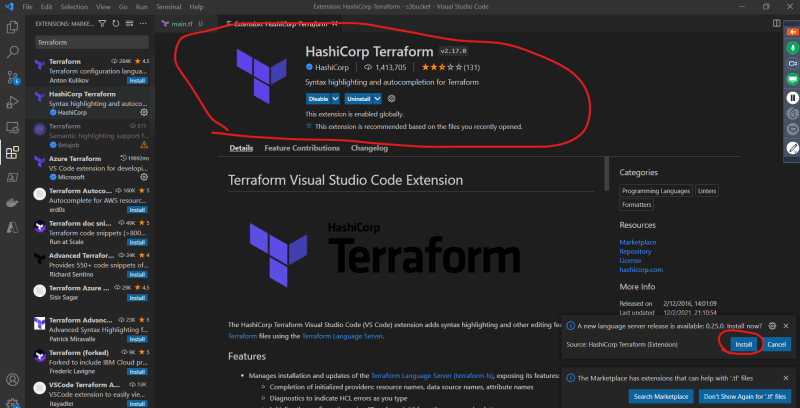
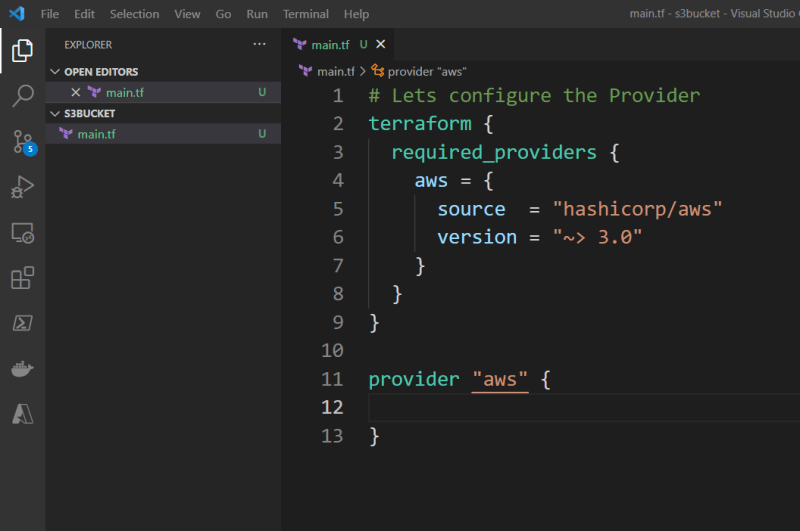
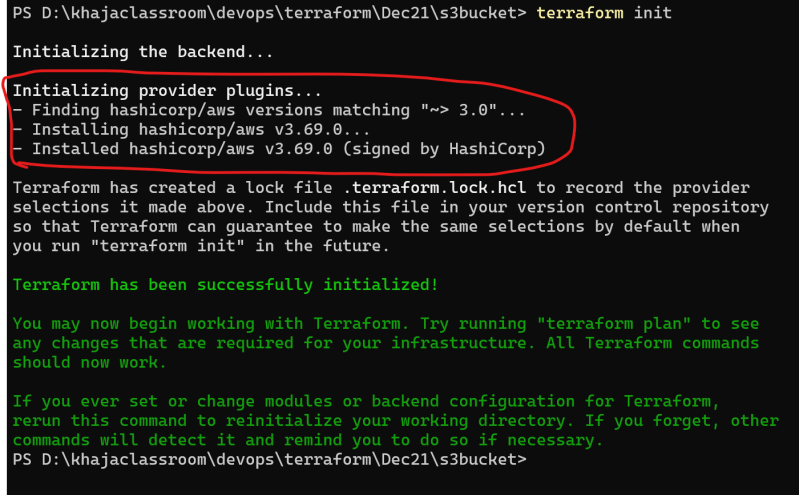
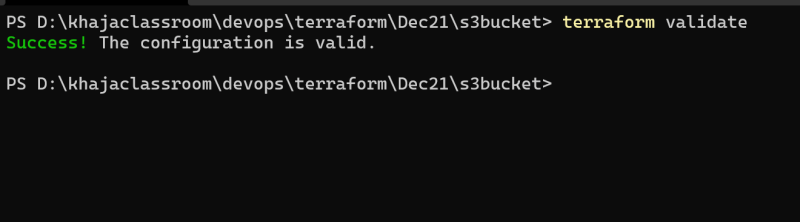
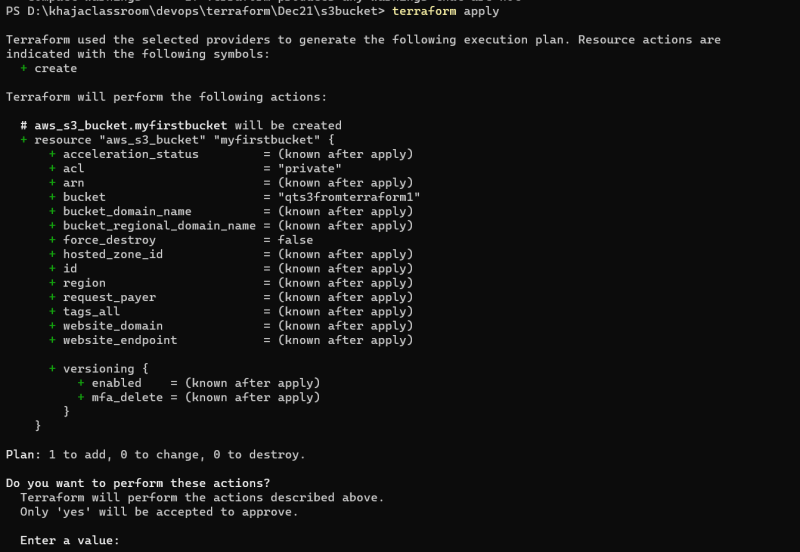
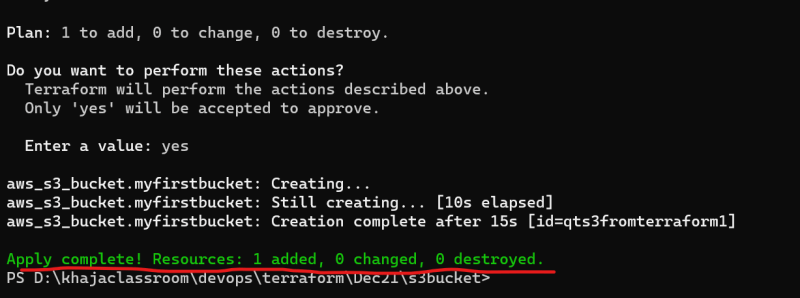
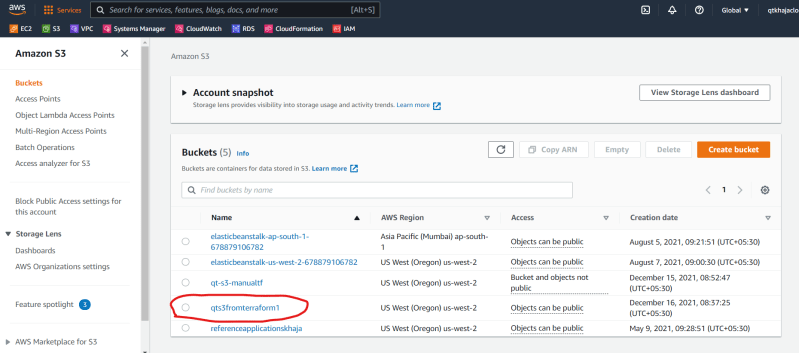
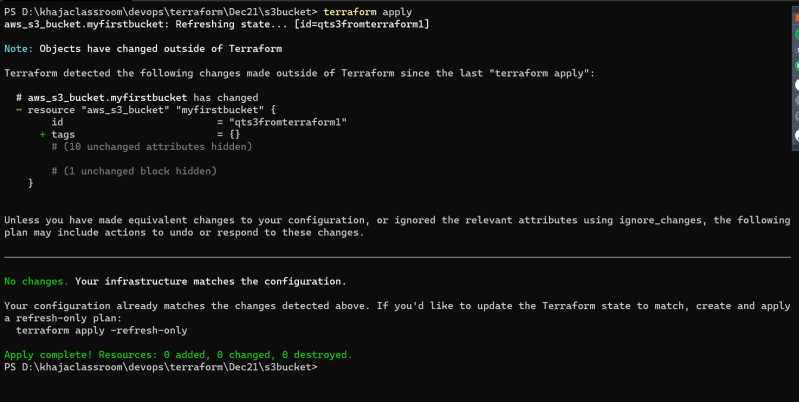
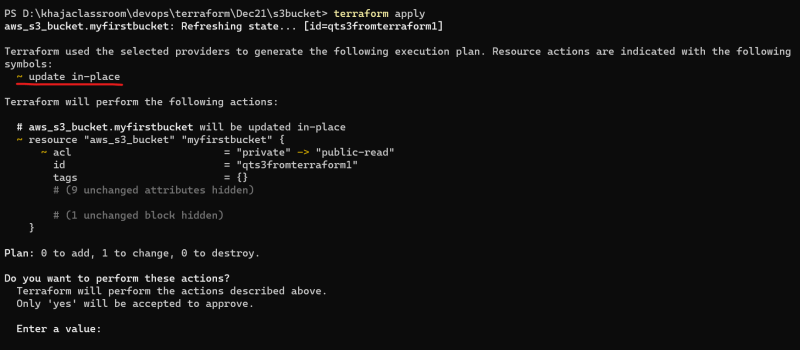
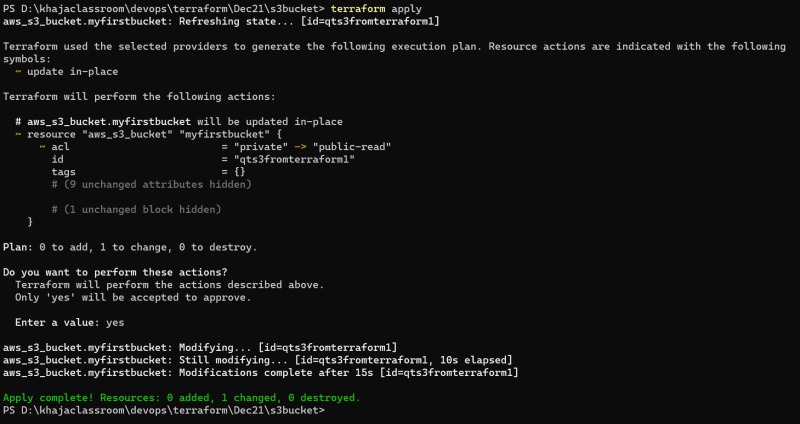
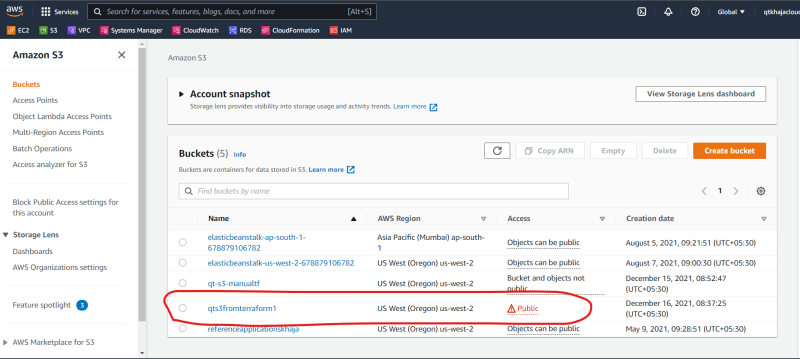
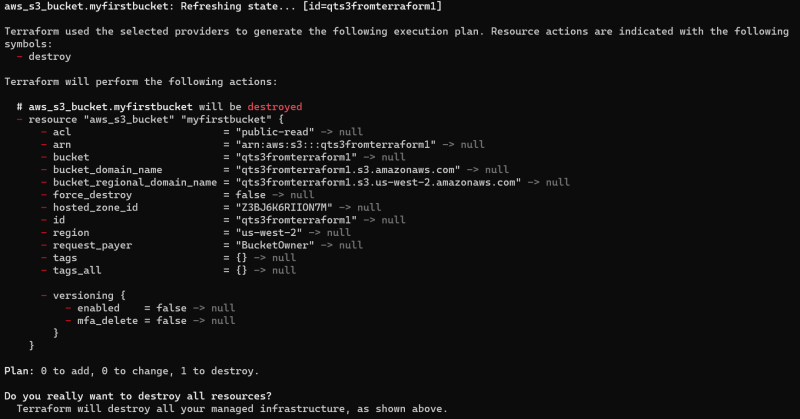
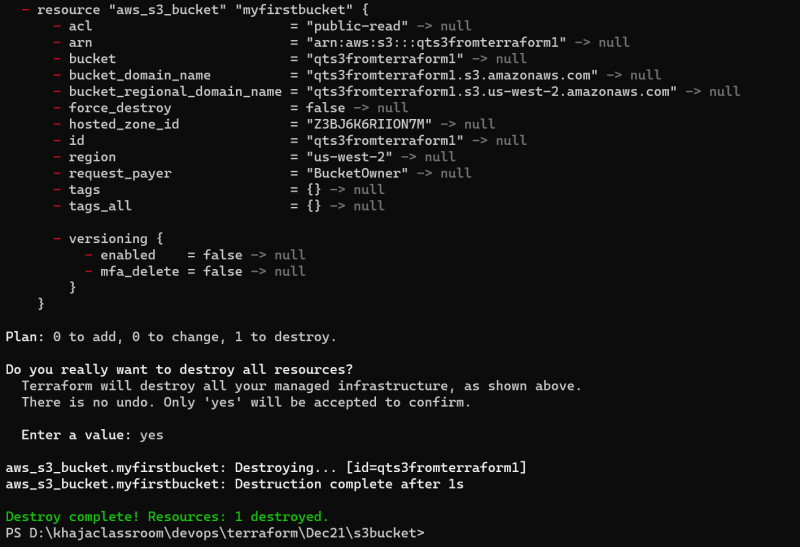
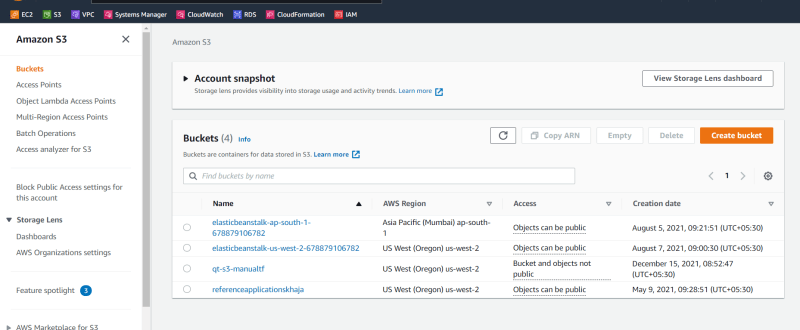
* Manual Steps:
  + Login into AWS Console
  + Follow the
  + steps     
* Approach in Terraform
  + Find the Provider:  
  + How to configure authentication 
  + Find a resource in AWS Provider to Create S3 bucket  

**Share this:**

DECEMBER 16, 2021

DevOps Classroom Series – 16/Dec/2021

**Activity: From Terraform Create an S3 bucket in AWS Cloud (Contd)**

* Create a new folder and create an empty file main.tf in it.
* Open this folder with vscode
* Install Terraform Extension from Hashicorp  
* Create the basic provider skeleton syntax 
* Install AWS CLI
  + Windows => choco install awscli
  + Mac => brew install awscli
* Create an IAM authentication as mentioned [Refer Here](https://serverless-stack.com/chapters/create-an-iam-user.html)
* Now execute aws configure and enter the secret access key id and access key, region (us-west-2) and output format => json
* So after configuring credentials. Execute terraform init to download the providers configured 
* Now we need to configure the resource s3\_bucket [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket)
* Now fill the required arguments [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket#argument-reference)
* Now let’s check for the validity of the configuration which we have written 
* Now try creating the resources by executing terraform apply  
* Now check for the resources created or not 
* Once the infra is created, lets execute terraform apply again 
* Terraform is idempotent i.e. when you run terraform multiple times to create the infrastructure it will lead to the same result i.e. your desired state
* Now lets change the acl to public-read & apply   
* Now lets try to delete the infrastructure which we have created   
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/c38455bd3291c61a43e7678803edee8fae8fdaf5) for the changes done in the main.tf file.

MAIN.TF

# Lets configure the Provider

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 3.0"

    }

  }

}

provider "aws" {

  region = "us-west-2"

}

#Resource for the s3

resource "aws\_s3\_bucket" "my\_bucket" {

    bucket = "qts3formattf"

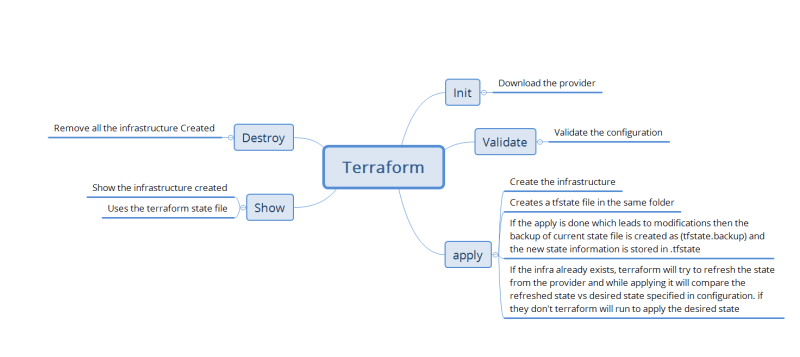
}

**Share this:**

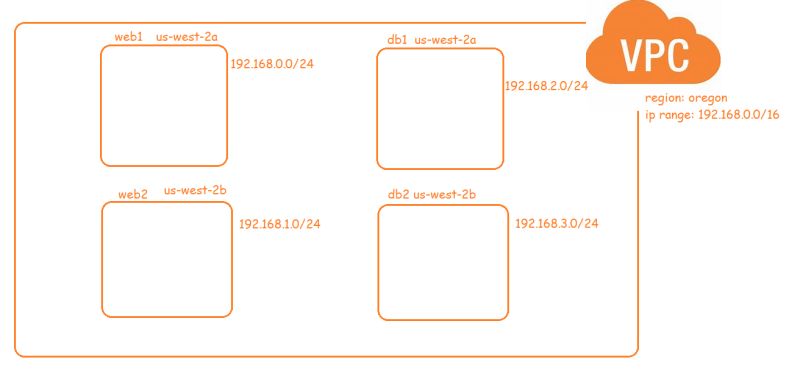
DECEMBER 17, 2021

DevOps Classroom Series – 17/Dec/2021

**Terraform Infra Creation**

* The terraform commands and its behavior 
* The terraform is storing the state in the local folder by default

**Activity: AWS infra 2**

* Phase 1: Implementation: Create a VPC as specified below 
* For manual steps watch classroom recording
* aws\_vpc: [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/vpc)
* aws\_subnet: [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/vpc)
* While creating subnet we need vpc id which we can get from the attributes of the vpc resource created. [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/vpc#attributes-reference)
* To use the attributes syntax is <resource-type>.<resource-name>.<attribute-name>
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/ac5b9433ec67a57e1a88445d4a261585f5c1aeb5) for the changes done.

[network.tf](https://github.com/asquarezone/TerraformZone/commit/ac5b9433ec67a57e1a88445d4a261585f5c1aeb5#diff-9c34ee96bede1365078bf14f60c66d1827a2e2f21332aa26fc188a0ff96971a4)

|  |
| --- |
|  |
| resource "aws\_vpc" "primary\_vpc" { |
|  |  | cidr\_block = "192.168.0.0/16" |
|  |  | tags = { |
|  |  | Name = "primary" |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  |  |
|  |  | resource "aws\_subnet" "web\_1" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2a" |
|  |  | cidr\_block = "192.168.0.0/24" |
|  |  | tags = { |
|  |  | Name = "Web1" |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | resource "aws\_subnet" "web\_2" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2b" |
|  |  | cidr\_block = "192.168.1.0/24" |
|  |  | tags = { |
|  |  | Name = "Web2" |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  |  |
|  |  | resource "aws\_subnet" "db\_1" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2a" |
|  |  | cidr\_block = "192.168.2.0/24" |
|  |  | tags = { |
|  |  | Name = "db1" |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | resource "aws\_subnet" "db\_2" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2b" |
|  |  | cidr\_block = "192.168.3.0/24" |
|  |  | tags = { |
|  |  | Name = "db2" |
|  |  | } |
|  |  | } |
|  |  |  |

[providers.tf](https://github.com/asquarezone/TerraformZone/commit/ac5b9433ec67a57e1a88445d4a261585f5c1aeb5#diff-a2c90b5383a324a48884f73b264fb9dc32fc303db5bcb34063cde684235002cf)

|  |
| --- |
| terraform { |
|  |  | required\_providers { |
|  |  | aws = { |
|  |  | source = "hashicorp/aws" |
|  |  | version = "~> 3.0" |
|  |  | } |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | # Configure the AWS Provider |
|  |  | provider "aws" { |
|  |  | region = "us-west-2" |
|  |  | } |

DECEMBER 18, 2021

DevOps Classroom Series – 18/Dec/2021

**Activity 2 AWS Infra Contd**

* We have created the resource which look as shown below

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 3.0"

}

}

}

# Configure the AWS Provider

provider "aws" {

region = "us-west-2"

}

resource "aws\_vpc" "primary\_vpc" {

cidr\_block = "192.168.0.0/16"

tags = {

Name = "primary"

}

}

resource "aws\_subnet" "web\_1" {

vpc\_id = aws\_vpc.primary\_vpc.id

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

cidr\_block = "192.168.0.0/24"

tags = {

Name = "Web1"

}

}

resource "aws\_subnet" "web\_2" {

vpc\_id = aws\_vpc.primary\_vpc.id

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

cidr\_block = "192.168.1.0/24"

tags = {

Name = "Web2"

}

}

resource "aws\_subnet" "db\_1" {

vpc\_id = aws\_vpc.primary\_vpc.id

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

cidr\_block = "192.168.2.0/24"

tags = {

Name = "db1"

}

}

resource "aws\_subnet" "db\_2" {

vpc\_id = aws\_vpc.primary\_vpc.id

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

cidr\_block = "192.168.3.0/24"

tags = {

Name = "db2"

}

}

* Lets try to parametrize the configuration, which gives the ability to the user pass information while running the template
* To parametrize we would use variables (input variables, terraform variables)
* [Refer Here](https://www.terraform.io/language/values/variables) for the docs
* To create a variable we need to define a variable block. Lets create a variable for vpc cidr

variable "vpc\_cidr" {

default = "192.168.0.0/16"

description = "This is the VPC cidr"

type = string

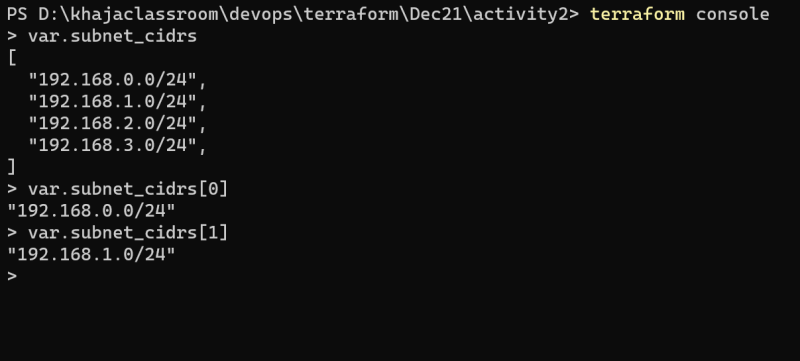
}

* To access/use this variable var.<VARIABLE\_NAME> is the syntax
* [Dec21/activity2/inputs.tf](https://github.com/asquarezone/TerraformZone/commit/c7aaa31ea07bd641ccd410fc68a6be1992b86257#diff-9083f1b2fc16bda8285fc1c1b99a275203fadac23865e9f1bf3d0e46934c7684)

|  |  |  |
| --- | --- | --- |
|  |  | @@ -0,0 +1,5 @@ |
|  |  | variable "vpc\_cidr" { |
|  |  | default = "192.168.0.0/16" |
|  |  | description = "This is the VPC cidr" |
|  |  | type = string |
|  |  | } |

* **2** [Dec21/activity2/network.tf](https://github.com/asquarezone/TerraformZone/commit/c7aaa31ea07bd641ccd410fc68a6be1992b86257#diff-9c34ee96bede1365078bf14f60c66d1827a2e2f21332aa26fc188a0ff96971a4)

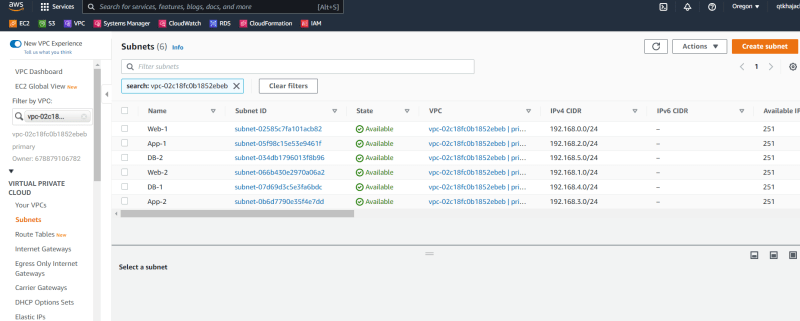
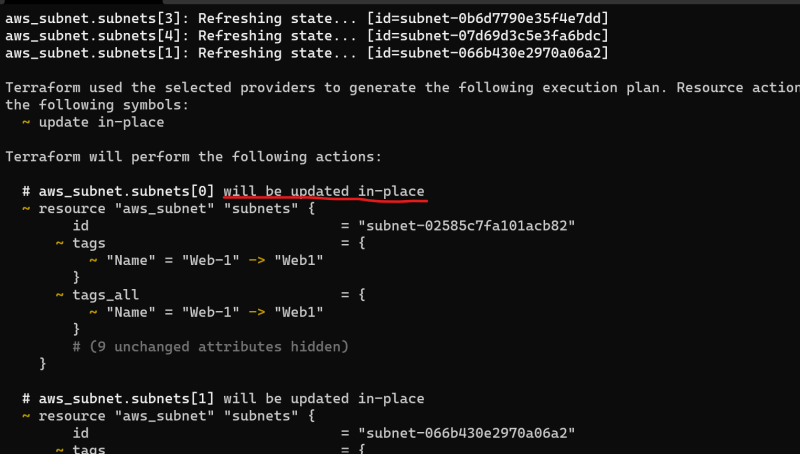
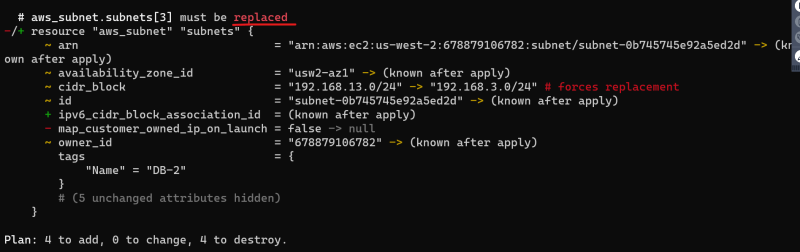
|  |  |  |
| --- | --- | --- |
|  |  | @@ -1,5 +1,5 @@ |
|  |  | resource "aws\_vpc" "primary\_vpc" { |
|  |  | cidr\_block = "192.168.0.0/16" |
|  |  | cidr\_block = var.vpc\_cidr |
|  |  | tags = { |
|  |  | Name = "primary" |
|  |  | } |

* To pass the variable value from cli terraform apply -var="vpc\_cidr=192.168.0.0/16"
* Lets create a variable for subnet cidr ranges 
* [Dec21/activity2/inputs.tf](https://github.com/asquarezone/TerraformZone/commit/274fefd9830fe83bf737c8dbedd14da3fcb5e3f8#diff-9083f1b2fc16bda8285fc1c1b99a275203fadac23865e9f1bf3d0e46934c7684)

|  |  |  |
| --- | --- | --- |
|  | | @@ -2,4 +2,21 @@ variable "vpc\_cidr" { |
|  |  | default = "192.168.0.0/16" |
|  |  | description = "This is the VPC cidr" |
|  |  | type = string |
|  |  | } |
|  |  |  |
|  |  |  |
|  |  | variable "subnet\_cidrs" { |
|  |  | default = ["192.168.0.0/24","192.168.1.0/24","192.168.2.0/24","192.168.3.0/24"] |
|  |  | description = "These are subnet cidr ranges" |
|  |  | } |
|  |  |  |
|  |  | variable "subnet\_azs" { |
|  |  | default = ["us-west-2a", "us-west-2b", "us-west-2a", "us-west-2b"] |
|  |  | description = "Availability Zones for the subnets" |
|  |  | } |
|  |  |  |
|  |  | variable "subnet\_names" { |
|  |  | default = ["Web-1", "Web-2", "DB-1", "DB-2"] |
|  |  | description = "Names of subnets" |
|  |  |  |
|  |  | } |

* **24** [Dec21/activity2/network.tf](https://github.com/asquarezone/TerraformZone/commit/274fefd9830fe83bf737c8dbedd14da3fcb5e3f8#diff-9c34ee96bede1365078bf14f60c66d1827a2e2f21332aa26fc188a0ff96971a4)

|  |  |  |
| --- | --- | --- |
|  | | @@ -8,38 +8,38 @@ resource "aws\_vpc" "primary\_vpc" { |
|  |  |  |
|  |  | resource "aws\_subnet" "web\_1" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2a" |
|  |  | cidr\_block = "192.168.0.0/24" |
|  |  | availability\_zone = var.subnet\_azs[0] |
|  |  | cidr\_block = var.subnet\_cidrs[0] |
|  |  | tags = { |
|  |  | Name = "Web1" |
|  |  | Name = var.subnet\_names[0] |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | resource "aws\_subnet" "web\_2" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2b" |
|  |  | cidr\_block = "192.168.1.0/24" |
|  |  | availability\_zone = var.subnet\_azs[1] |
|  |  | cidr\_block = var.subnet\_cidrs[1] |
|  |  | tags = { |
|  |  | Name = "Web2" |
|  |  | Name = var.subnet\_names[1] |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  |  |
|  |  | resource "aws\_subnet" "db\_1" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2a" |
|  |  | cidr\_block = "192.168.2.0/24" |
|  |  | availability\_zone = var.subnet\_azs[2] |
|  |  | cidr\_block = var.subnet\_cidrs[2] |
|  |  | tags = { |
|  |  | Name = "db1" |
|  |  | Name = var.subnet\_names[2] |
|  |  | } |
|  |  | } |
|  |  |  |
|  |  | resource "aws\_subnet" "db\_2" { |
|  |  | vpc\_id = aws\_vpc.primary\_vpc.id |
|  |  | availability\_zone = "us-west-2b" |
|  |  | cidr\_block = "192.168.3.0/24" |
|  |  | availability\_zone = var.subnet\_azs[3] |
|  |  | cidr\_block = var.subnet\_cidrs[3] |
|  |  | tags = { |
|  |  | Name = "db2" |
|  |  | Name = var.subnet\_names[3] |
|  |  | } |
|  |  | } |
|  |  |  |

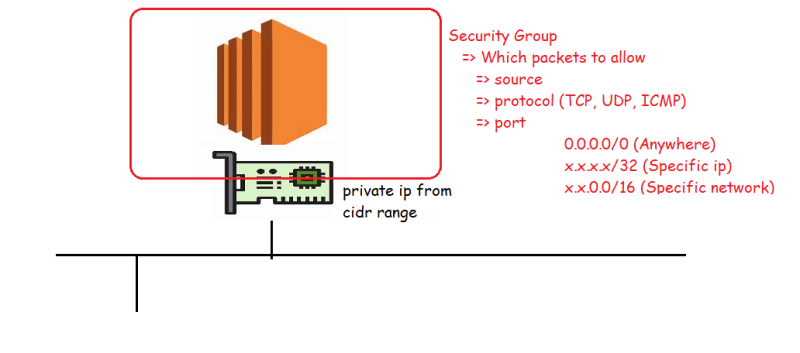
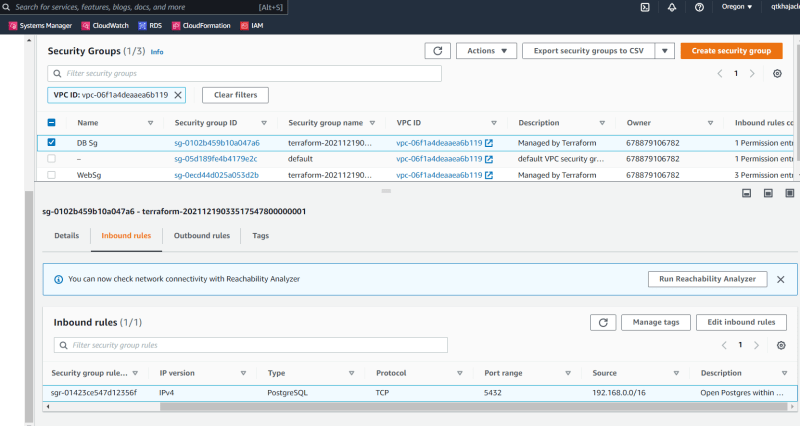
* Now Lets try to create subnet resources by using the count-meta argument [Refer Here](https://www.terraform.io/language/meta-arguments/count) 
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/9d3aa52e8d7b6792b8cfecef4fb74e0d753e8394) for the change set with changes to use the count.
* Impact of changes in the terraform configuration on already created infra:
  + Updates in place 
  + Must be replaced => Existing resource will be deleted and new one will be created. 

**Share this:**

DECEMBER 19, 2021

DevOps Classroom Series – 19/Dec/2021

**Activity 2 AWS Infra Contd**

* Now we need to create a ec2 instance (virtual machine), So lets try to understand how to do it manually
* [Refer Here](https://www.youtube.com/watch?v=me2s3mTNwGo&list=PLuVH8Jaq3mLszrC7lv68a0VcrDripW-HK&index=2) to understand create the ec2 instance
* While creating ec2 instance, we provide following inputs
  + ami-id
  + network settings:
    - vpc-id
    - subnet-id
    - public ip => Enabled/Disabled
  + Tags
  + Security Group 
  + Key-Pair
* Lets try to create a security group for web servers
  + Allow all traffic on SSH (TCP=>22), HTTP(TCP=>80), HTTPS(TCP=>443)
  + [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group) for the documentation.
  + [Refer Here](https://github.com/asquarezone/TerraformZone/commit/64a9bd47bd69732a8cb3de0662fc1123fe7c5f1b) for the changes to be done
  + After applying the changes view the security group in AWS [Refer Here](https://directdevops.blog/2021/12/19/devops-classroom-series-19-dec-2021/terraform39.png)
* Lets also use locals to represent the values which we are using in terraform configuration but we dont expect these values as user inputs [Refer Here](https://www.terraform.io/language/values/locals)
* Lets create a security group for database => Postgres (5432)
  + Lets open Postgres only to ec2 instance in our vpc
  + [Refer Here](https://github.com/asquarezone/TerraformZone/commit/ae50a4df0753bc1ac8d21e521d89a63d3fa1a702) for the changes done.
  + The result 

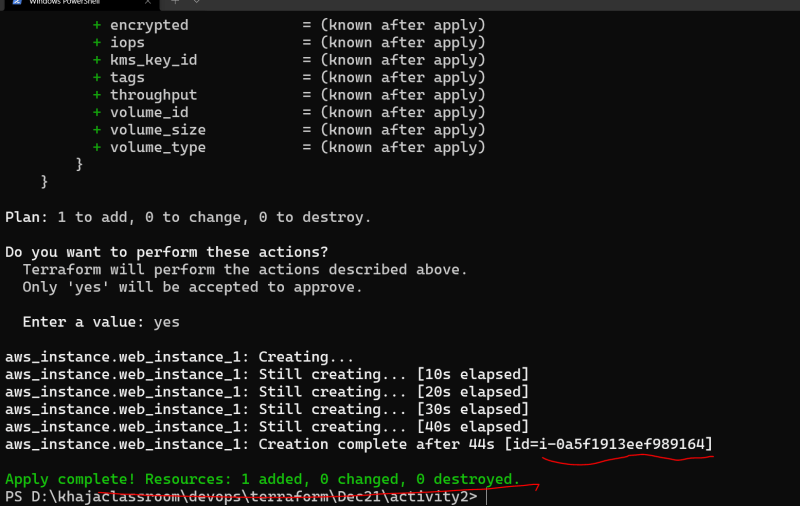
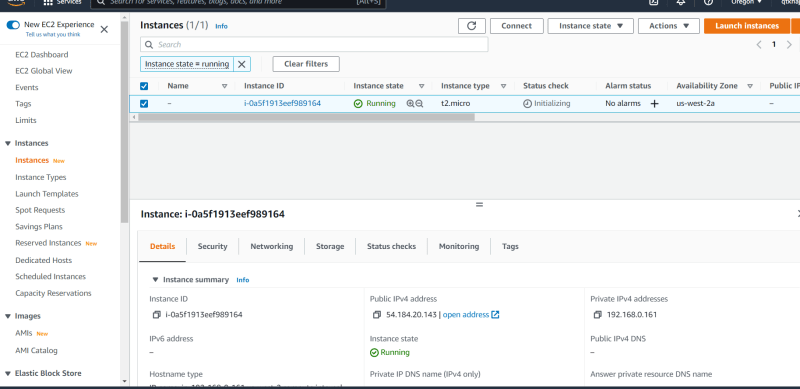
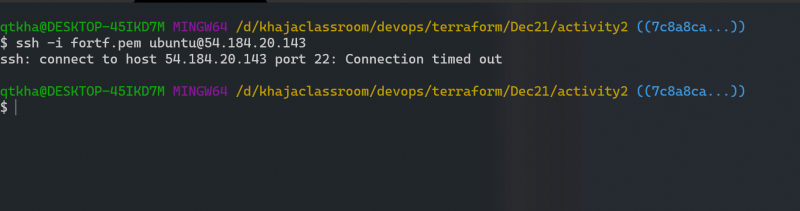
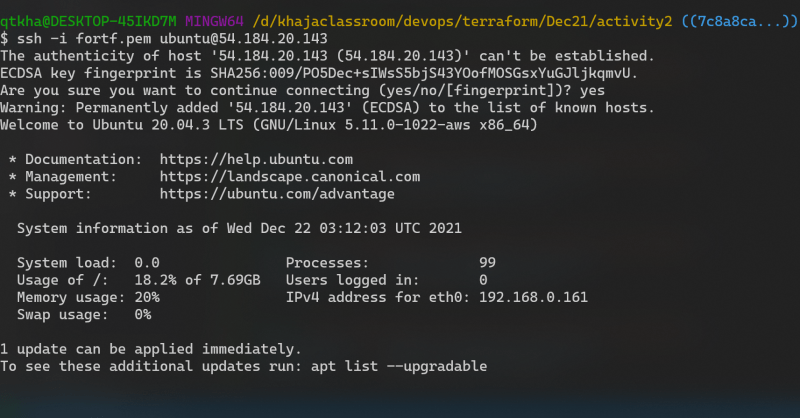
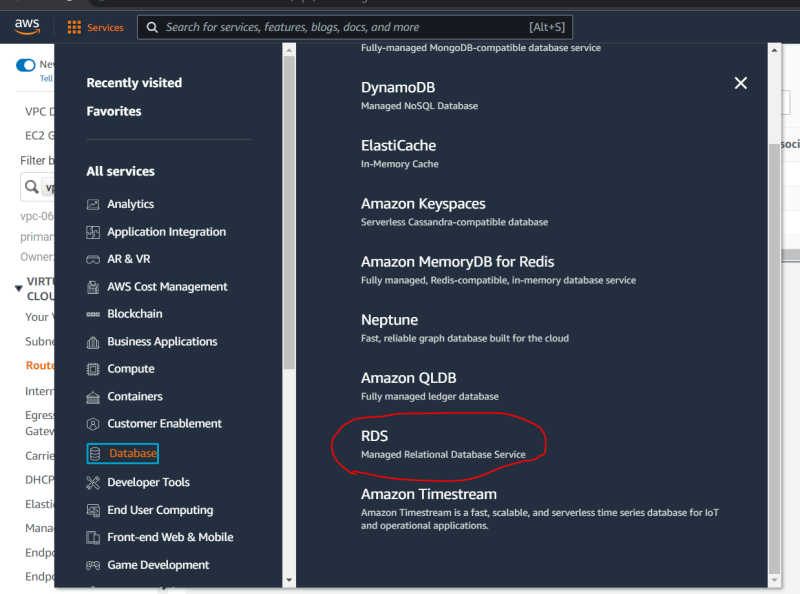
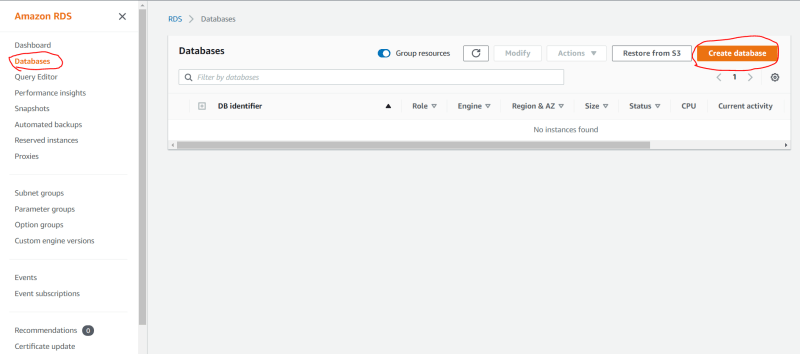
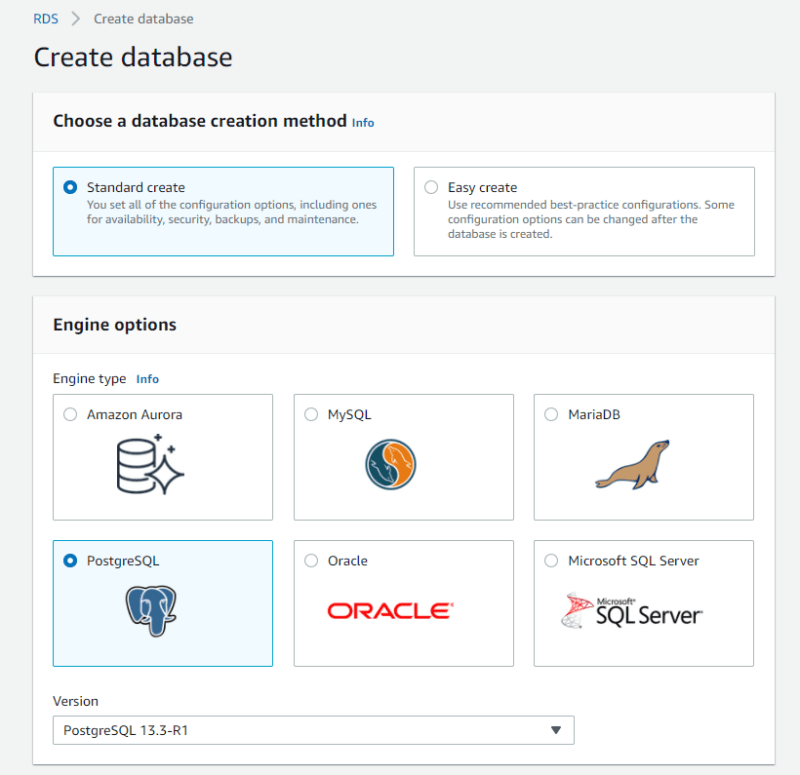
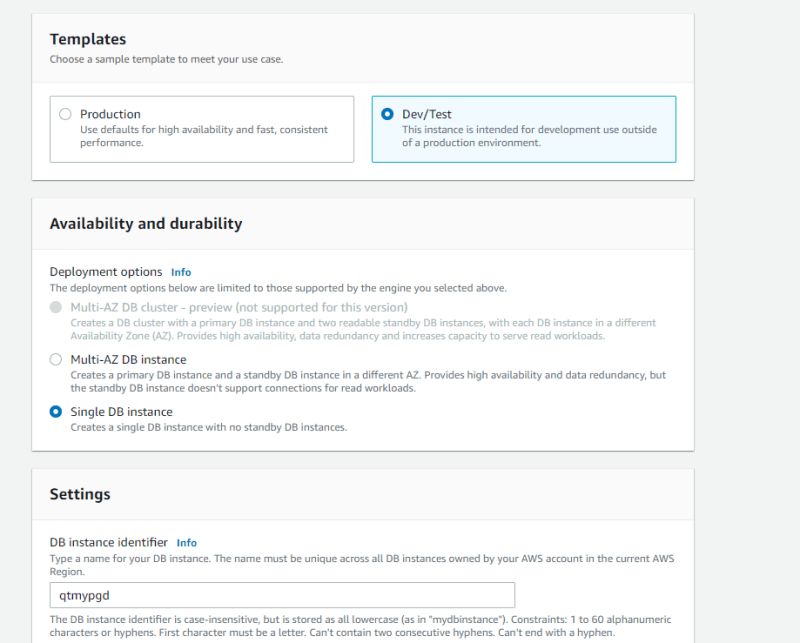
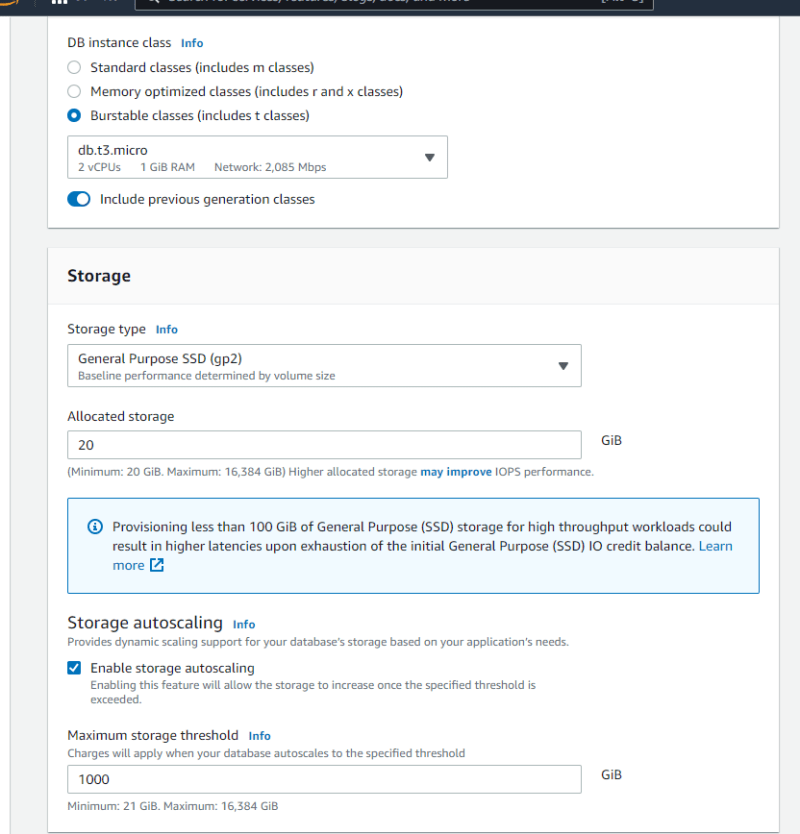
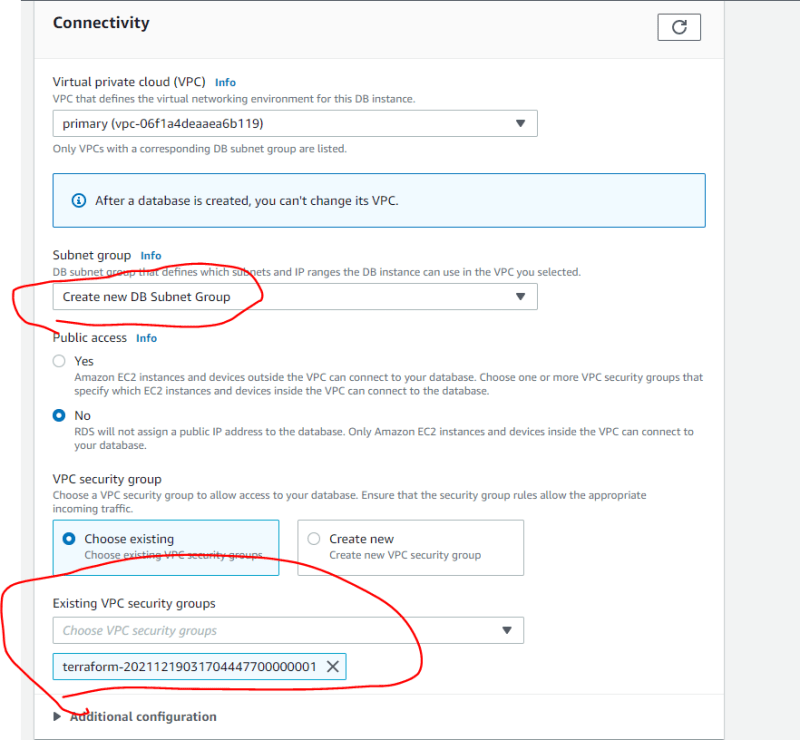
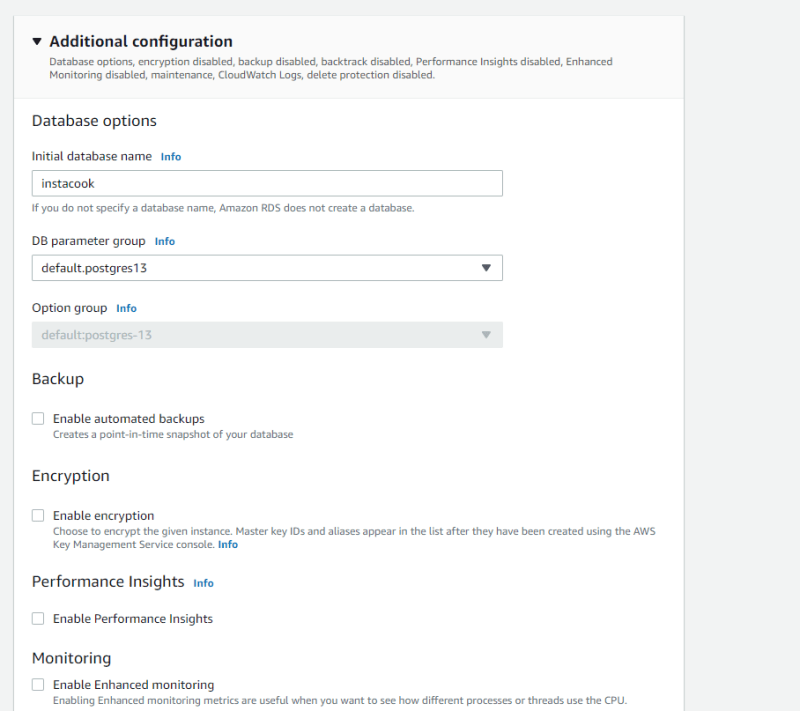
**Share this:**

* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2021/12/19/devops-classroom-series-19-dec-2021/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2021/12/19/devops-classroom-series-19-dec-2021/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2021/12/19/devops-classroom-series-19-dec-2021/?share=jetpack-whatsapp&nb=1)

DECEMBER 22, 2021

DevOps Classroomnotes 22/Dec/2021

**Activity 2 AWS infra continued**

* Lets create an ec2 instance with ami-id ami-0892d3c7ee96c0bf7
* We have already created a security group and we would like to use the existing key pair fortf
* [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance) for the terraform ec2 resource
* Created the ec2 instance. [Refer Here](https://github.com/asquarezone/TerraformZone/commit/7c8a8ca5067ba1c3a4a17356000497981619f468) for the terraform changes.  
    
  
* Now lets try to login into the ec2 instance create  
  
* Now for public connectivity, lets create an internet gateway and attach it to the vpc
* [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/internet_gateway) for the internet gateway resource
* To create route tables [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table)
* Now we need to associate route tables to subnets [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/route_table_association)
* After making the changes as mentioned [Refer Here](https://github.com/asquarezone/TerraformZone/commit/f164fc9c3e50501ce80f47a48eb308e8872503e9), We are able to login into ec2 instance created  
  
* We would need to deploy an application inside this ec2 instance. Which we will be doing very soon.
* We would like to create an database instance in our vpc hosting some database.
* In this exercise, lets try to create an Postgres Database in free tier.
* Manual Steps:  
    
    
    
    
    
    
  
* From the images we can make out that the following are the major inputs
* Engine => postgres/mysql/oracle/sql server
* version => which version of database
* Dbsubnet group =>
* network => vpc, subnet & security group
* instance size => db.t2.micro
* Credentials
* other settings
  + for backups
  + monitoring
  + storage size

**Share this:**

DECEMBER 23, 2021

DevOps Classroomnotes 23/Dec/2021

**Activity 2 AWS infra continued**

* Adding Database:
* Creating Db subnet group from two db subnets [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/db_subnet_group)
* Now lets create a database from the resource [Refer Here](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/db_instance).
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/4dd41cdb64efd29d4b8c90c80c112c1d455702ad) for the changes done
* Generally it is a good idea to show some outputs to the users whenever they execute terraform apply
* For writing outputs [Refer Here](https://www.terraform.io/language/values/outputs) for official documentation
* Lets add two simple outputs and see the terraform output while apply

output "web1\_publicip" {

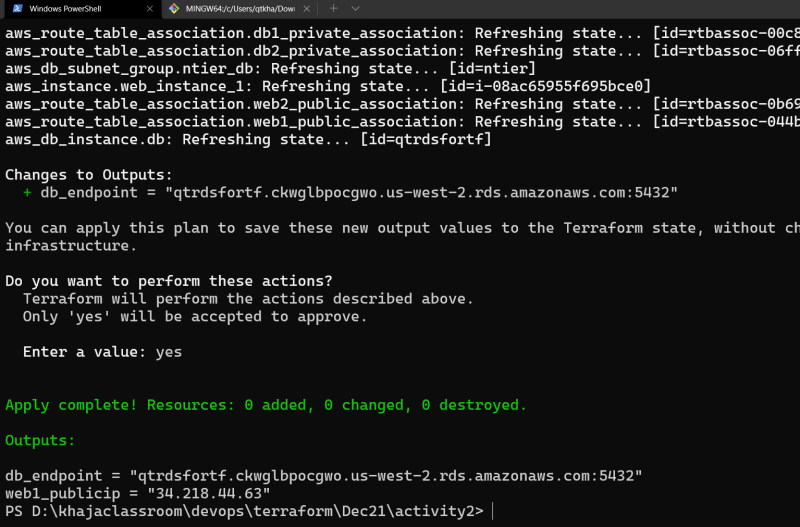
value = aws\_instance.web\_instance\_1.public\_ip

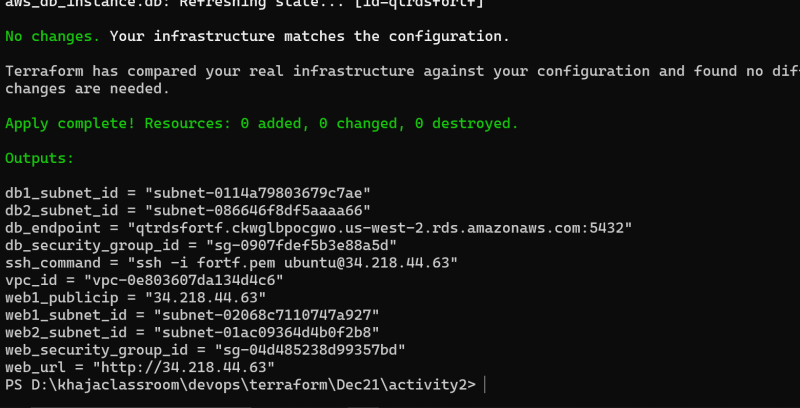
}

output "db\_endpoint" {

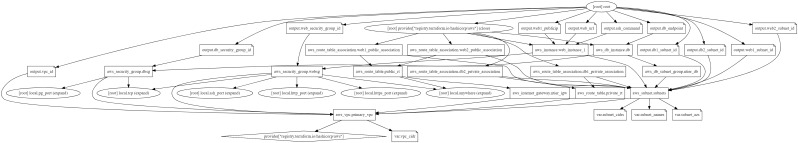
value = aws\_db\_instance.db.endpoint

}

  
\* Lets create more outputs  
\* vpc id  
\* subnet id  
\* security group ids  
\* ec2 ip address  
\* db endpoint  
\* ec2 url => http://<publicip&gt;

* Terraform has inbuilt functions which will help us [Refer Here](https://www.terraform.io/language/functions)
* Now applying format function and using the outputs section [Refer Here](https://github.com/asquarezone/TerraformZone/commit/7d8d853c27ee39652e287aa8933f6a489458d4f3) for the changes.  
  

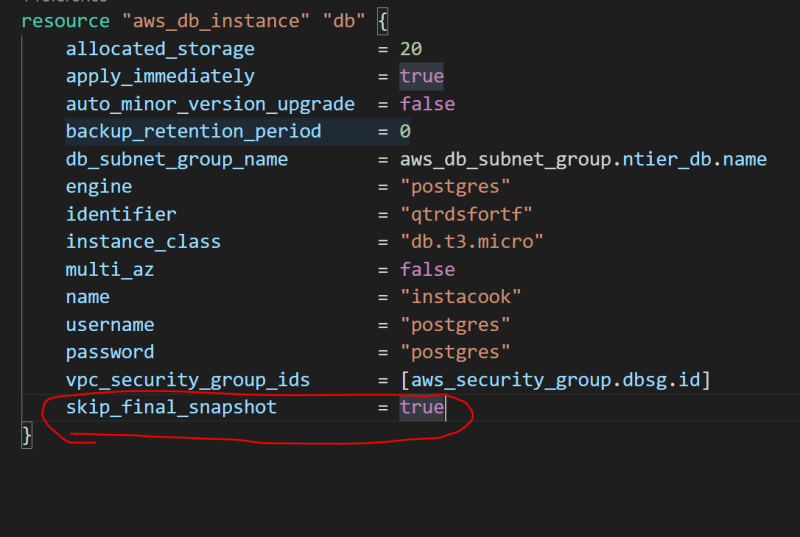
**Dependencies in Terraform**

* We can generate a graph from terraform command which will generate dot format, which can be used to create a graph  
  
* Terraform has two kinds of dependencies
* implicit
  + Created by referring attributes of one resource in another
* explicit
  + Using depends\_on

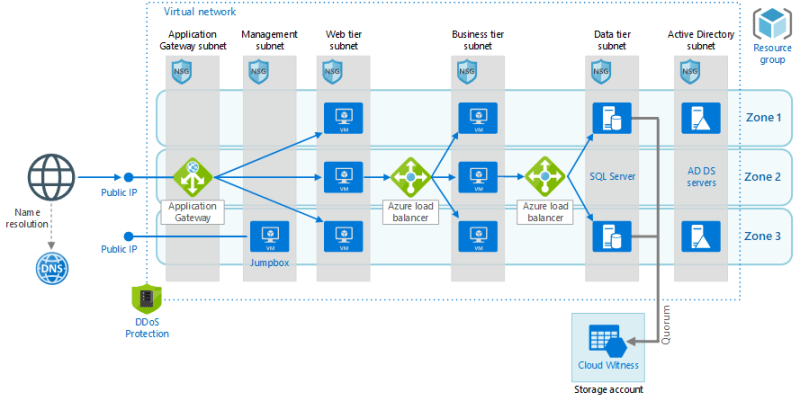
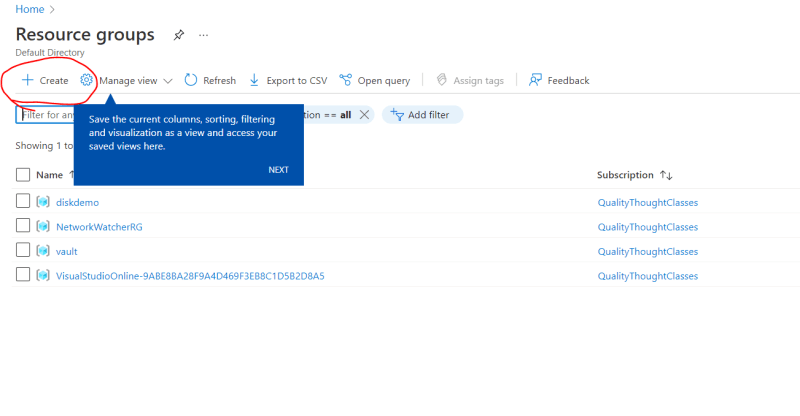
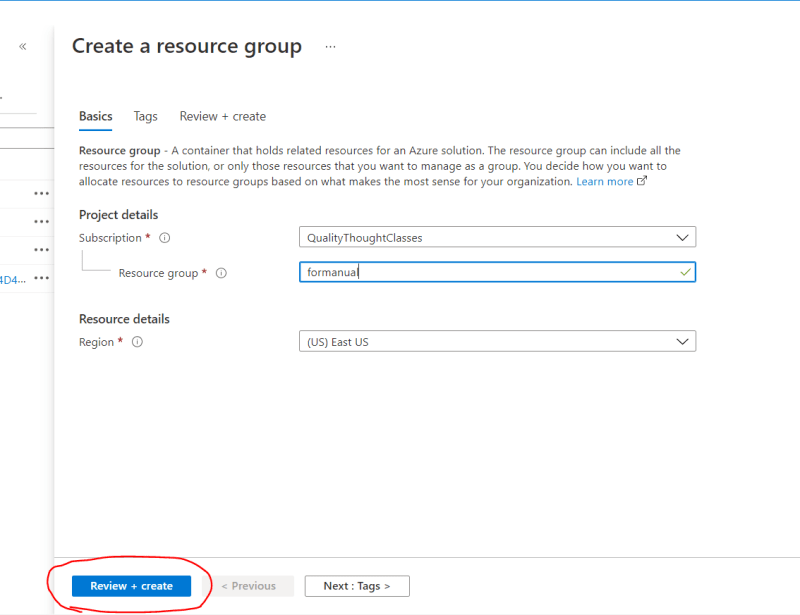
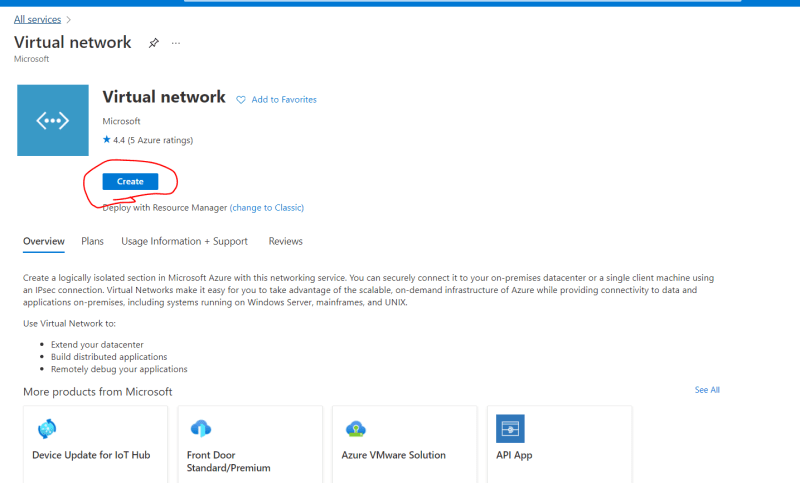
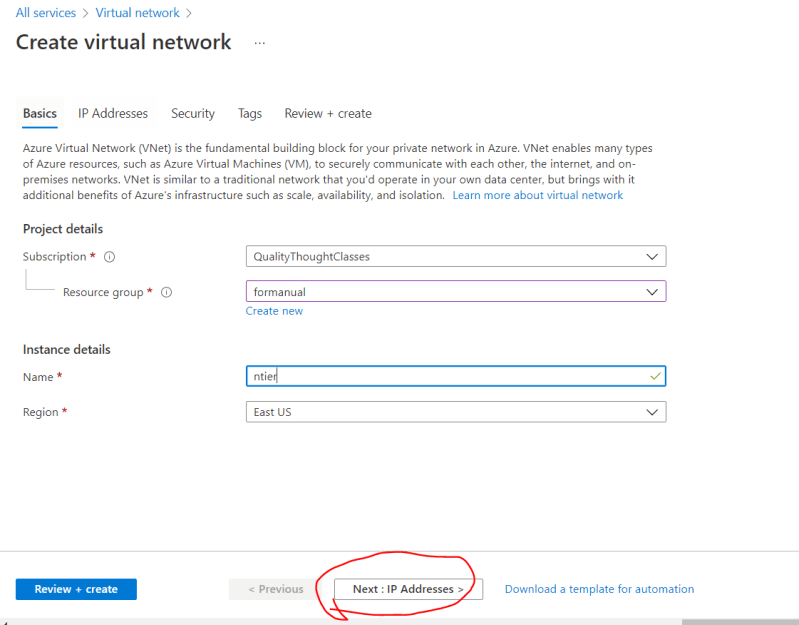
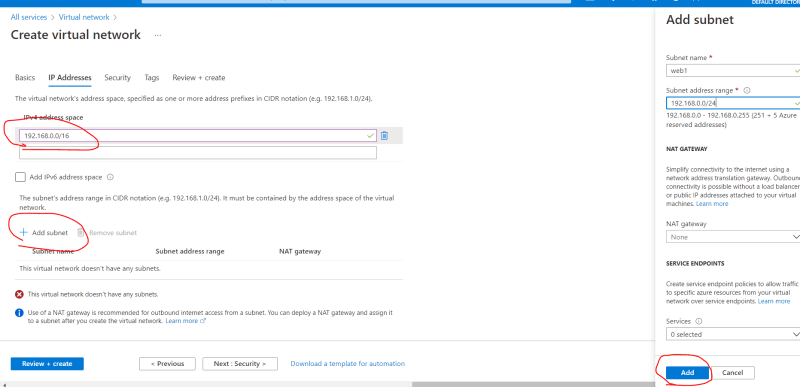
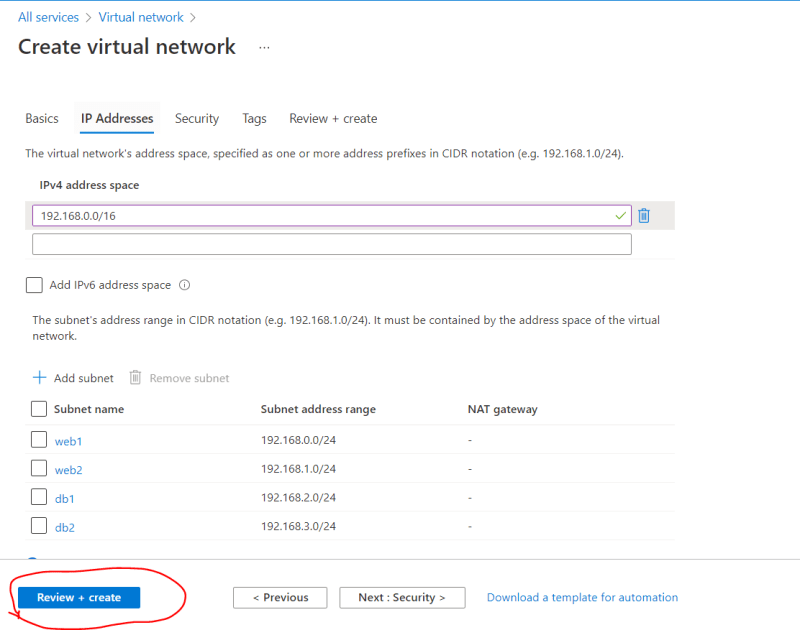
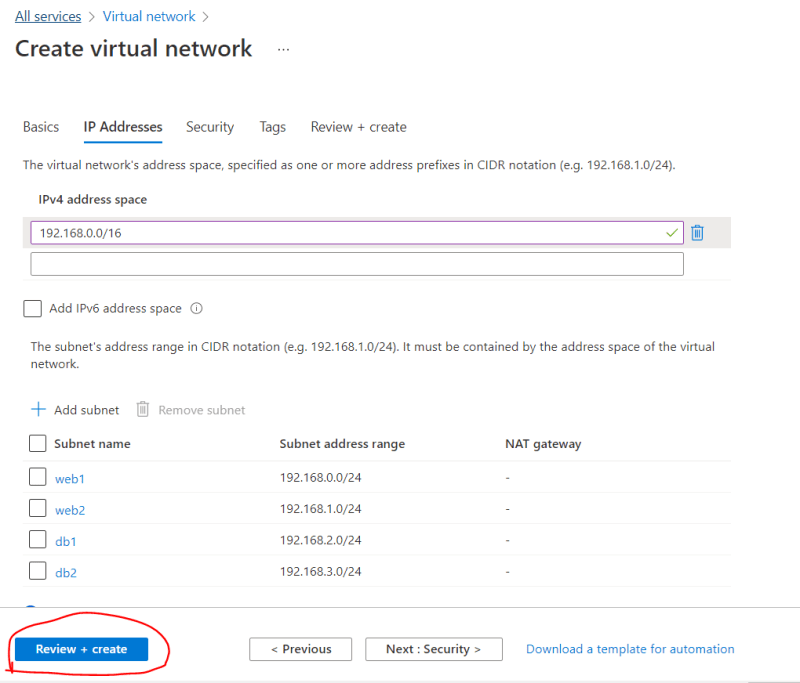
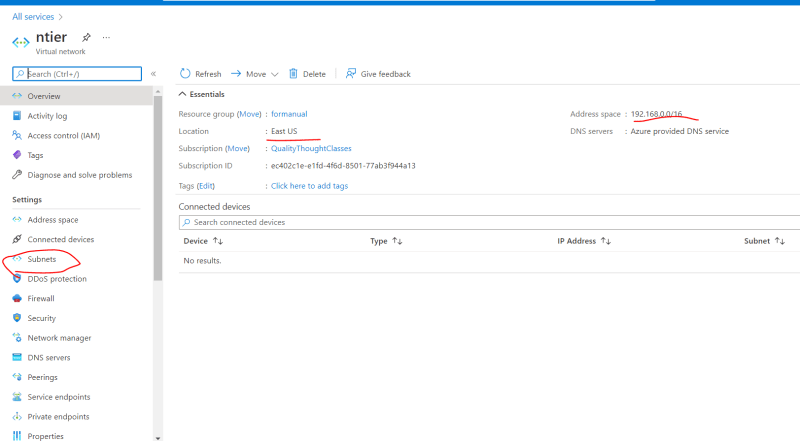
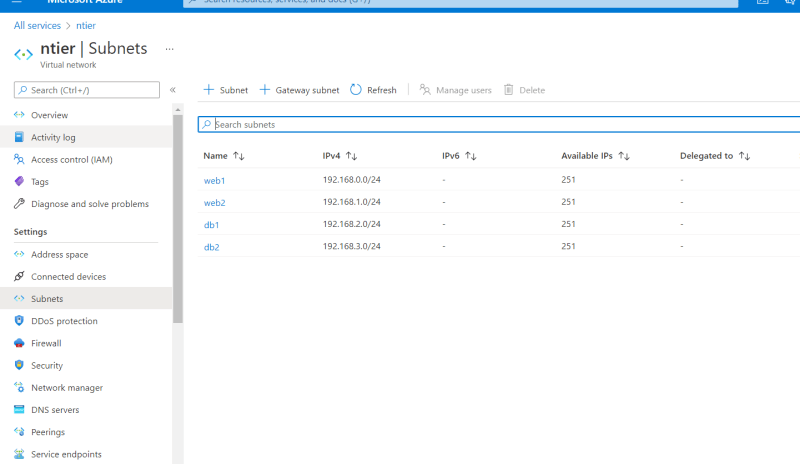
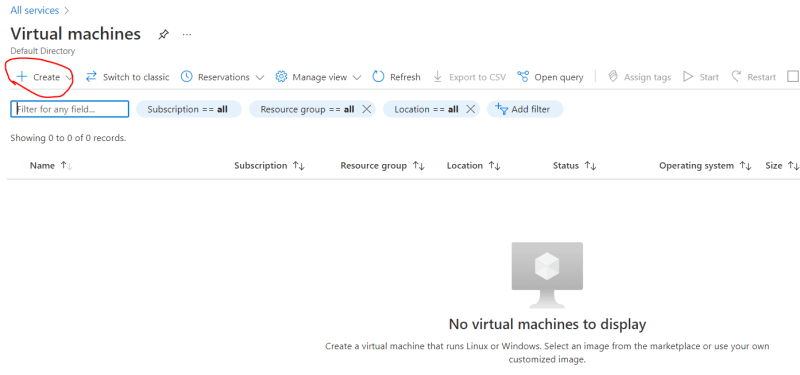
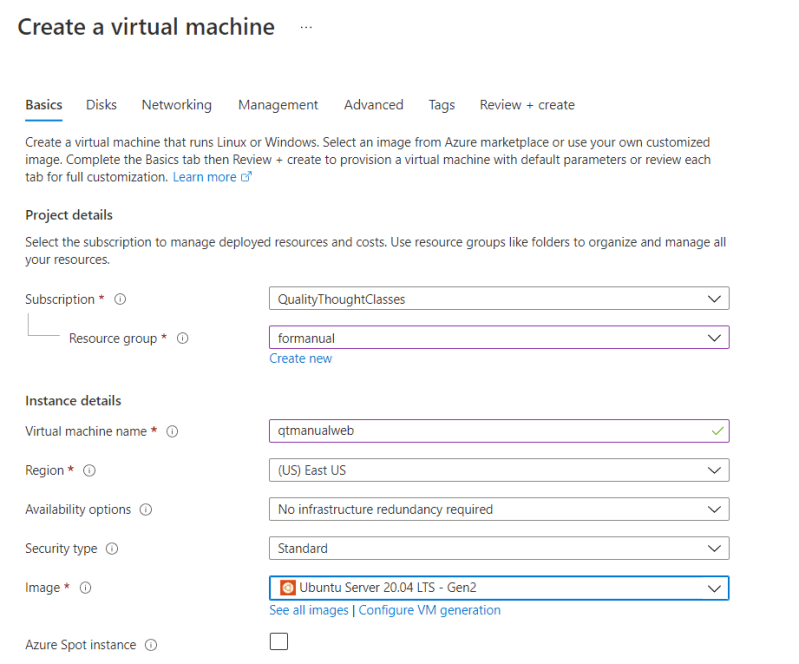
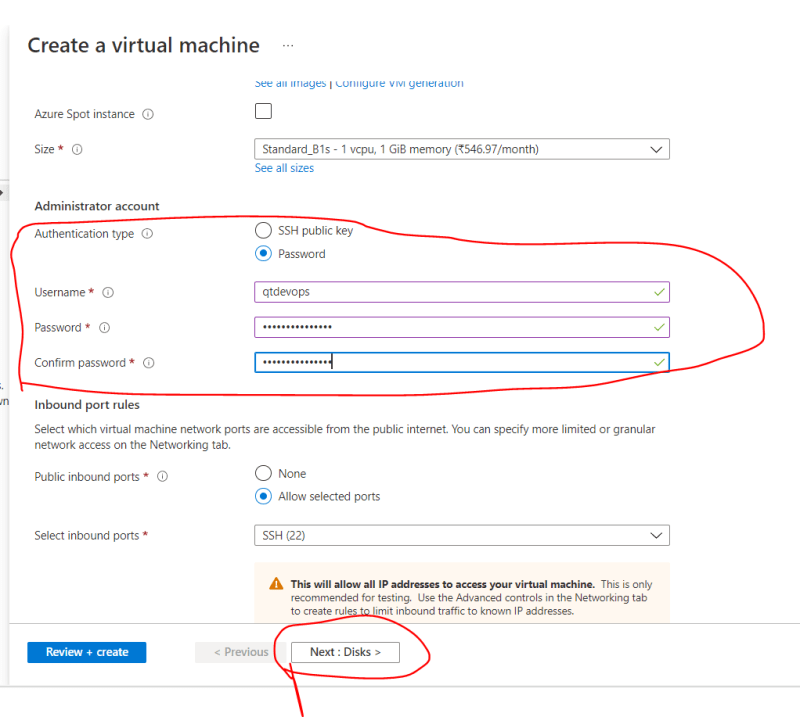
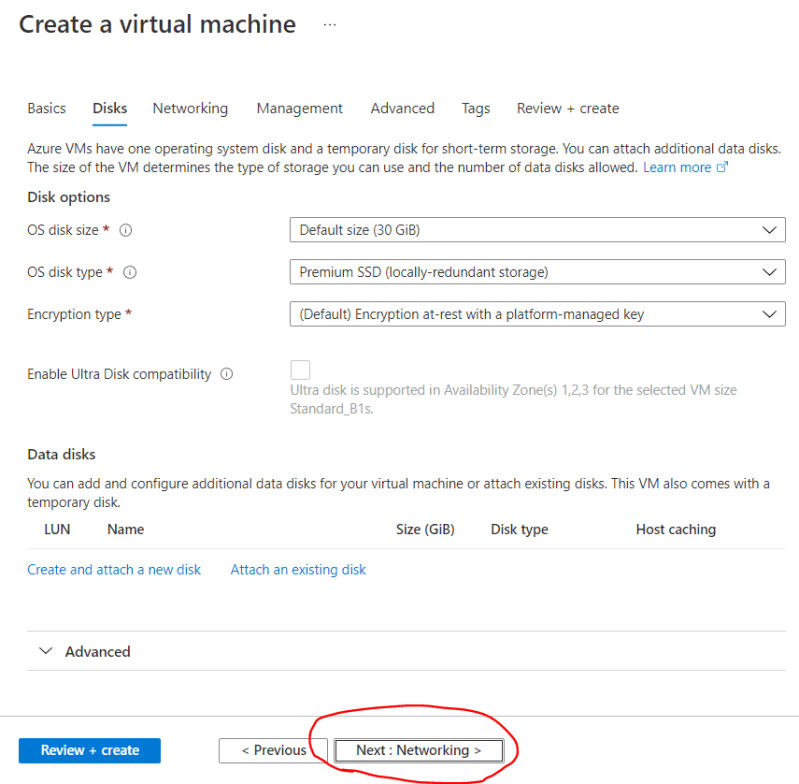
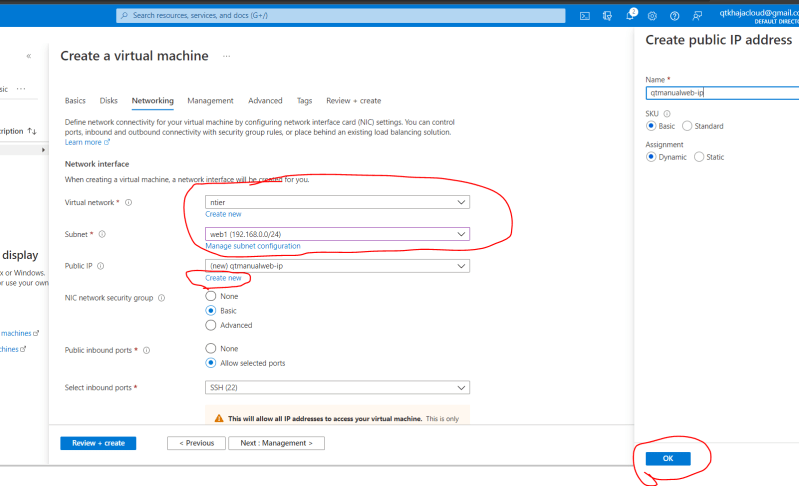
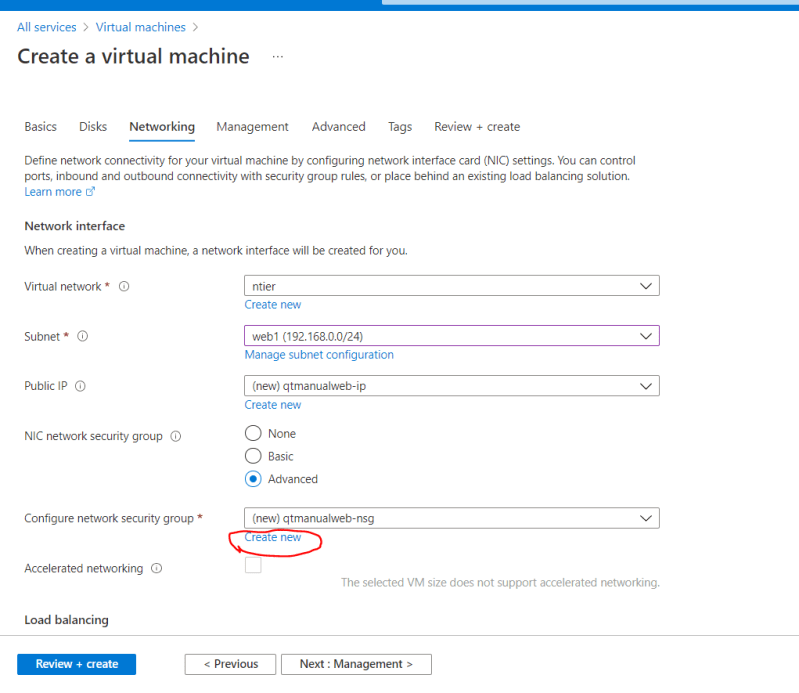
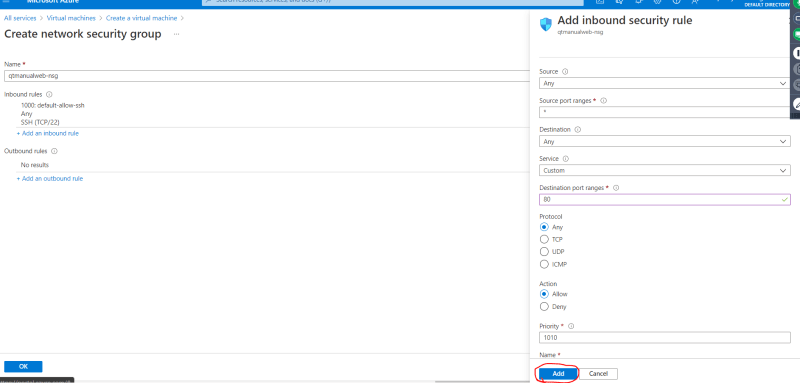
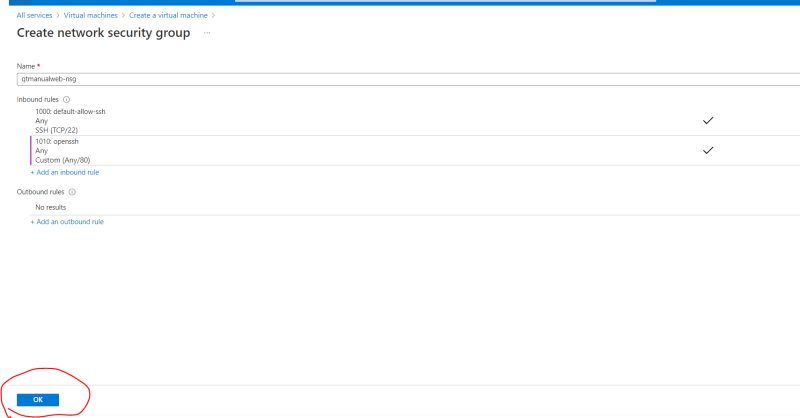
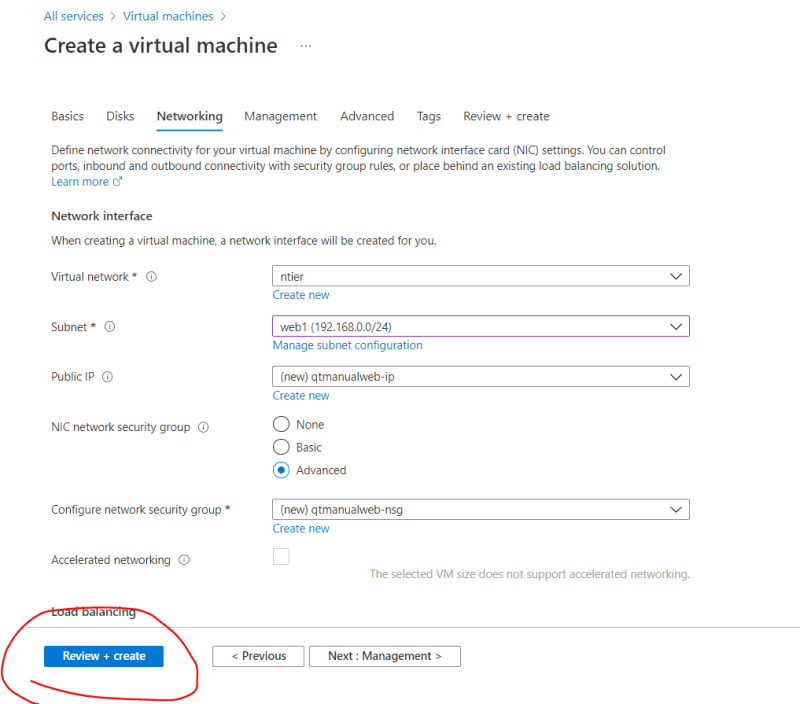
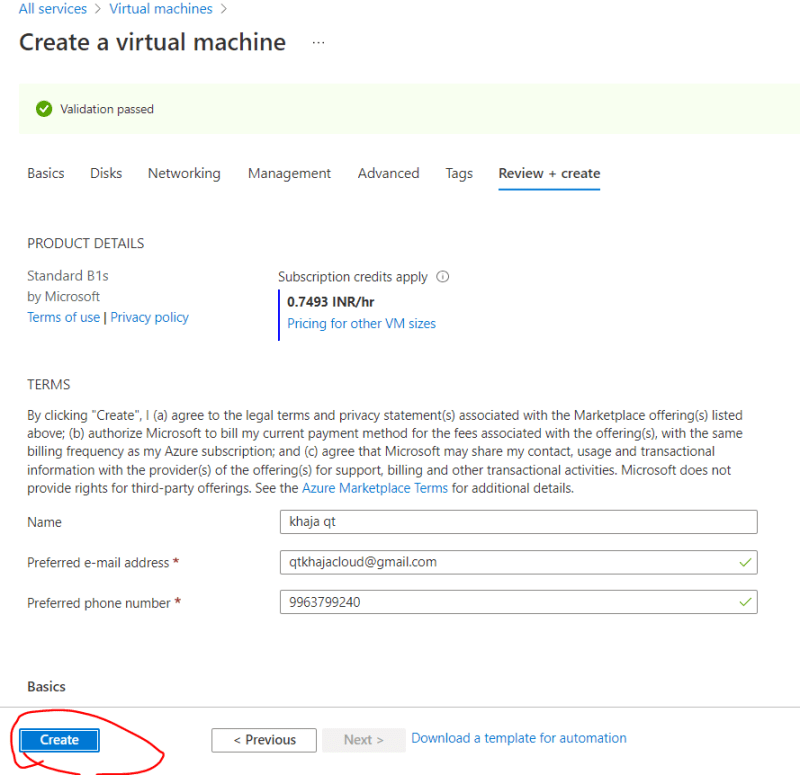
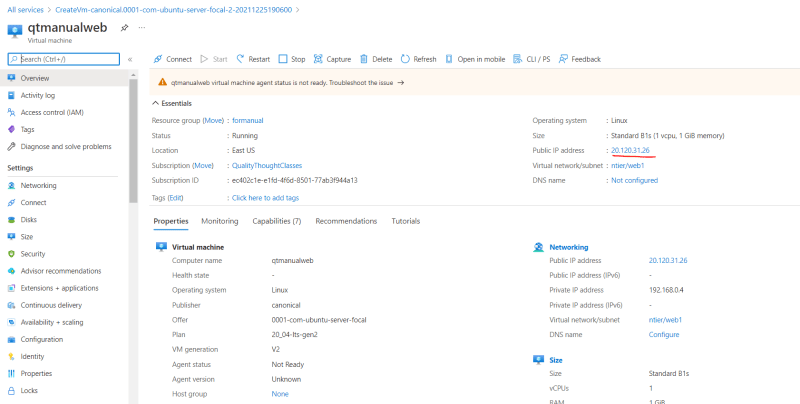
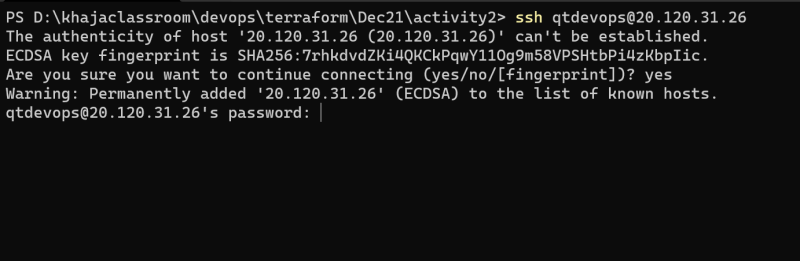
DECEMBER 25, 2021

DevOps Classroomnotes 25/Dec/2021

**Activity 2: Contd**

* We have two things left
* Error which we got while destroying. [Refer Here](https://github.com/asquarezone/TerraformZone/commit/394cd48072e39cbafd9d5001b4518cc68d93da3b) for the fix  
  
* Deploying the application on the linux vm
  + This will be done post the 3 acvitity

**Activity 3: Create a ntier architecture in Azure**

* Reference Architecture  
  
* In Azure Every resource has to belong to a resource group  
    
  
* Now lets create a virtual Network  
    
    
    
    
  
* In Azure
* any resource by default is public if it has public ip & private if it doesnot have public ip.
* To route the traffic from internet into vnet and with in vnet we dont need to create a route table because this is part of default routing.
* Linux Virtual machines support key based authentication as well as password based authentication. User needs to provide username & Key/password
* Now after creating vnet look into the resource  
    
  
* Now lets create a linux vm with ubuntu in web1 subnet  
    
    
    
    
    
    
    
    
    
  
* Once the vm is create let try to login into the vm  
    
  
* Now for realizing this activity in Azure using terraform we need a provider [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs)for the azurerm provider
* This can be done by adding the provider.tf with the following and now execute terraform init

terraform {

required\_providers {

azurerm = {

source = "hashicorp/azurerm"

version = "=2.46.0"

}

}

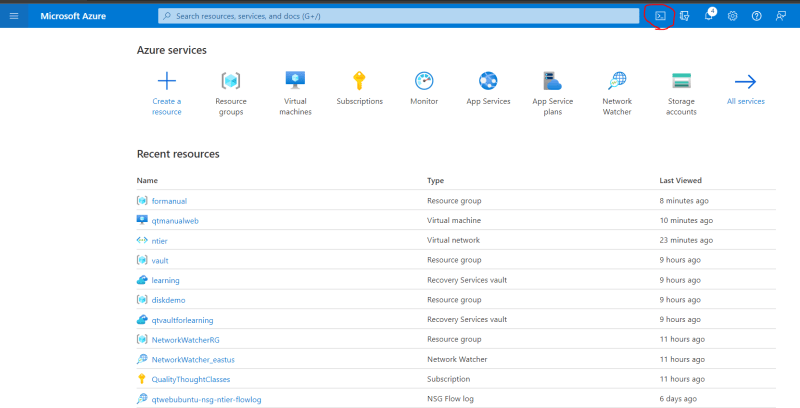
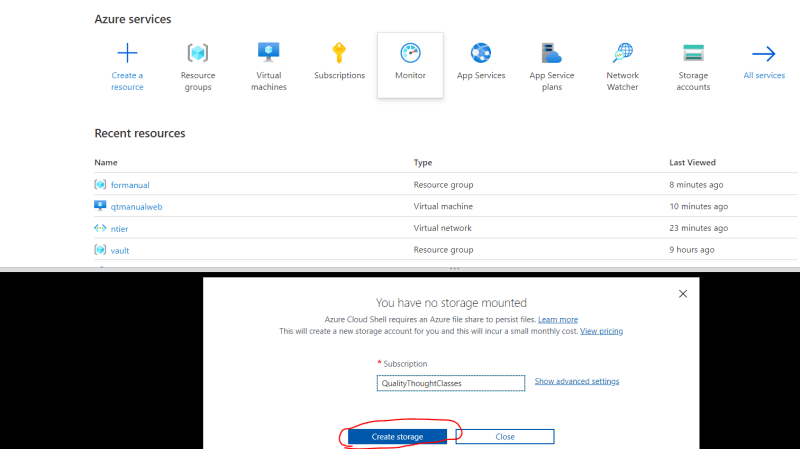
}

# Configure the Microsoft Azure Provider

provider "azurerm" {

features {}

}

* Authentication in Azure from terraform can be done in multiple ways, generally on ci based systems we would use service prinicipal approach [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/service_principal_client_secret) and in developer systems we can use cli based approach. [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/guides/azure_cli)
* To review all the authentication mechanisms [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs#authenticating-to-azure)
* Lets try to configure azure with Service Principal
* Launch Cloud Shell  
    
  
* Now run the following commands and make a note of client\_id, client\_secret and tenant\_id. and then in the next command make a note of subscription id

az ad sp create-for-rbac --role Contributor --query "{client\_id: appId, client\_secret: password, tenant\_id: tenant}"

az account show --query "{subscription\_id: id}"

* We can pass client id, client secret, tenant\_id and subscription\_id from arguments of the provider. But it is not a good practice
* We will be creating environmental variables
* Linux

export ARM\_CLIENT\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_CLIENT\_SECRET="00000000-0000-0000-0000-000000000000"

export ARM\_SUBSCRIPTION\_ID="00000000-0000-0000-0000-000000000000"

export ARM\_TENANT\_ID="00000000-0000-0000-0000-000000000000"

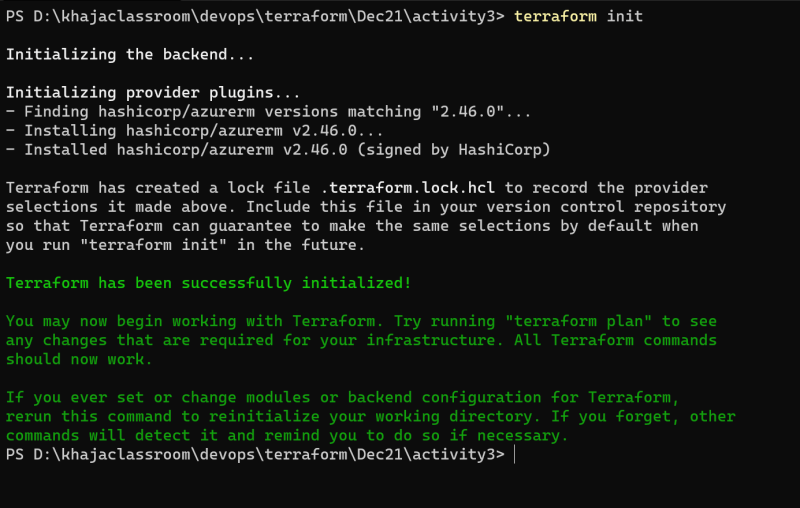
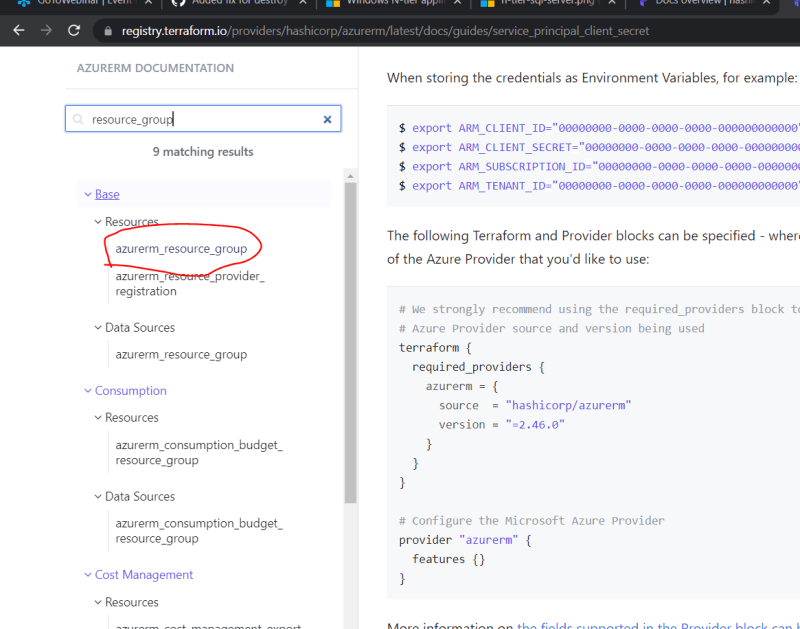
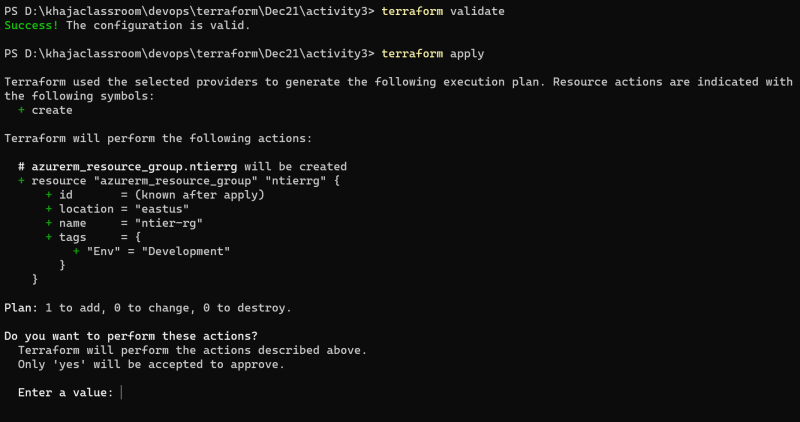
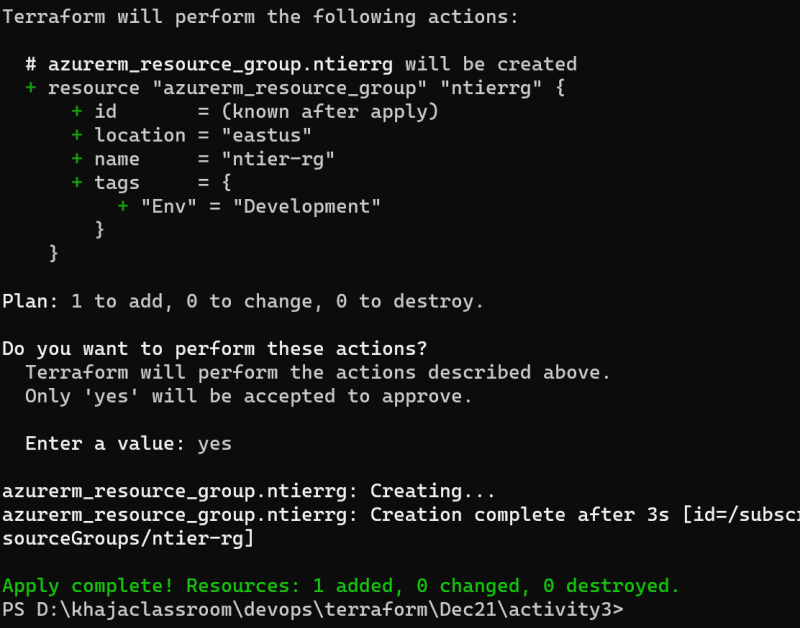
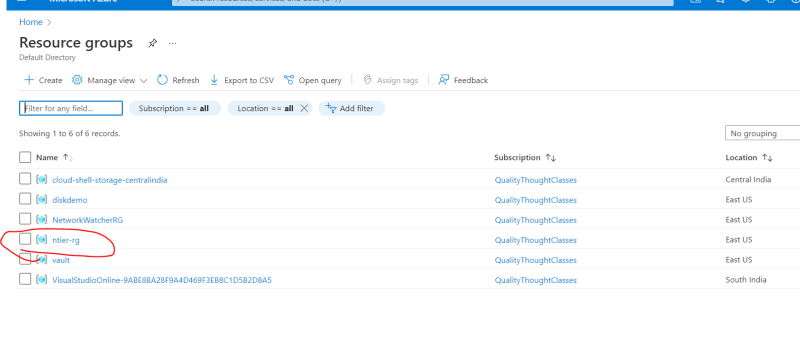
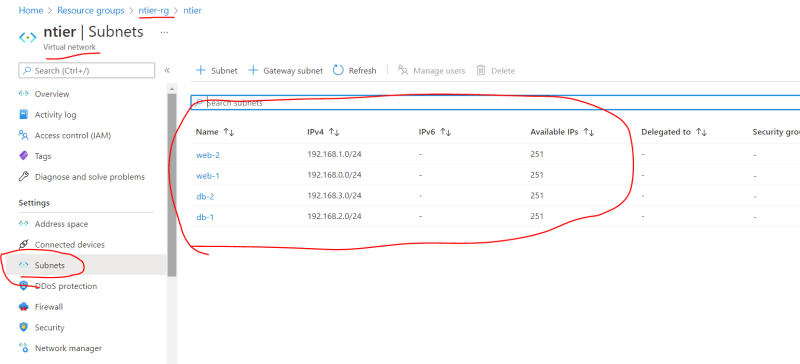
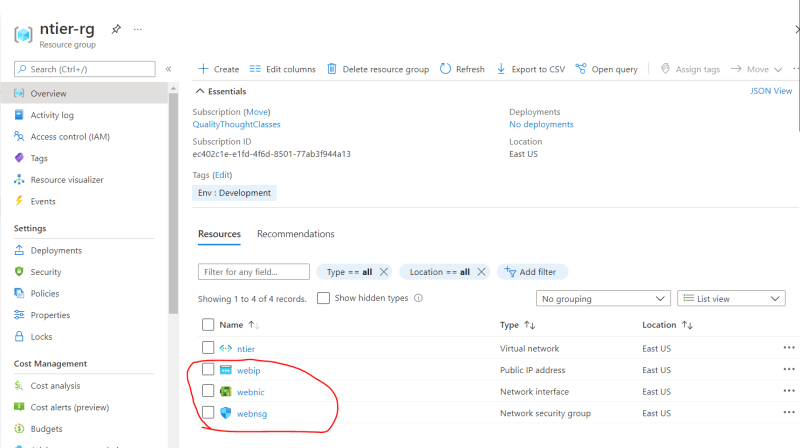
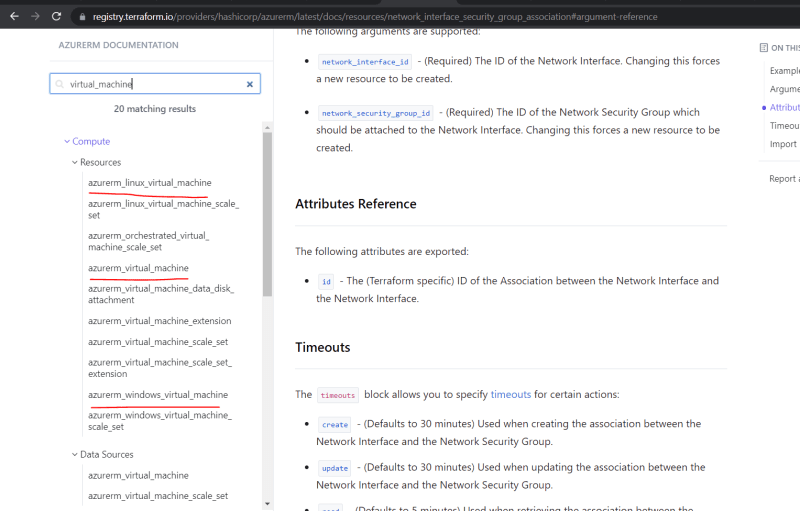
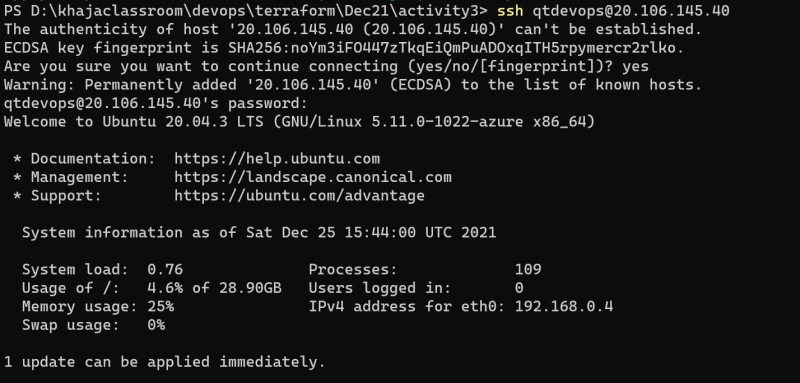
* Windows Powershell

$env:ARM\_CLIENT\_ID="00000000-0000-0000-0000-000000000000"

$env:ARM\_CLIENT\_SECRET="00000000-0000-0000-0000-000000000000"

$env:ARM\_SUBSCRIPTION\_ID="00000000-0000-0000-0000-000000000000"

$env:ARM\_TENANT\_ID="00000000-0000-0000-0000-000000000000"

* Lets execute terraform init  
  
* The first step is to create a resource group  
  
* [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/resource_group) for the official docs
* Now create a resource group. [Refer Here](https://github.com/asquarezone/TerraformZone/commit/874c22554660c25657d979e71c61e1c094870b0e) for the changes.  
    
    
  
* Now lets create a virtual network. [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/virtual_network) for the official docs
* Now lets add subnets, while adding subnets lets use dynamic functions to calcuate cidr ranges and make use of explicit dependencies. [Refer Here](https://github.com/asquarezone/TerraformZone/commit/325e4503299c28cdffdda9815076bc9dd9513931)  
  
* Now we need to create a network security group for web (80 and 22 port open to all) [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_security_group) for the official docs
* Now lets create a public ip address [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/public_ip) for official docs
* Now lets try to create network interface [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_interface)
* Now we need to associate nsg to nic [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/network_interface_security_group_association)
* At this point we have created all the network related elements
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/04fad776842d0ca4b6ab6505724a4d08bdd5be5f) for the changes done  
  
* Now we need to create a linux vm with ubuntu 20.04  
  
* Lets use azurerm\_linux\_virtual\_machine. [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux_virtual_machine) for the official docs
* Now apply the changes [Refer Here](https://github.com/asquarezone/TerraformZone/commit/0593b02cf8e6272f5af9f886a285367cfddc514e) for the changes done  
  

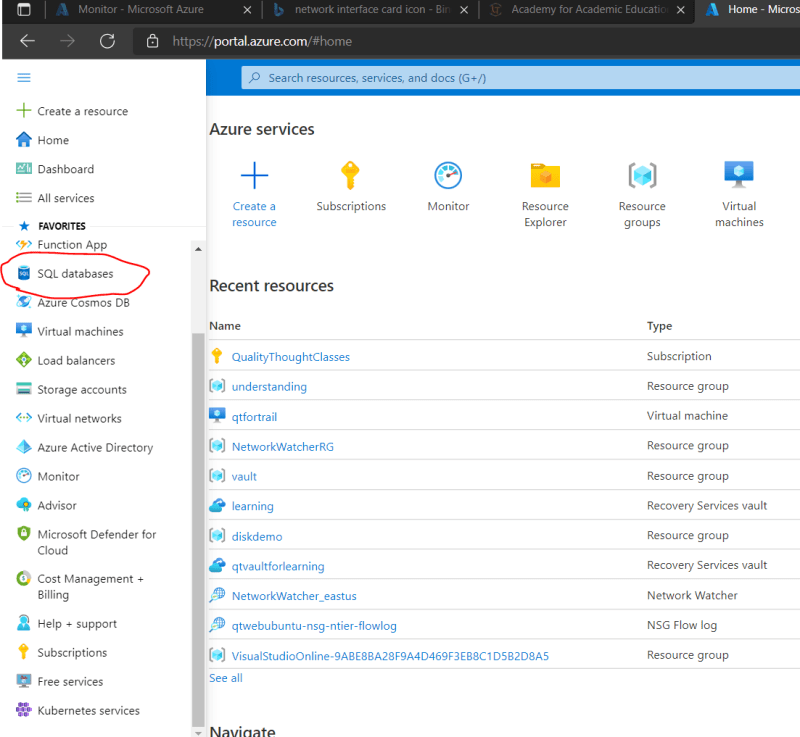
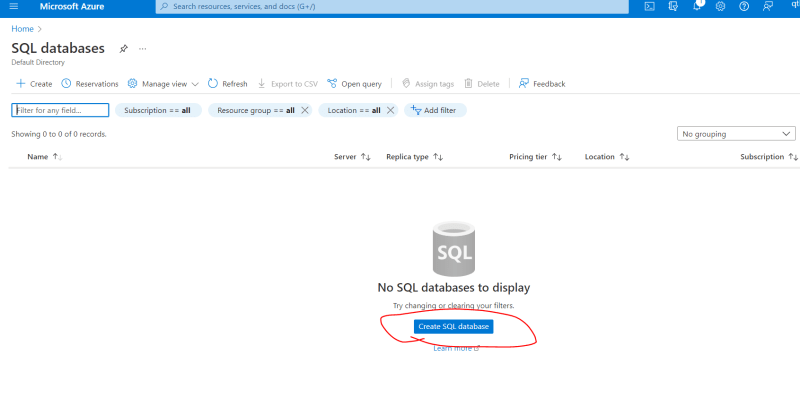
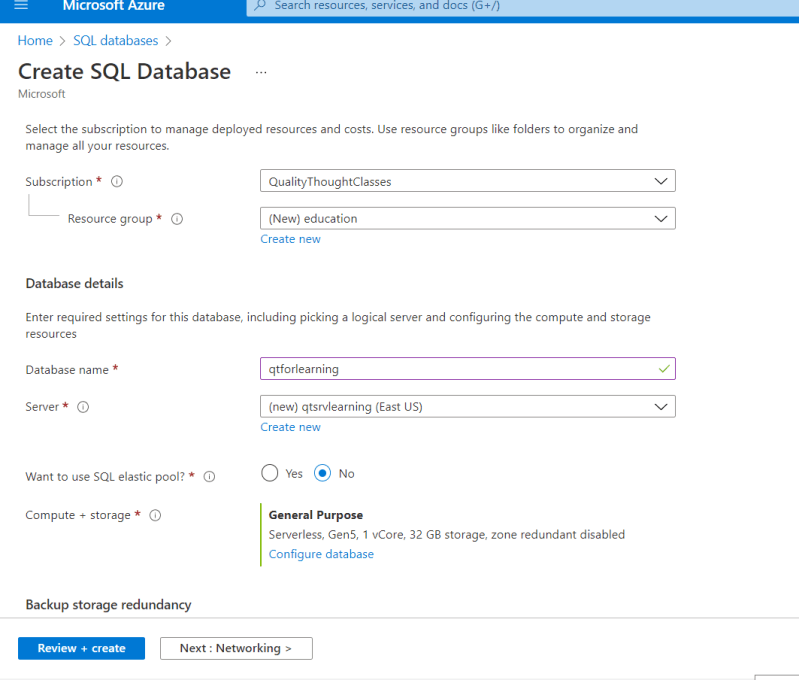
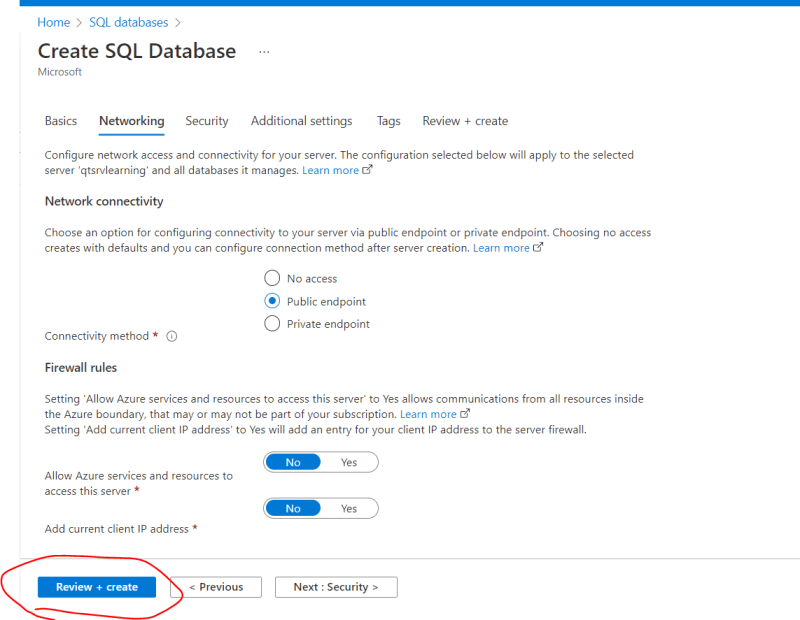
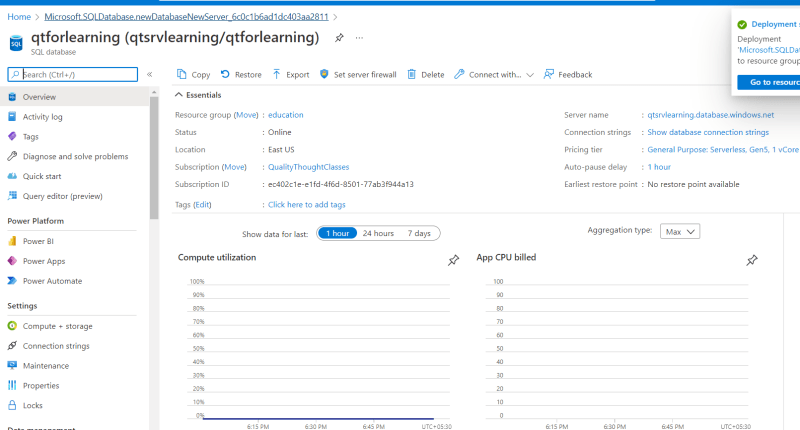
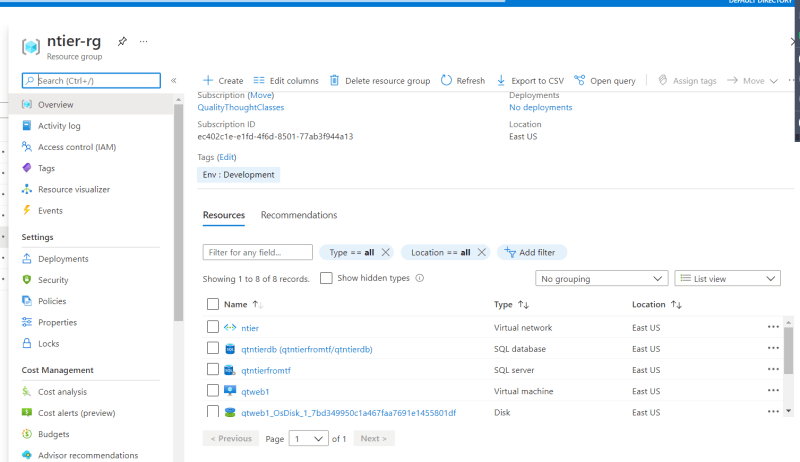
**Share this:**

* [Click to share on Twitter (Opens in new window)](https://directdevops.blog/2021/12/25/devops-classroomnotes-25-dec-2021/?share=twitter&nb=1)
* [Click to share on Facebook (Opens in new window)](https://directdevops.blog/2021/12/25/devops-classroomnotes-25-dec-2021/?share=facebook&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://directdevops.blog/2021/12/25/devops-classroomnotes-25-dec-2021/?share=jetpack-whatsapp&nb=1)

DECEMBER 26, 2021

DevOps Classroomnotes 26/Dec/2021

**Activity 3: Create a ntier architecture in Azure contd**

* So we need to create a database
* Manually creating a database  
    
    
    
  
* While creating a database we need to provide
* database details
* server details
* credentials
* size
* Network endpoints  
  
* Lets create a sql server [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/sql_server)
* Lets create an azure sql database [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/sql_database)
* Now apply the configuration. [Refer Here](https://github.com/asquarezone/TerraformZone/commit/f86c3d91ef93bef6b0b8fc322b819f11965251f9) for the changes done and check the resources in the resource group  
  
* Now lets add a vnet network rule [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/sql_virtual_network_rule) and firewall rule [Refer Here](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/sql_firewall_rule)
* Lets try to use the following conditional expression to create a service endpoint for db-1 subnet

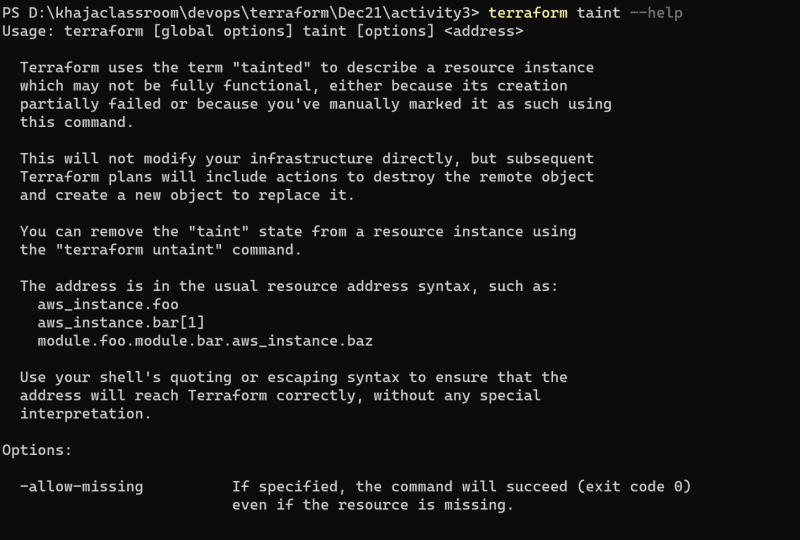
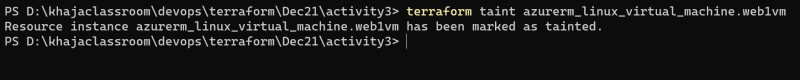
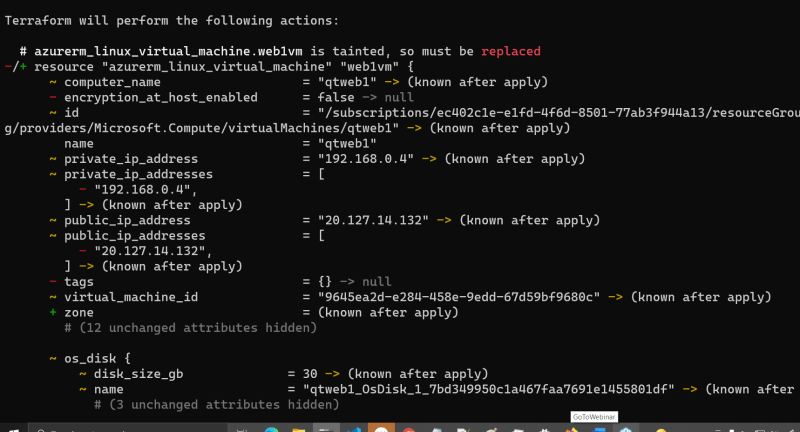
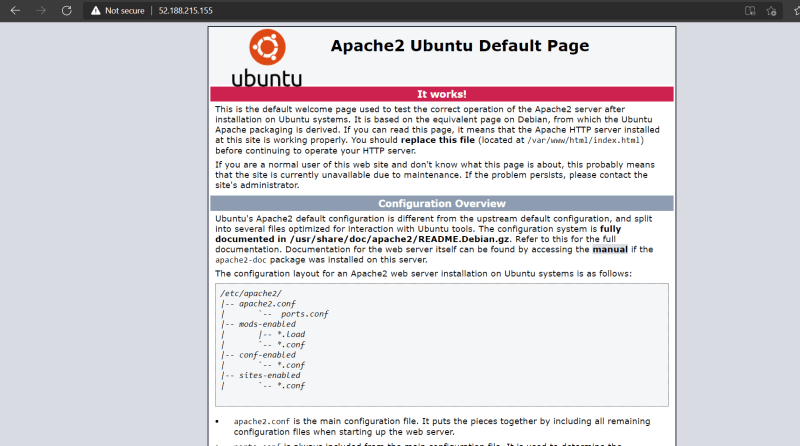
var.subnet\_names[count.index] == "db-1"? ["Microsoft.sql"]: []

* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/403879fd0c20067ee8fee70da59f94a0f24da426) for the changes done.

**Terraform plan**

* All these days we were executing terraform apply to create infrastructure
* Terraform apply command internally creates a plan and then executes the plan
* We can explicity create a plan terraform plan -out <filename> and then terraform apply <filename>

**Terraform Provisioners**

* Terraform provisioners can execute specific actions on the local machine or on the remote machine in order to prepare server or other infrastructure objects.
* If you are using local machine provisioners we need not establish any connections, but if we are using remote machine provisioners we need connections
* To specify connections for remote machines terraform has a connection block [Refer Here](https://www.terraform.io/language/resources/provisioners/connection) for the official docs
* Lets try to use remote-exec connection to execute a inline script on the linux server
* Terraform provisioner written with in resource will be executed only when that resource is created.
* Solution 1: is to taint the resource i.e. mark the resource for recreation.  
  
* Resource can be marked to taint by execution terraform taint <resourcetype>.<resourcename>  
  
* Now if we execute terraform apply the resource will be destroyed and recreated.
* Execute Terraform apply  
    
  
* Taint is designed to recreate the resource when you observe mismatches
* Note: before apply command is executed if you want to undo taint terraform untaint <resourcetype>.<resourcename>
* This approach will execute the script only once while creation.
* Now lets assume you have written a shell script for deploying application & whenever you execute terraform apply you want to do provisioning as we might be deploying the new version of the application generated from recent build.
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/e05e4ccb3b21416c1f2983eb1bb9df9a62790aeb) for the changes
* Solution 2: Use Terraform null resource in null provider.
* The null resource will be executed every time you execute apply command. [Refer Here](https://registry.terraform.io/providers/hashicorp/null/latest/docs/resources/resource) for null resource documentation
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/a336970b6b896710f69843280a0b9988744bff7a) for the changes done to include the null provisioner
* Note we have used build\_id as trigger, so whenever build\_id changes the terraform will executed the provisioner

terraform apply -var "build\_id=2" -auto-approve

DECEMBER 28, 2021

DevOps Classroomnotes 28/Dec/2021

**Terraform Provisioning**

* Lets provision the virtual machine to run a java application spring-petclinic [Refer Here](https://referenceapplicationskhaja.s3.us-west-2.amazonaws.com/spring-petclinic-2.4.2.jar) for the java application
* Steps for manually installing
* install openjdk 11
* dowload the jar file and execute the jar file by the following command java -jar <path to spring-petclinic.jar>
* commands in linux

sudo apt update

sudo apt install openjdk-11-jdk -y

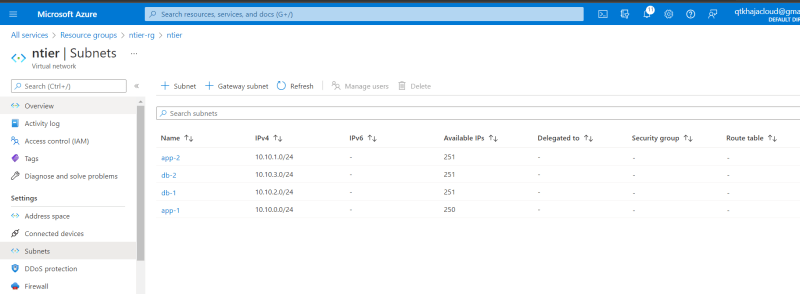
mkdir ~/apps && cd ~/apps

wget https://referenceapplicationskhaja.s3.us-west-2.amazonaws.com/spring-petclinic-2.4.2.jar

java -jar spring-petclinic-2.4.2.jar &

* This application works on port 8080 so security group rule needs to be created
* Lets use file provisioner to copy the shell script from local machine to remote machine
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/7e23e1c616154961b4aac4680ad5e7c3e59d9517) for the changes done and these changes deploy spring petclinic
* Known issue to be fixed: The application is not running after terraform closes the remote-exec session.
* Passing variables from commandline every time is not a good idea.
* So create a file with .tfvars extension and pass them to terraform apply command

terraform apply -var-file="<filename.tfvars>"

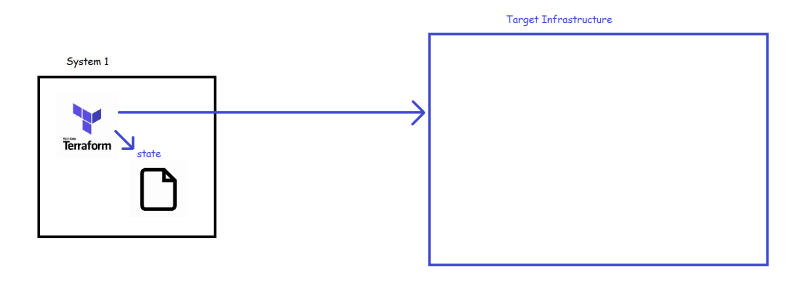
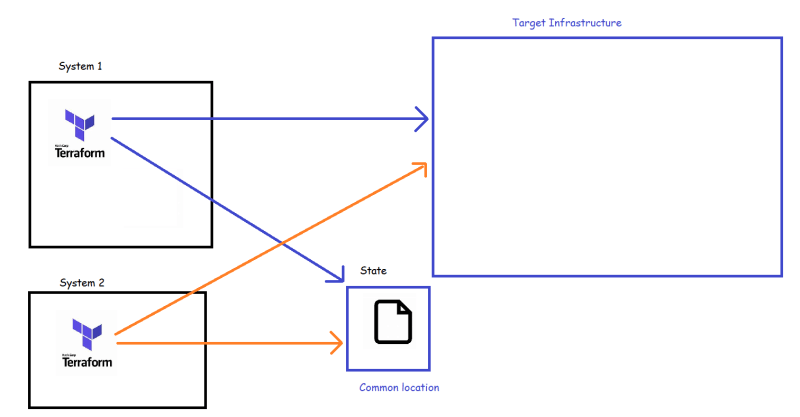
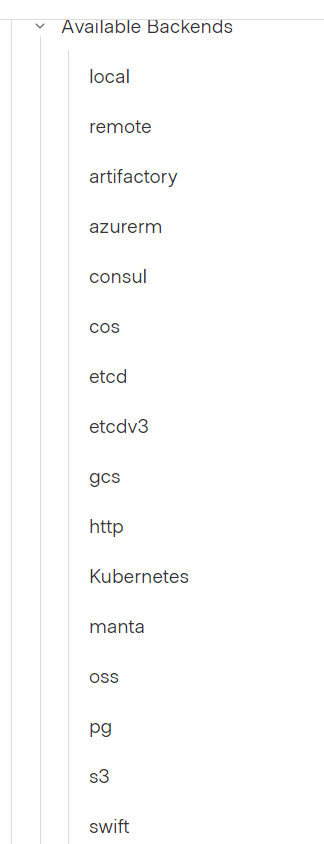
  
\* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/23827c7036bea6100427fcbe09912dae1666aaaf) for the changes done  
\* Next Steps:  
\* From terraform we will call the ansible playbook and chef cookbook  
\* Backends  
\* Note: try this stuff on AWS

**Share this:**

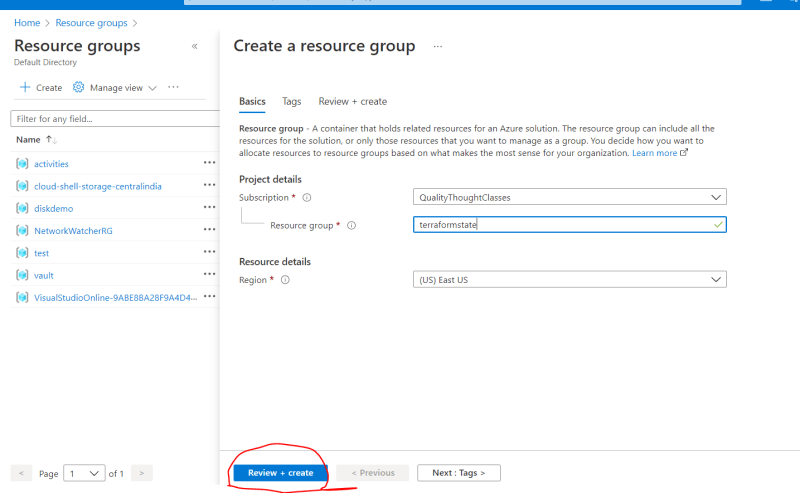
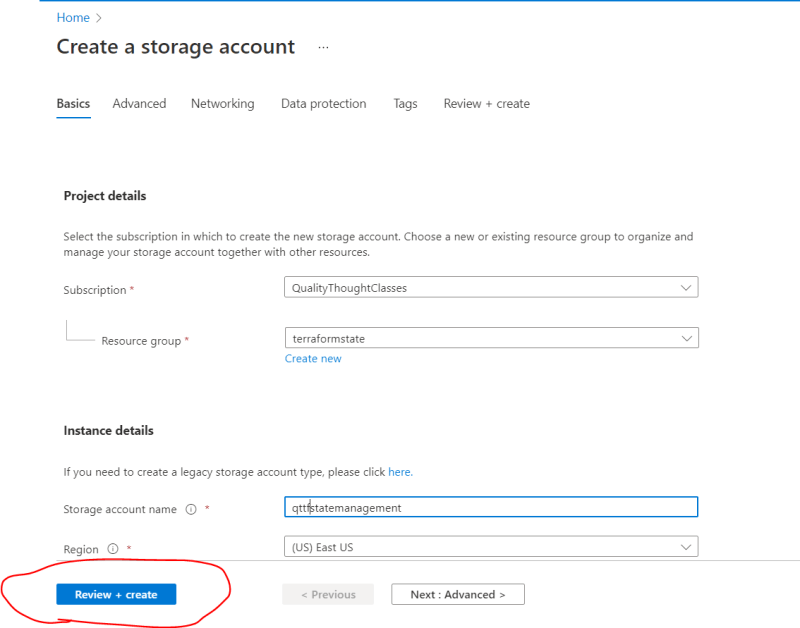
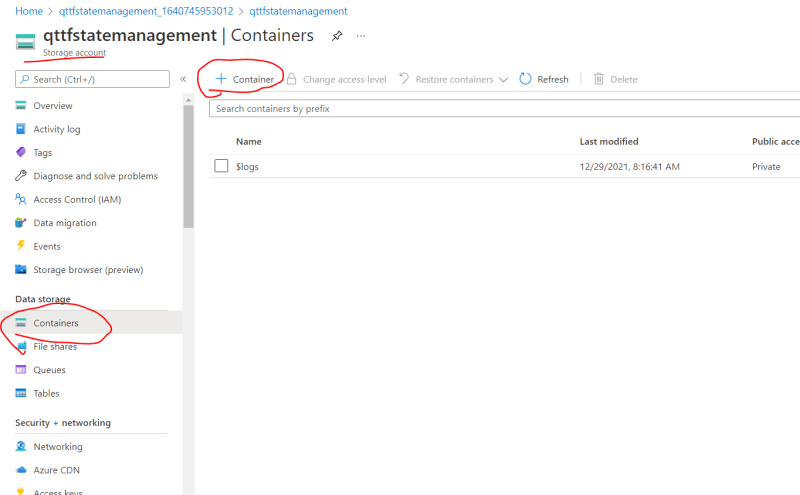
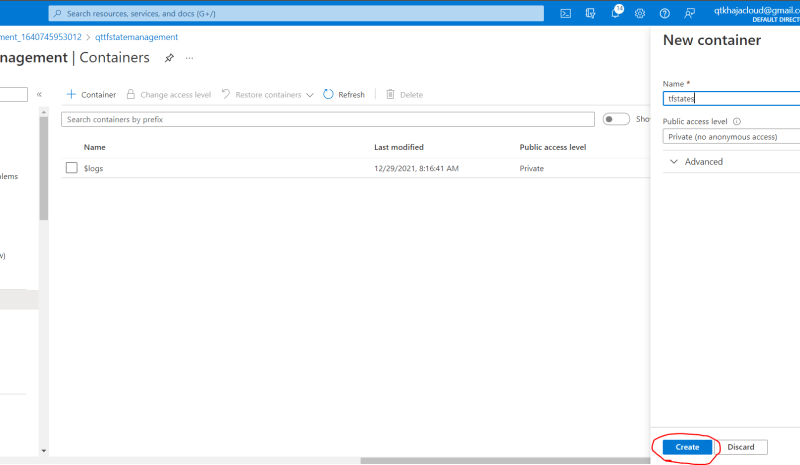
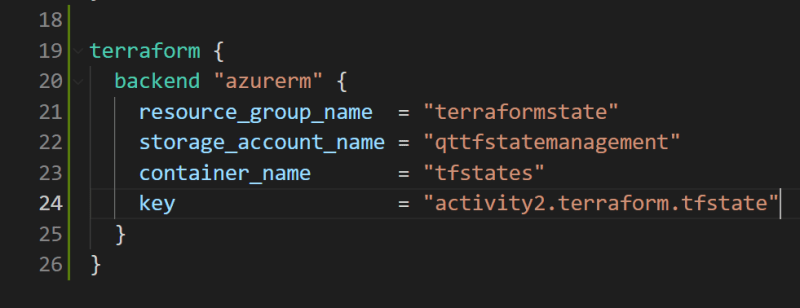
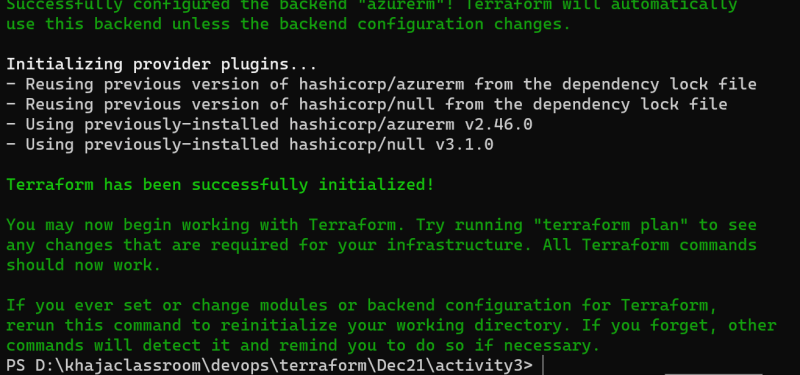
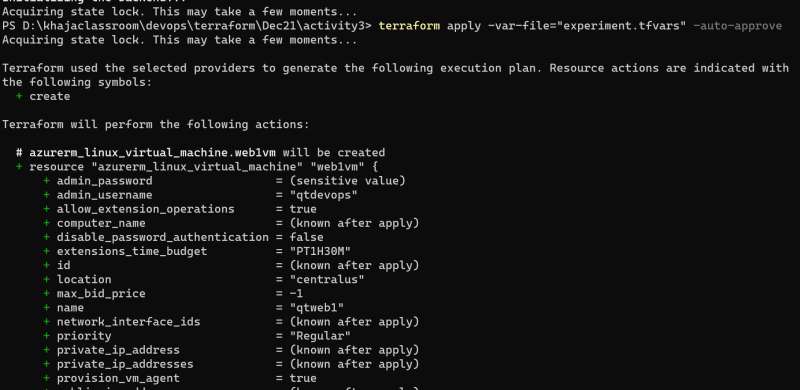
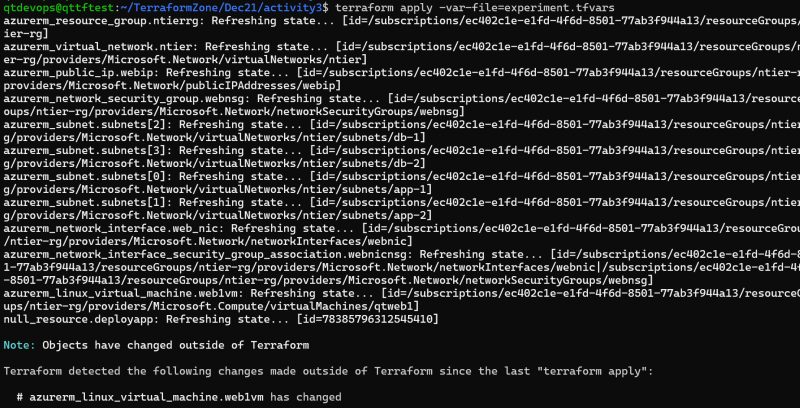
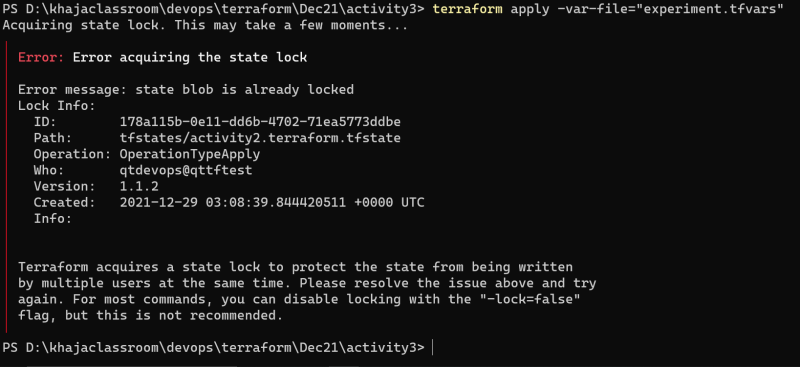
DECEMBER 29, 2021

DevOps Classroomnotes 29/Dec/2021

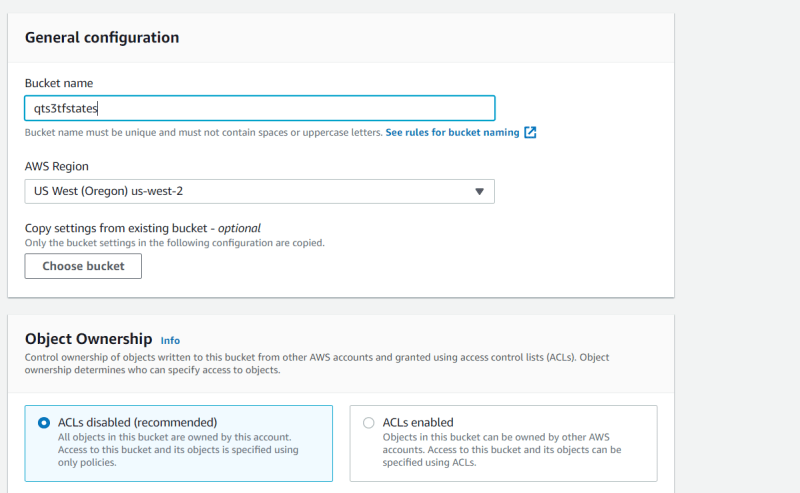
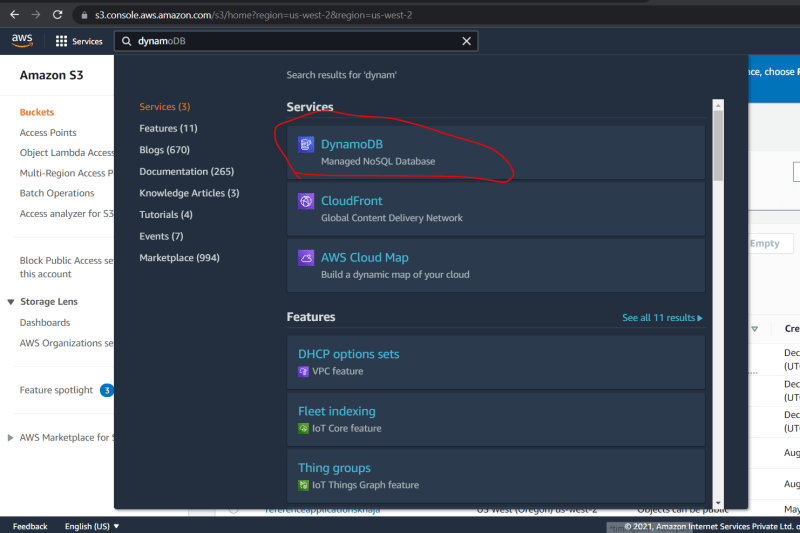
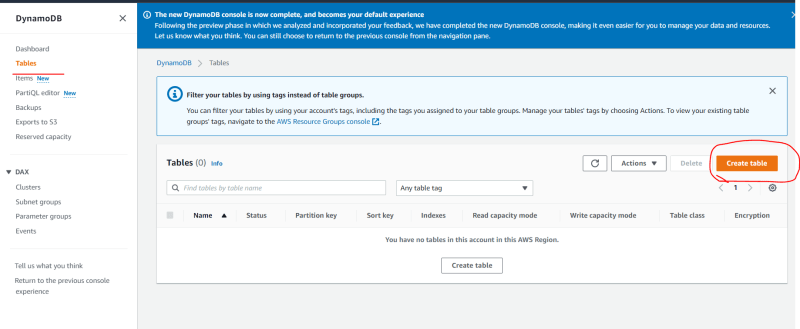
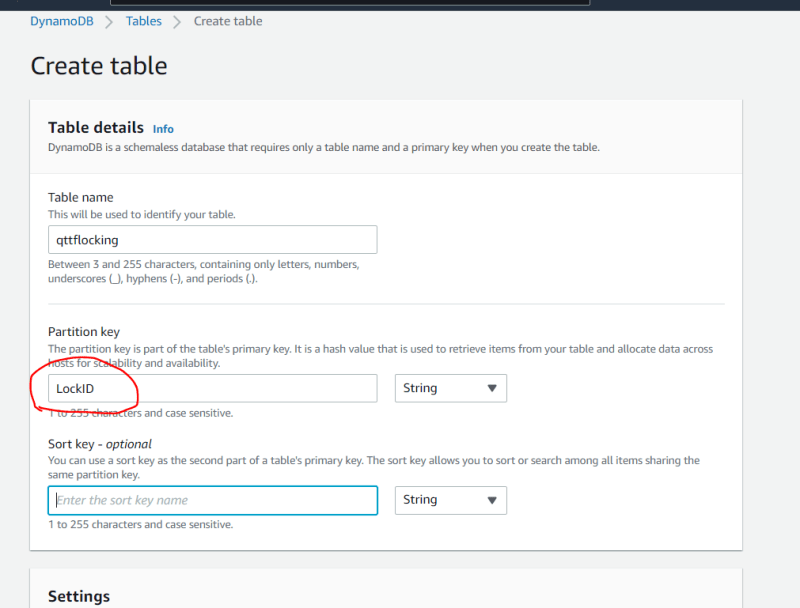
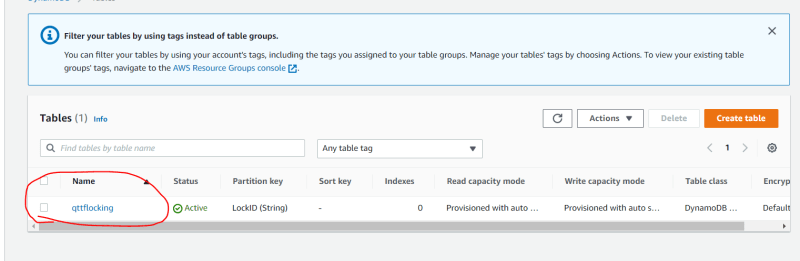
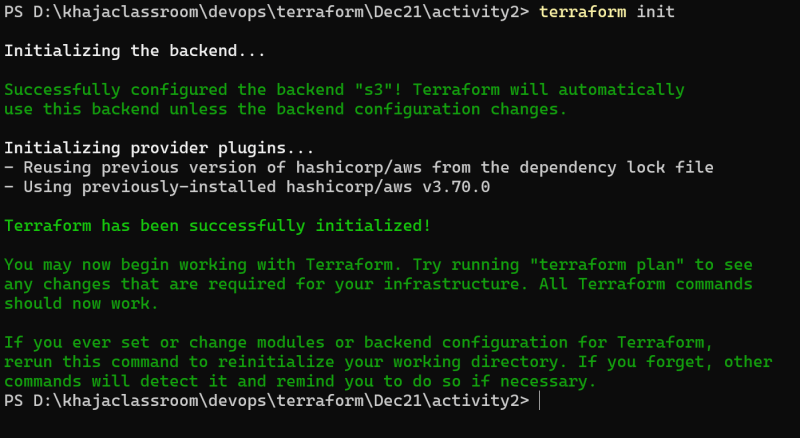
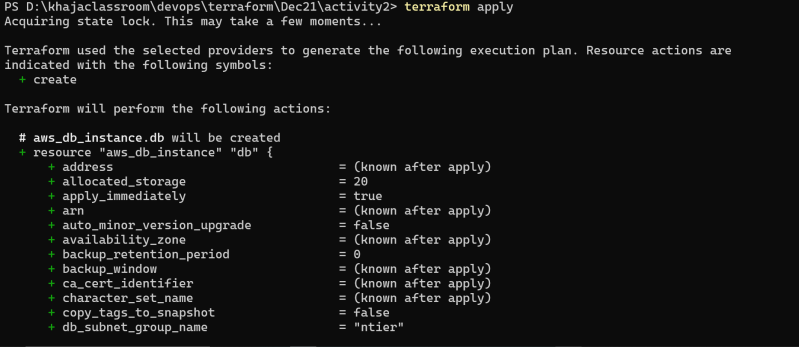
**Terraform Backends**

* Terraform by default when applied to create infrastructure will create a local state file, so when multiple users execute the terraform on multiple machines it leads to creation of multiple target infrastructures which is not idle.  
  
* The solution for this is if we can store the state file in some common location which is referred as Backend, then even if multiple users try to apply it leads to same infrastructure.  
  
* To avoid multiple users writing to the state file at the same time, terraform implements locking
* [Refer Here](https://www.terraform.io/language/settings/backends) for the official docs of the backend
* Terraform supports the following backends  
  
* Lets first implement the backend using azurerm and then we will implment s3 based backend

**Azurerm Backend**

* Lets create a resource group  
  
* Now create a storage account  
  
* Once the storage account is created, create a container  
    
  
* Now lets add the backend to our terraform  
  
* Now execute terraform init to initialize the backend  
  
* Now lets execute terraform apply -var-file="experiment.tfvars" -auto-approve  
  
* To simulate the multi user scenario, let create a new linux vm and install terrform over there and see parallel executions
* Now lets apply terraform on linux system  
  
* Before the state has been applied lets try to simulate apply by other user i.e on my windows system  
  

**AWS S3 Backend**

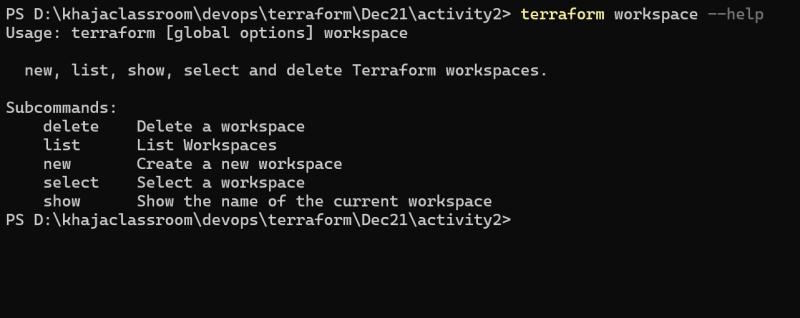
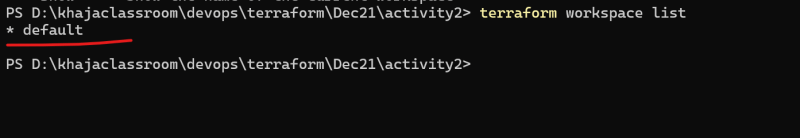
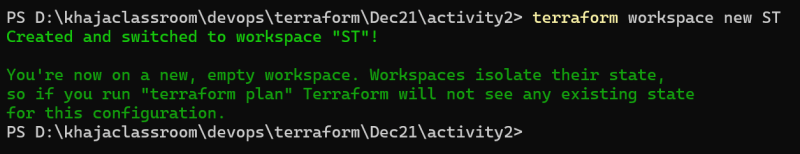
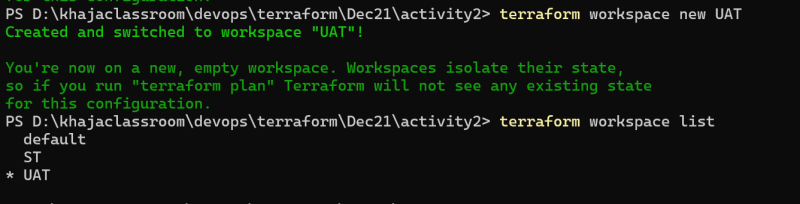
* [Refer Here](https://www.terraform.io/language/settings/backends/s3) for official docs
* Create an s3 bucket  
  
* Lets create a dynamodb table for locking purposes  
    
    
    
  
* Now initialize  
  
* And Apply  
  
* As this user has acquired the lock which will be released upon completion of terraform execution till that moment no other user will be able to create infra.

**Share this:**

DECEMBER 30, 2021

DevOps Classroomnotes 30/Dec/2021

**Terraform Workspaces**

* Generally when we write terraform configuration, we would like to create multiple environments such as
* Developer
* System Test
* Performance Test
* Staging
* Production
* For all the environments our infrastructure is same, the arguments might differ, There might be some additional resources in some environments
* To handle this we would not creat multiple copies of infrastructure and create different backend configurations.
* To deal with single template and multi environments terraform introduced a concept called workspaces. [Refer Here](https://www.terraform.io/language/state/workspaces) for the official docs
* Now lets explore terraform workspace cli  
    
  
* Lets try to create two workspaces
* ST => System Test  
  
* UAT => Staging  
  
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/483615e478afe1d331248c495484a29efc4e368c) for some of the changes done
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/c66c17d4fdf73e3f0c0c47c832c98d9aab42dc61) for the fixes done to the output variable
* Select the current workspace as ST

terraform workspace select ST

terraform apply -var-file="env/ST.tfvars"

* Now lets select the current workspace as UAT

terraform workspace select UAT

terraform apply -var-file="env/UAT.tfvars"

* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/803891d95301b2d379da0f776975ecf177b6c84e) for some of the changes done to accomodate multi environments
* To set the current workspace we are using terraform workspace select, the other way of doing SET TF\_WORKSPACE to the active environment
* Other environmental variables of terraform [Refer Here](https://www.terraform.io/cli/config/environment-variables)
* Exercise: Try to create all the tag with with env-<name>

tags = {

Name = format("%s-WebSg", terraform.workspace),

Env = terraform.workspace

}

* From CI/CD Engines when we need manage Terraform infra creation. From the Jenkins/Azure DevOps pipelines, You will have an agent/node where terraform is installed and credentials configured/passed

# if the terraform code is available in some git try to configure the git to clone

terraform init

terraform workspace select <build-parameter>

terraform apply -var-file=<varfile> -auto-approve

* Next Steps:
* Reusable Templates => Modules => Registry
* Creating a Module and using community module

**Share this:**

DECEMBER 31, 2021

DevOps Class room notes 31/Dec/2021

**Terraform Modules**

* Terraform moduels are container which have resources defined together.
* There are two kinds of modules
* Published Modules: These modules are published in Terraform Registry [Refer Here](https://registry.terraform.io/browse/modules)
* Local/Private Modules

**Activity 4: Using Published Terraform Module to Create VPC in AWS & vnet in Azure**

* Use the published module to create vpc [Refer Here](https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest)
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/198db66f0f79300cf3e4d2f15044b87a5fb9a52f)
* Now lets try to use a module to create a virtual network in azure
* [Refer Here](https://github.com/asquarezone/TerraformZone/commit/%20fc3f7651bdf965492dad1f8e4e4ba3d0bdebc97b) for the changes
* To use the module [Refer Here](https://www.terraform.io/language/modules/syntax) for module syntax
* Modules can be hosted in Registry or git or various other supported sources [Refer Here](https://www.terraform.io/language/modules/sources)

**Activity 5: Lets create our own module and use it**

* Lets reuse our activity 3 virtual network created in Azure and make it a module and activity 2 also as a module
* Lets continue this activity on sunday