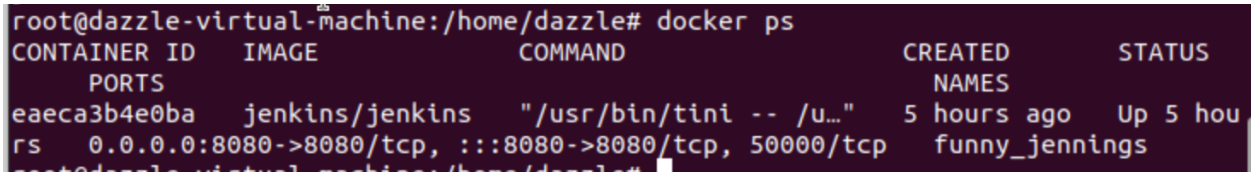
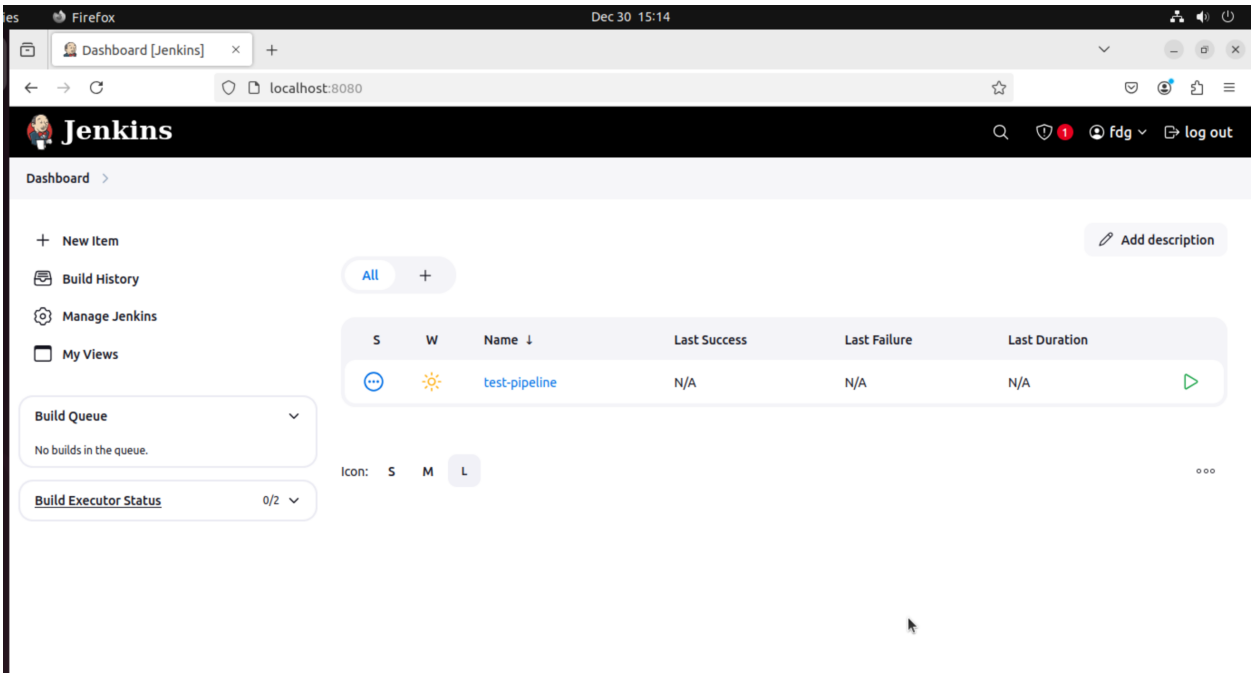
Deploying a virtual machine to Azure using Terraform Jenkins CI/CD

In this section, we will create Jenkins pipelines to deploy our infrastructure to Azure Portal. In our Terraform scripts, we would like to create a Virtual machine and its resources on Azure. To achieve this goal first we would like to install Jenkins on our Ubuntu system.

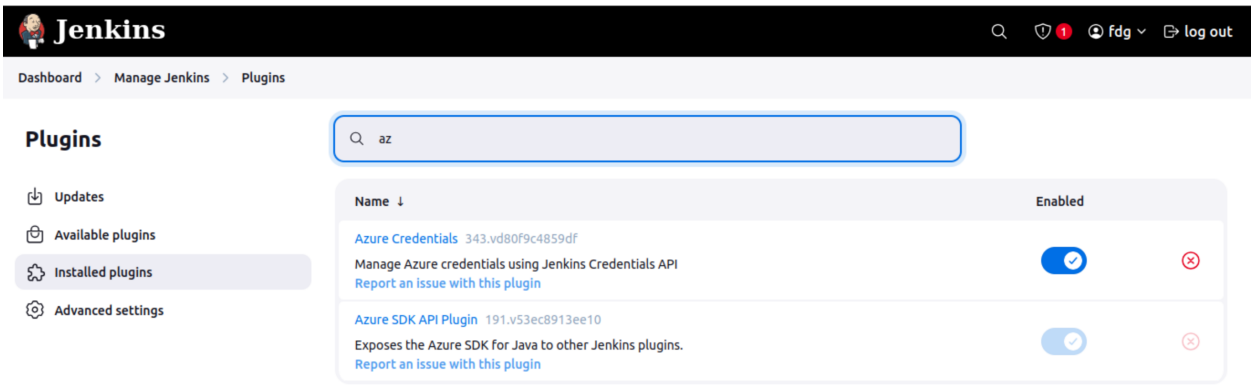
```
docker pull jenkins/jenkins
docker create volume jenkins-volume
docker run -d -v jenkins-volume:/var/jenkins_home -p 8080:8080 j
```

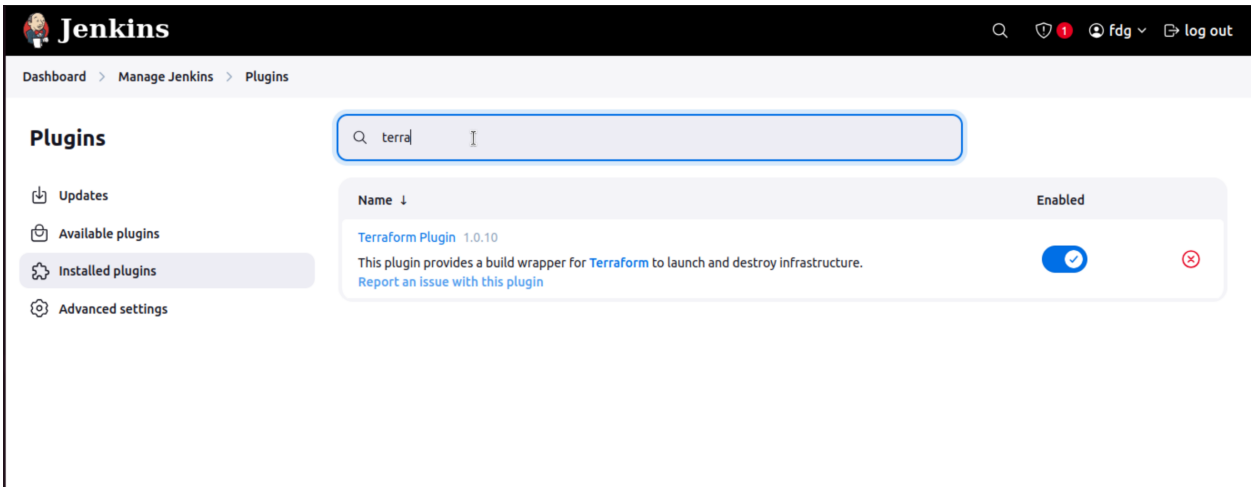
We can see that our jenkins now runs in a docker container;



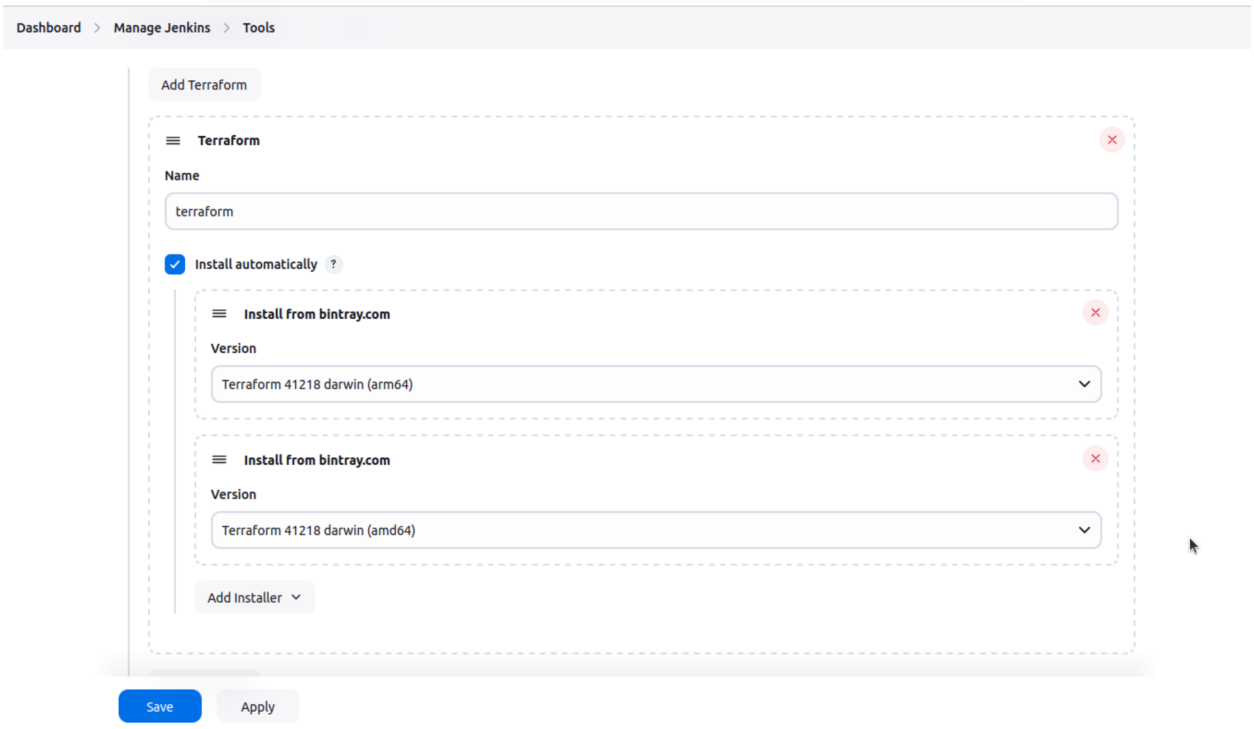


We need to install our plugins to Jenkins such as Terraform and Azure Credentials;



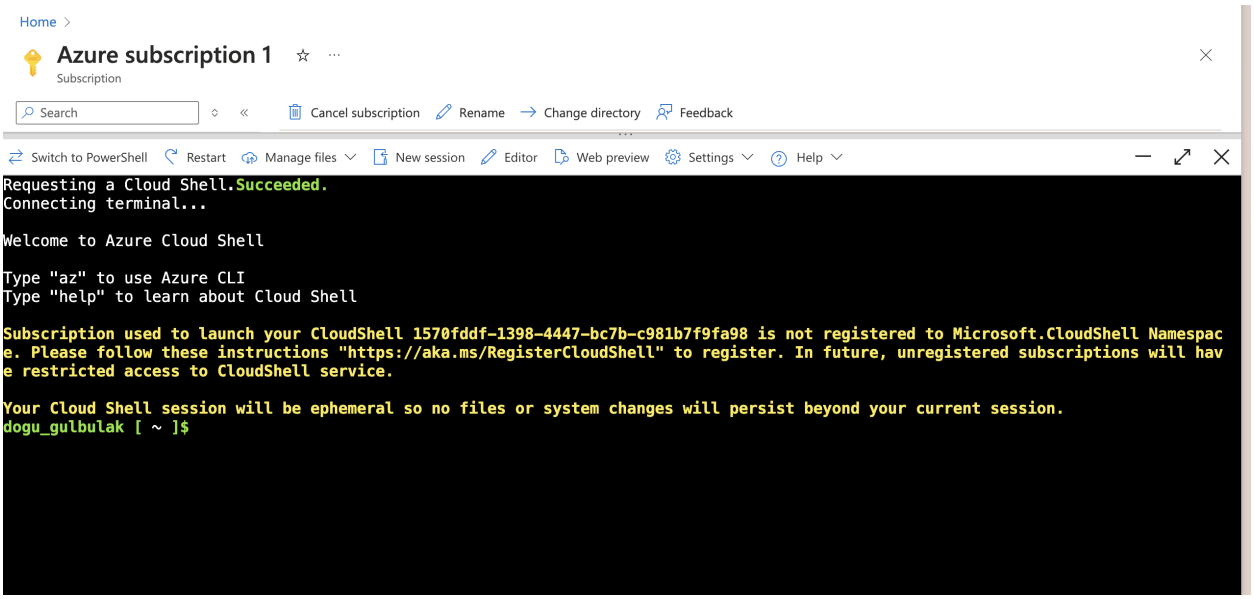


We need to add Terraform tool;



Now we would like to create an Azure Service Principle to use RBAC to provision our infrastructure (owner or contributor role).

Let’s move over to Azure CLI;




```
az account show --subscription 1570fddf-1398-4447-bc7b-c981b7f9

az ad sp create-for-rbac --name "jenkins-test-sp" --role contri

#If you want to assign new permissions to a service principal f

az role assignment create --assignee cd15bdaa-5a08-4440-a8af-ba
```

```
firat [ ~ ]$ az ad sp create-for-rbac --name "jenkins-test-sp" --role contributor --scopes /subscriptions/ba38195f-caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins
Found an existing application instance: (id) a5d9c9ac-8650-4a8e-b8d1-6f925f136469. We will patch it.
Creating 'contributor' role assignment under scope '/subscriptions/ba38195f-caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins'
The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into your source control. For more information, see https://aka.ms/azadsp-cli
{
  "appId": "cd15bdaa-5a08-4440-a8af-babfb22e4d07",
  "displayName": "jenkins-test-sp",
  "password": "en38Q~aNTSDPAD254qIBUsJoR8XLONc2K3Yngafq",
  "tenant": "f6eea791-8ab1-4047-9408-b47ff76cfda9"
}
```

We will use these to create a secret inside Jenkins Credentials to manage resources in Azure. To create a service principle you can follow this guide if you do not have necessary permissions;

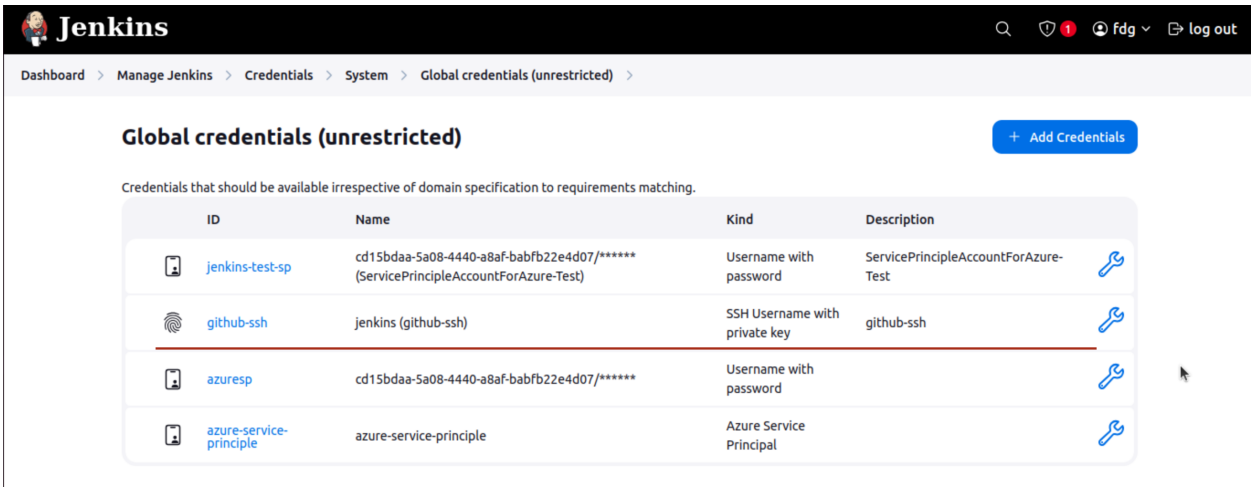
https://learn.microsoft.com/en-us/entra/identity-platform/howto-create-service-principal-portal

We will add the service principle to Jenkins Credentials as azure service principal. To add this as a credential on Jenkins, you need to install azure credentials plugin to Jenkins.

For more detailed steps;

https://learn.microsoft.com/en-us/azure/developer/jenkins/deploy-to-azure-app-service-using-azure-cli

For code checkout step from Github repo I added ssh key into Jenkins as a credential as well;



Our pipeline will consist of 3 basic steps, 1) azure login, 2) checkout terraform codes from github 3) terraform init and apply

Here is the pipeline:

```
pipeline {
    agent any
    environment {
        credentials = credentials('jenkins-test-sp')
    }


    stages {
```

```
        stage('Az-login') {
            steps {
                script {
                    withCredentials([azureServicePrincipal('azu
                    // Authenticate with Azure
                    sh '''
                    az login --service-principal \
                    --username $AZURE_CLIENT_ID \
                    --password $AZURE_CLIENT_SECRET \
                    --tenant $AZURE_TENANT_ID
                    '''
                    }

                }
            }
        }
        stage('Checkout Code') {
            steps {
                git url: 'git@github.com:0x24dazzle/test-jenkins
                    credentialsId: 'github-ssh',
                    branch: 'deploy-vm2'
            }
        }
        stage('Deploy') {
            steps {
                script {
                    //withAzureServicePrincipal('azure-service-p
                    sh '''




                    export ARM_SUBSCRIPTION_ID=$AZURE_SUBSCRIPTI
                    export ARM_CLIENT_ID=$AZURE_CLIENT_ID
```

```
                    export ARM_CLIENT_SECRET=$AZURE_CLIENT_SECRE
                    export ARM_TENANT_ID=$AZURE_TENANT_ID

                    terraform init -upgrade
                    terraform init
                    terraform apply -auto-approve
                '''
                    //}
                }

            }
        }
    }
}
```

Now let's move over to our linux terminal to create and push our terraform code.

Password authentication to github was removed. Create a public ssh and add it to your github repository. Then add your ssh repository url to .git/config file. For further information;

https://mkyong.com/git/github-keep-asking-for-username-password-when-git-push/

Initialize our branch

```
git branch deploy-vm2
git checkout deploy-vm2
git commit -m "first commit message"
git add .
git push -u origin deploy-vm2
```

```
nothing to commit, working tree clean
root@dazzle-virtual-machine:/home/dazzle/Documents/terraform# git branch
  deploy-vm
* deploy-vm2
```

Now we can create our terraform files;

We will use providers.tf to use azure provider;

```
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 3.0"
    }
  }
  required_version = ">= 1.1.3"
}
provider "azurerm" {
    features{}
}
```

Here is the variables.tf file:

```
variable "resource_group_name" {
  description = "Name of the resource group"
  default     = "test-jenkins2"
}

variable "location" {
  description = "Azure region for the resources"
  default     = "West Europe"
}

variable "admin_username" {
  description = "Admin username for the VM"
  default     = "azureuser"
}

variable "admin_password" {
  description = "Admin password for the VM"
```

```
  default     = "P@ssw0rd1234!"
}
```

Here is the outputs.tf:

```
output "public_ip_address" {
  description = "Public IP address of the virtual machine"
  value       = azurerm_public_ip.public_ip.ip_address
}
```

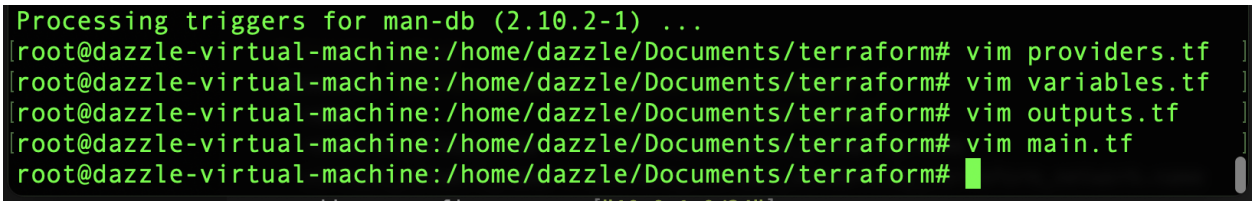
In main.tf we have several resources; resource group, virtual network, subnet, public_ip, network security group (with the security rules for allowing inbound ssh connections over port22), network interface card (with the IP configuration that uses our newly created subnet and public IP explicitly), and lastly, our virtual machine with the necessary components such as storage and virtual machines image, username and password etc.

main.tf:

```
resource "azurerm_resource_group" "rg" {
  name     = var.resource_group_name
  location = var.location
}

resource "azurerm_virtual_network" "vnet" {
  name                = "myVnet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

resource "azurerm_subnet" "subnet" {
  name                 = "mySubnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.1.0/24"]
```

```
}

resource "azurerm_public_ip" "public_ip" {
  name                = "myPublicIP"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method   = "Dynamic"
}

resource "azurerm_network_security_group" "nsg" {
  name                = "myNSG"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  security_rule {
    name                       = "SSH"
    priority                   = 1000
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "22"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }
}

resource "azurerm_network_interface" "nic" {
  name                = "myNIC"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name                          = "myNICConfig"
    subnet_id                     = azurerm_subnet.subnet.id
    private_ip_address_allocation = "Dynamic"
```

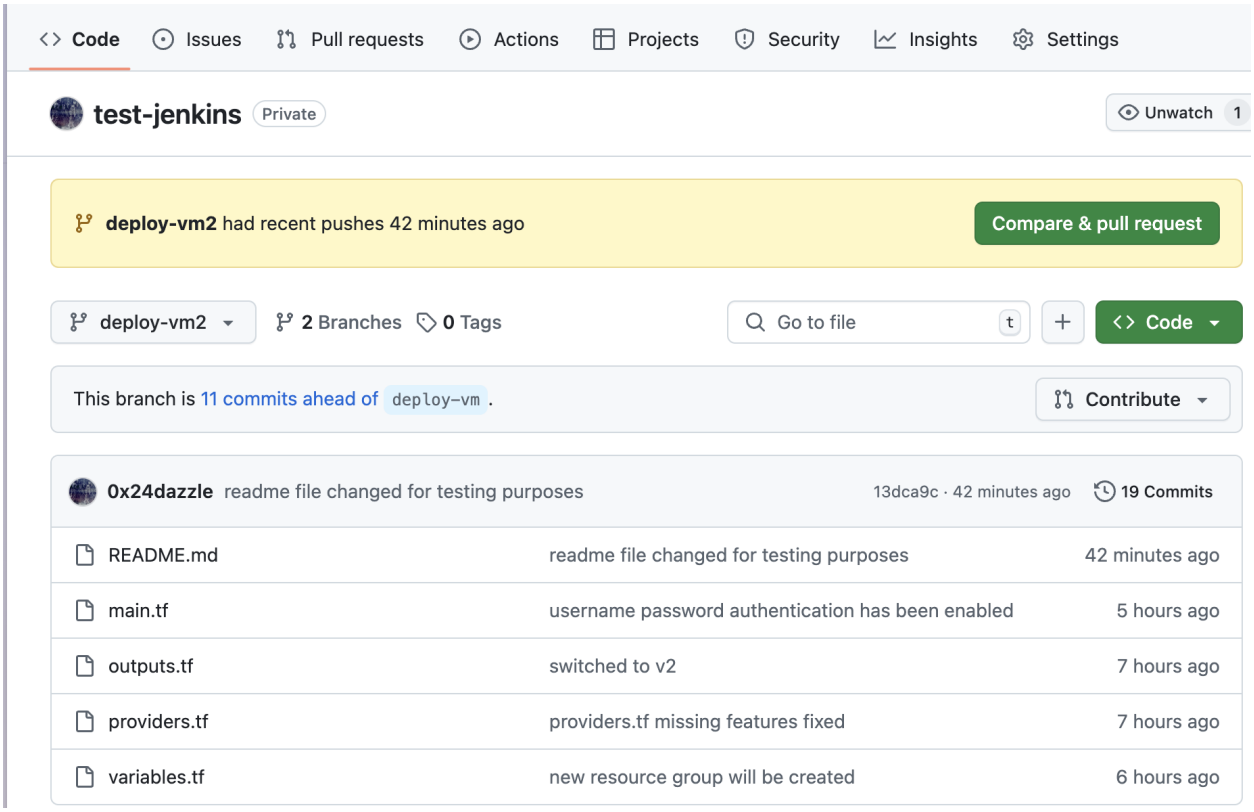
```
    public_ip_address_id          = azurerm_public_ip.public_ip
  }
}

resource "azurerm_linux_virtual_machine" "vm" {
  name                = "myVM"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  size                = "Standard_B1s"
  admin_username      = var.admin_username
  admin_password      = var.admin_password
  network_interface_ids = [
    azurerm_network_interface.nic.id,
  ]

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-jammy"
    sku       = "22_04-lts-gen2"
    version   = "latest"
  }
  disable_password_authentication = false
}
```

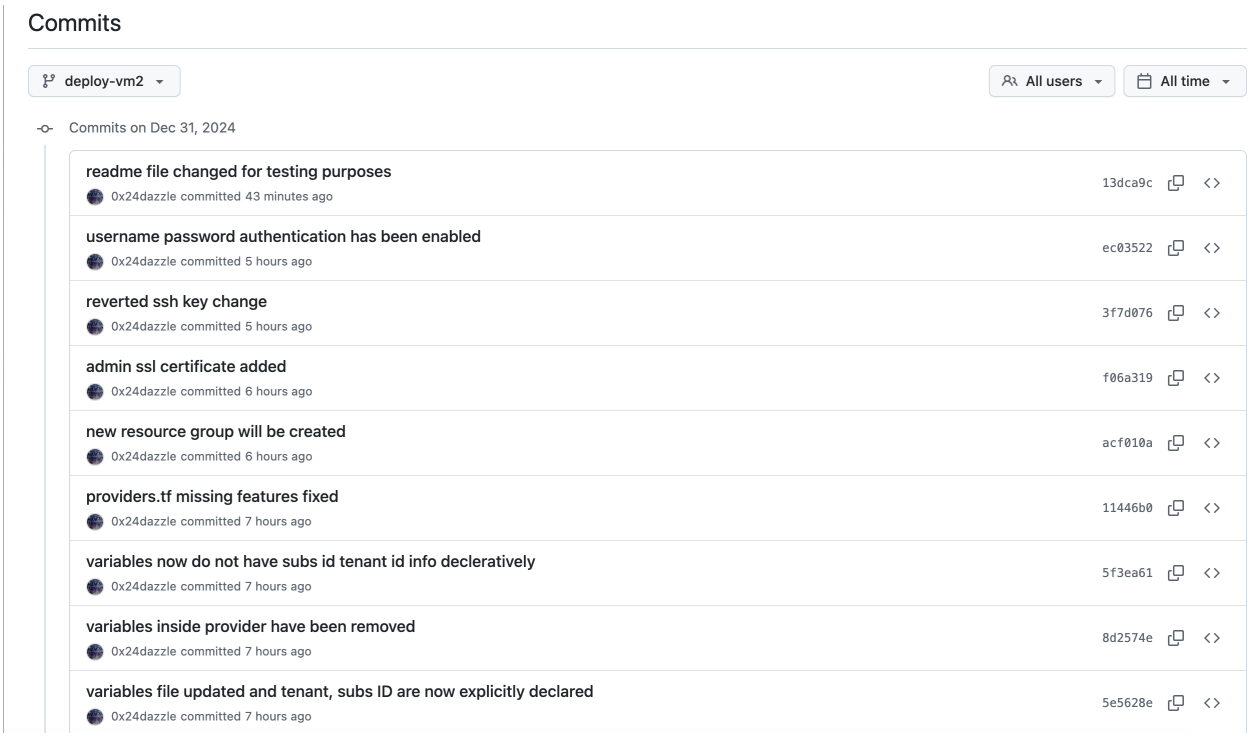


After adding our files we will push to our github repository;

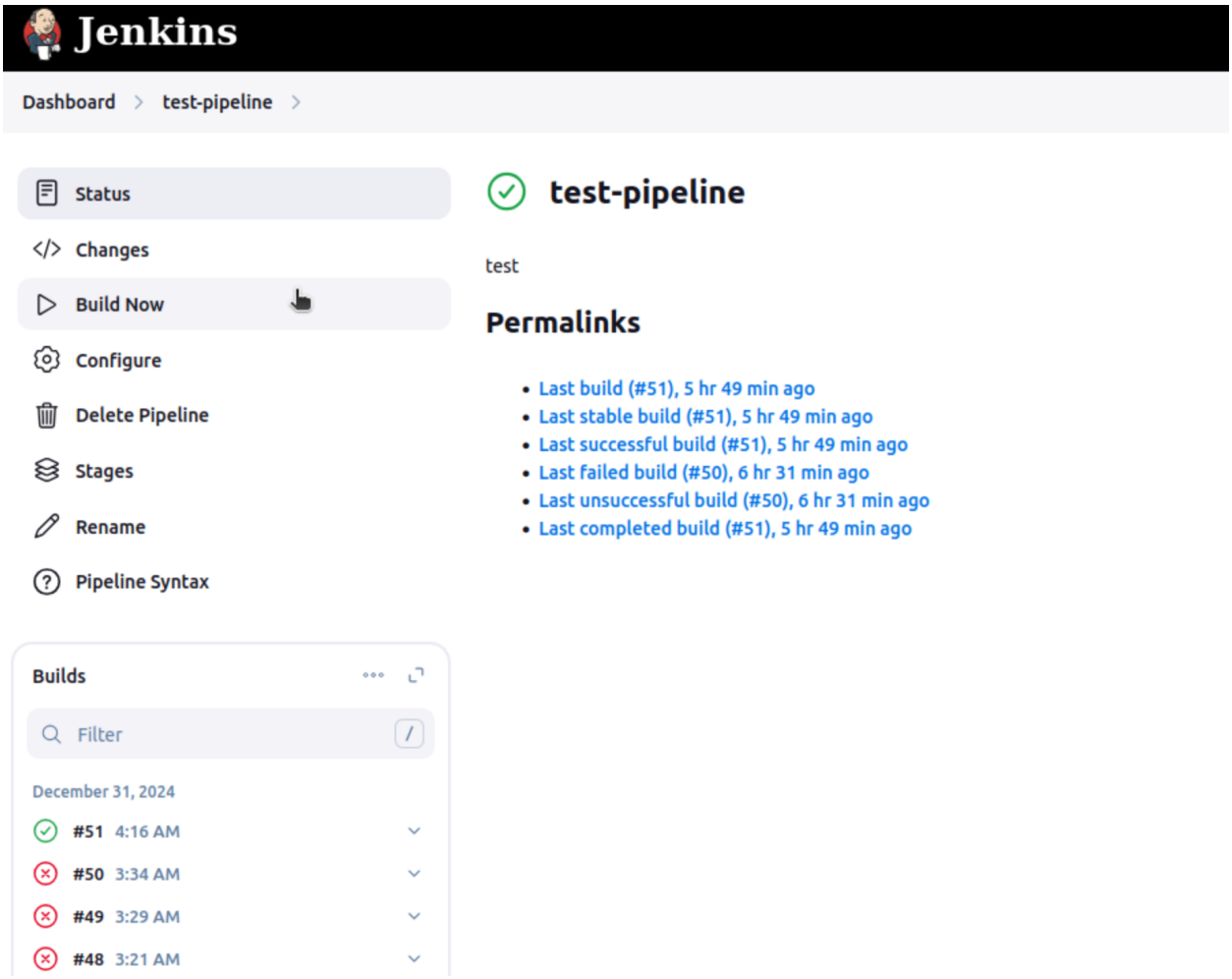
```
git branch
git add .
git commit -m "message"
git push -u origin deploy-vm2
```

Our github looks like this;

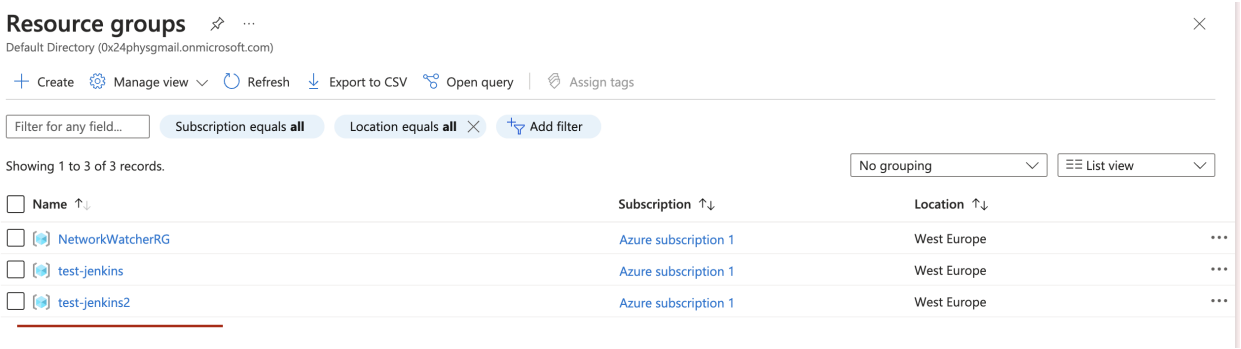


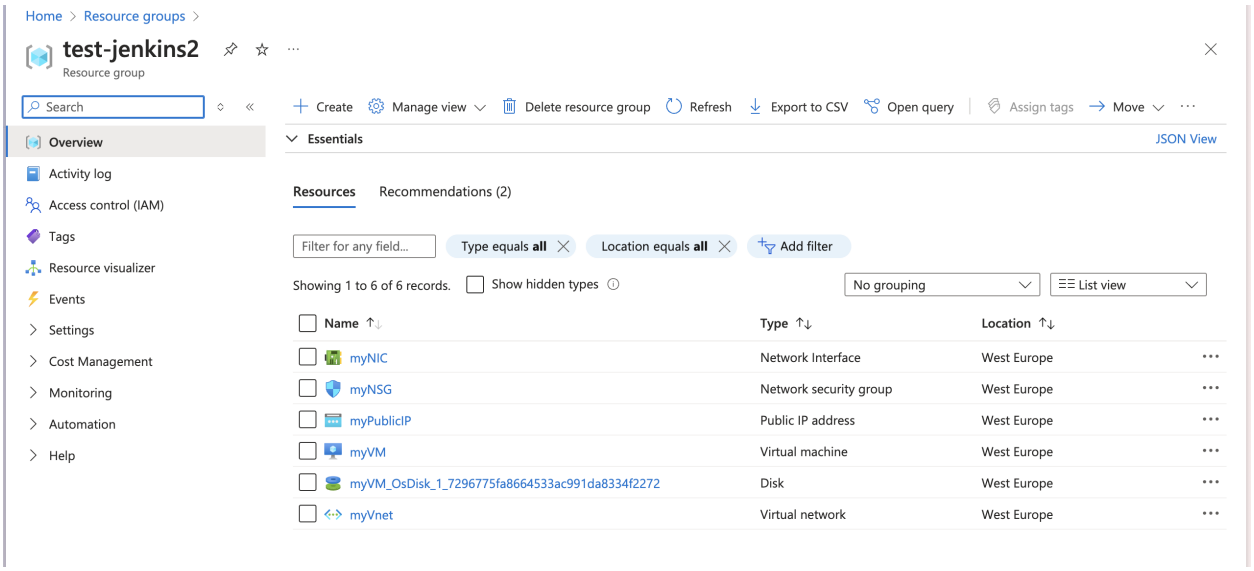


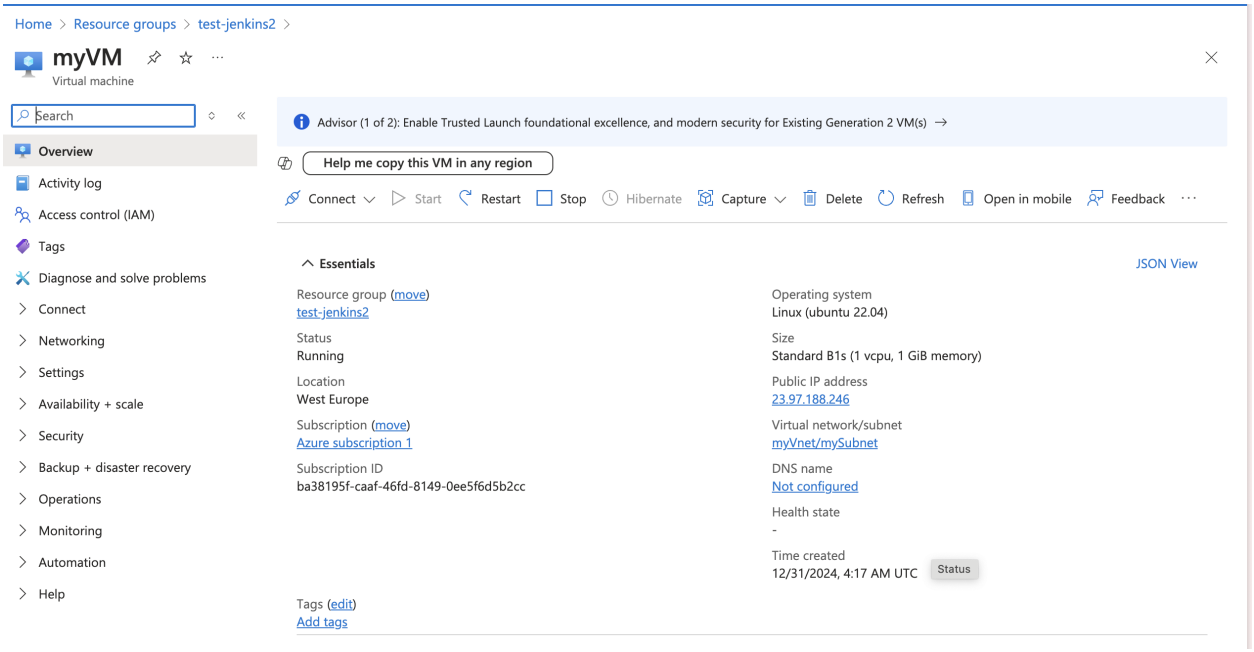
After building our pipeline, we can see that the resources are getting created on Azure;


Jenkins
Dashboard
test-pipeline
Status
Changes
Build Now
Configure
Delete Pipeline
Stages
Rename
Pipeline Syntax
Builds
Filter
December 31, 2024
#51 4:16 AM
#50 3:34 AM
#49 3:29 AM
#48 3:21 AM
test-pipeline
test
Permalinks
Last build (#51), 5 hr 49 min ago
Last stable build (#51), 5 hr 49 min ago
Last successful build (#51), 5 hr 49 min ago
Last failed build (#50), 6 hr 31 min ago
Last unsuccessful build (#50), 6 hr 31 min ago
Last completed build (#51), 5 hr 49 min ago

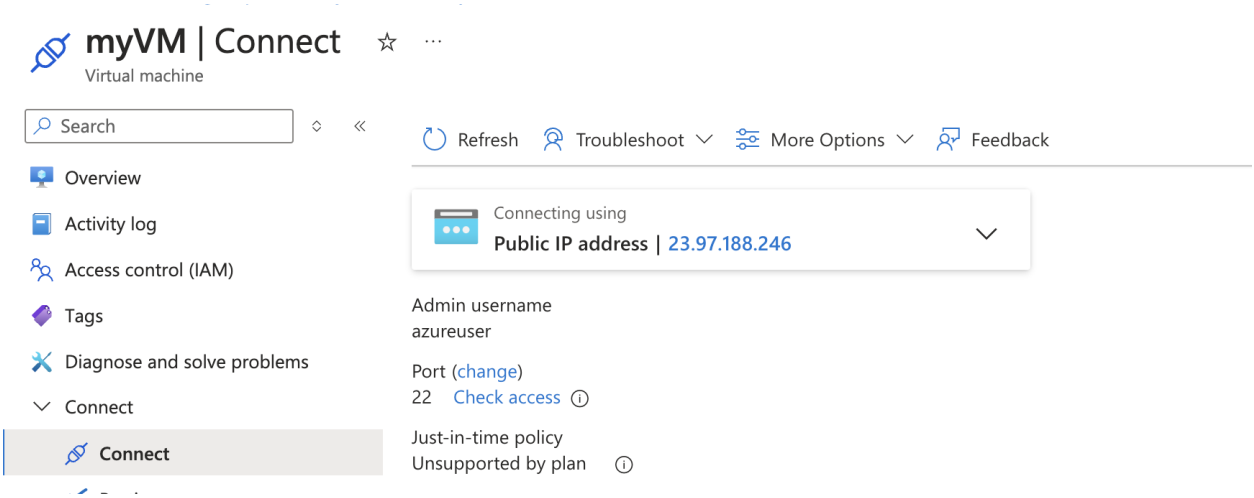
```
[0m[1mazurerm_virtual_network.vnet: Creating...[0m[0m
[0m[1mazurerm_network_security_group.nsg: Creating...[0m[0m
[0m[1mazurerm_network_security_group.nsg: Creation complete after 5s [id=/subscriptions/ba38195f-
caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins2/providers/Microsoft.Network/networkSecurityGroups/
myNSG][0m
[0m[1mazurerm_public_ip.public_ip: Creation complete after 5s [id=/subscriptions/ba38195f-
caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins2/providers/Microsoft.Network/publicIPAddresses/
myPublicIP][0m
[0m[1mazurerm_virtual_network.vnet: Creation complete after 7s [id=/subscriptions/ba38195f-
caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins2/providers/Microsoft.Network/virtualNetworks/myVnet][0m
[0m[1mazurerm_subnet.subnet: Creating...[0m[0m
[0m[1mazurerm_subnet.subnet: Creation complete after 8s [id=/subscriptions/ba38195f-caaf-46fd-8149-0ee5f6d5b2cc/
resourceGroups/test-jenkins2/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/mySubnet][0m
[0m[1mazurerm_network_interface.nic: Creating...[0m[0m
[0m[1mazurerm_network_interface.nic: Still creating... [10s elapsed][0m[0m
[0m[1mazurerm_network_interface.nic: Creation complete after 14s [id=/subscriptions/ba38195f-
caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins2/providers/Microsoft.Network/networkInterfaces/myNIC][0m
[0m[1mazurerm_linux_virtual_machine.vm: Creating...[0m[0m
[0m[1mazurerm_linux_virtual_machine.vm: Still creating... [10s elapsed][0m[0m
[0m[1mazurerm_linux_virtual_machine.vm: Still creating... [20s elapsed][0m[0m
[0m[1mazurerm_linux_virtual_machine.vm: Still creating... [30s elapsed][0m[0m
[0m[1mazurerm_linux_virtual_machine.vm: Still creating... [40s elapsed][0m[0m
[0m[1mazurerm_linux_virtual_machine.vm: Still creating... [50s elapsed][0m[0m
[0m[1mazurerm_linux_virtual_machine.vm: Creation complete after 53s [id=/subscriptions/ba38195f-
caaf-46fd-8149-0ee5f6d5b2cc/resourceGroups/test-jenkins2/providers/Microsoft.Compute/virtualMachines/myVM][0m
[0m[1m[32m
Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
[0m[0m[1m[32m
```







Let’s connect to our VM using public IP address and username password;




```
root@dazzle-virtual-machine:/home/dazzle/Documents/terraform# ssh azureuser@23.97.188.246
azureuser@23.97.188.246's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.5.0-1025-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Dec 31 10:10:52 UTC 2024

  System load:  0.03              Processes:             105
  Usage of /:   7.1% of 28.89GB   Users logged in:       0
  Memory usage: 35%               IPv4 address for eth0: 10.0.1.4
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Tue Dec 31 09:23:37 2024 from 85.22.68.226
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

azureuser@myVM:~$
```

What could be improved further;

1. Ssh keys could be added to enhance security
2. Instead of using public IP address, a bastion or a load balancer could be used

3. Tags can be used to improve resource organization and cost tracking
4. depends_on can be used to be sure to implement resources in the right order
5. Diagnostic tools can be used to troubleshoot any issues