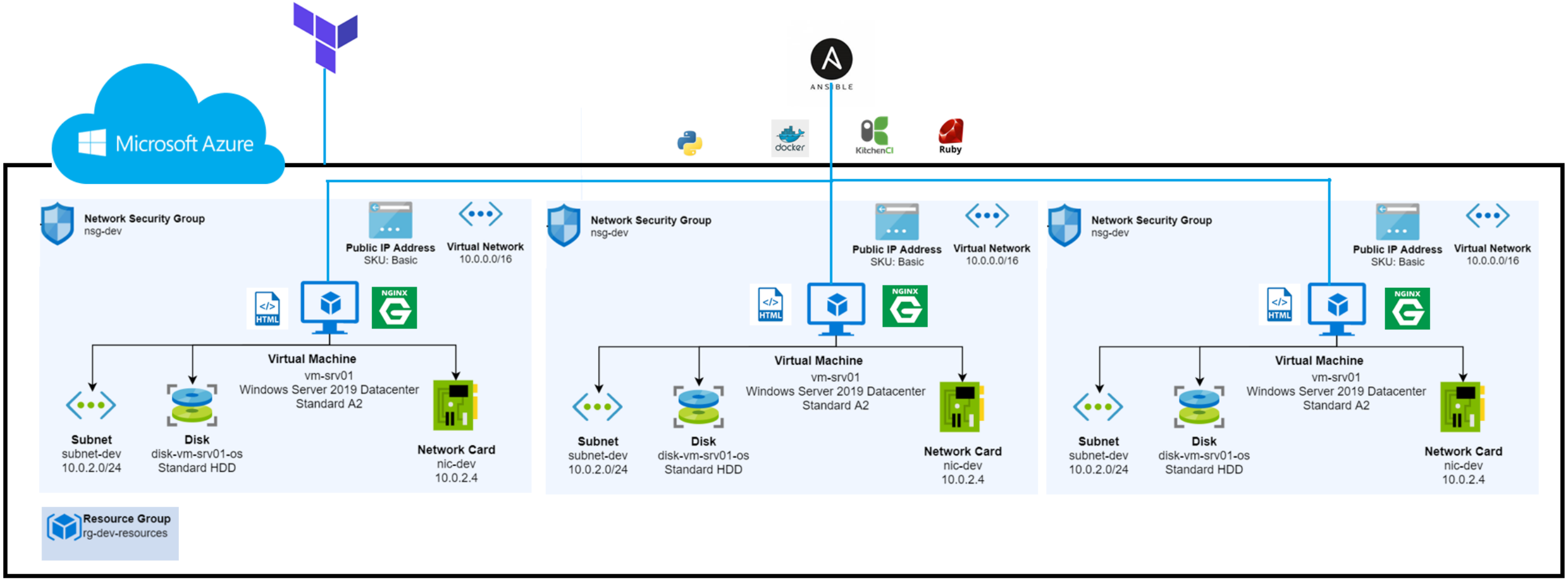
Ansible and Terraform on Microsoft Azure and AWS

- Shartul Kumar

GSK: Task 1

- Task 1: Terraform code to spin up VMs in Azure/GCP (or local virtual box)
  - All associated IaC code
  - README.md file with complete steps to replicate the setup

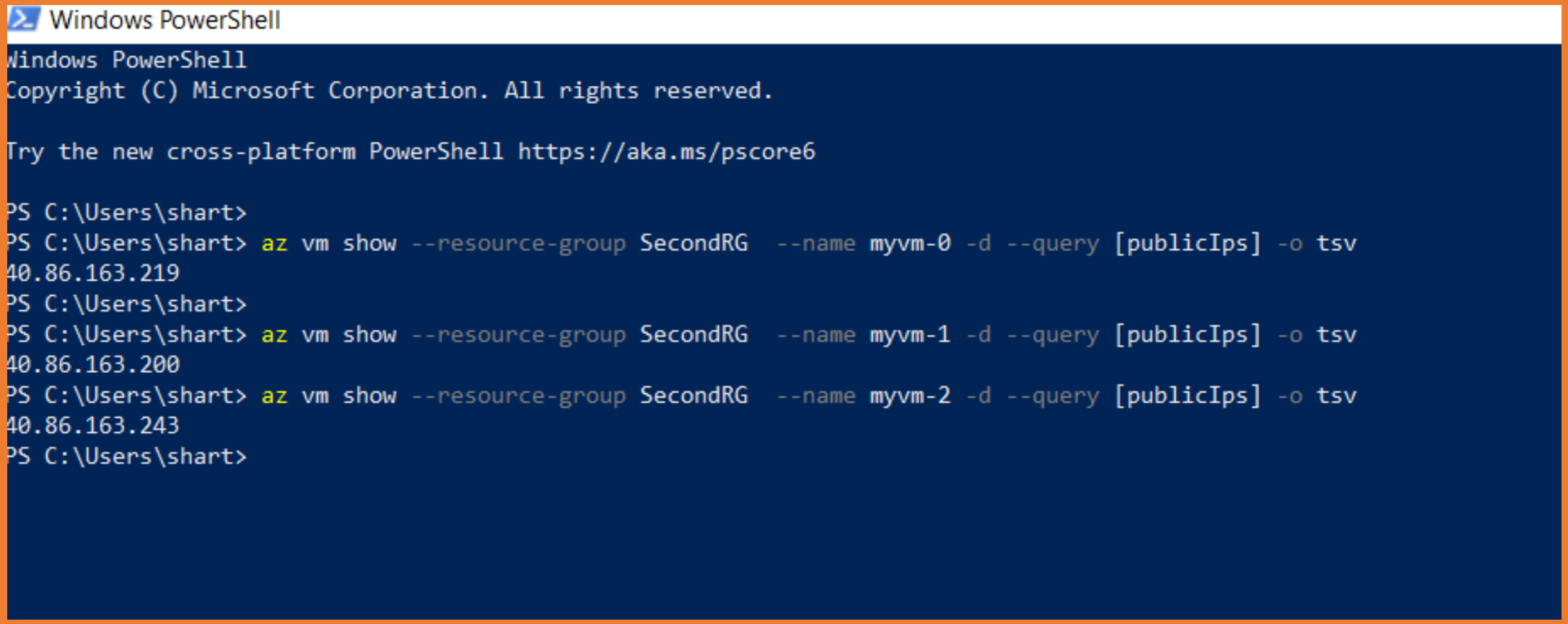
Task 1: Overall Solution Approach



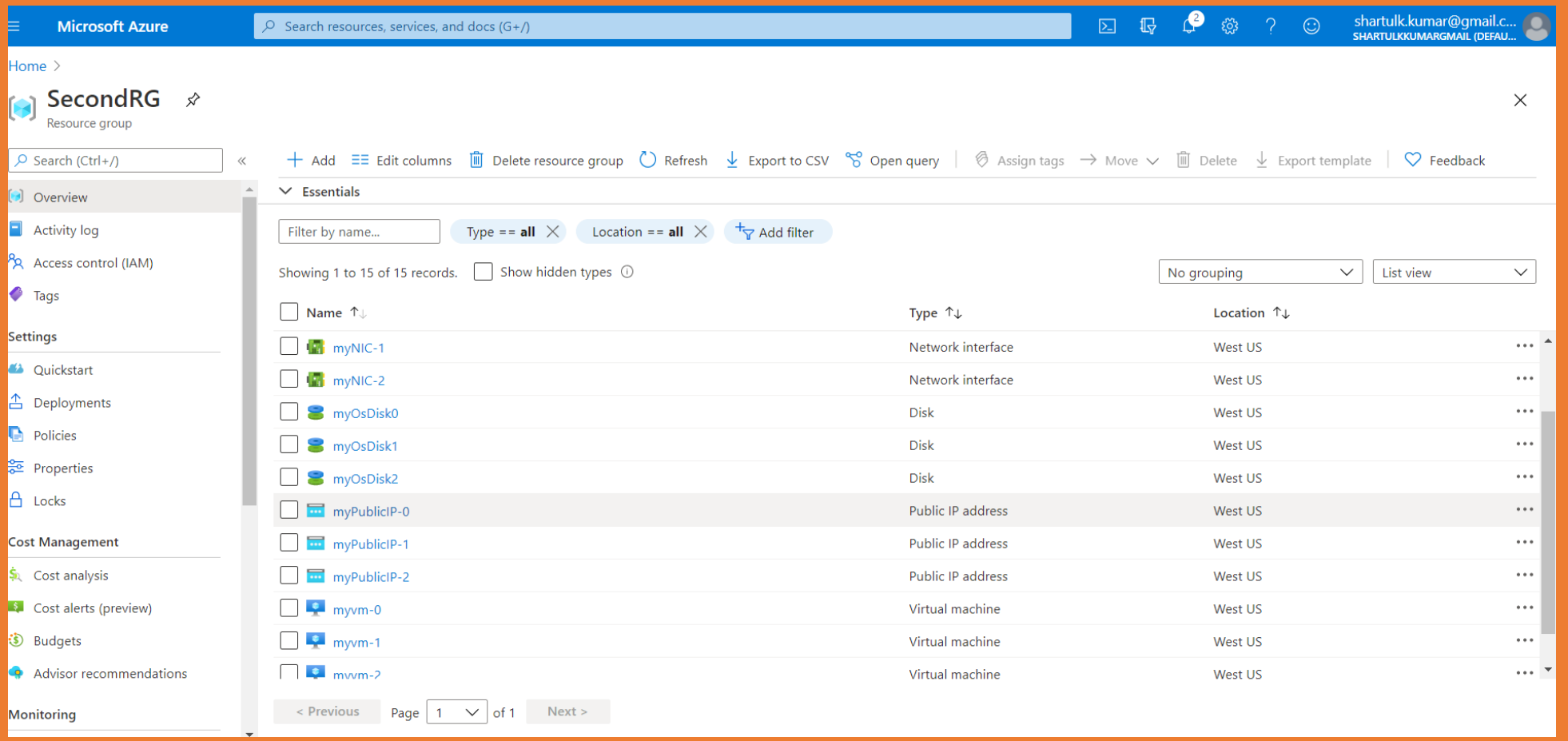
terraform: execution result

```
zurerm_network_interface_security_group_association.example[0]: Creation complete after 2s [id=/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Network/networkInte
faces/myNIC-0|/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup]
zurerm_network_interface_security_group_association.example[1]: Creation complete after 5s [id=/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Network/networkInte
faces/myNIC-1|/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup]
zurerm_network_interface_security_group_association.example[2]: Creation complete after 7s [id=/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Network/networkInte
faces/myNIC-2|/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Network/networkSecurityGroups/myNetworkSecurityGroup]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [10s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [10s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [10s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [20s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [20s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [20s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [30s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [30s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [30s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [40s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [40s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [40s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [50s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [50s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [50s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [1m0s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [1m0s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [1m0s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [1m10s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [1m10s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [1m10s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [1m20s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [1m20s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [1m20s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [1m30s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [1m30s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [1m30s elapsed]
zurerm_linux_virtual_machine.myterraformvm[0]: Still creating... [1m40s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Still creating... [1m40s elapsed]
zurerm_linux_virtual_machine.myterraformvm[2]: Still creating... [1m40s elapsed]
zurerm_linux_virtual_machine.myterraformvm[1]: Creation complete after 1m45s [id=/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Compute/virtualMachines/myvm-1]
zurerm_linux_virtual_machine.myterraformvm[0]: Creation complete after 1m47s [id=/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Compute/virtualMachines/myvm-0]
zurerm_linux_virtual_machine.myterraformvm[2]: Creation complete after 1m49s [id=/subscriptions/4b787b90-ec08-4b9c-8dbf-54dd5e97cbd0/resourceGroups/SecondRG/providers/Microsoft.Compute/virtualMachines/myvm-2]

pply complete! Resources: 18 added, 0 changed, 0 destroyed.

utputs:

-address-hostname = [
 "myvm-0:",
 "myvm-1:",
 "myvm-2:",

hartul@Shartul-O14FB8K:/mnt/d/terraform-labs/terraform$
```

terraform: execution result
3 VMs public IP address



terraform: execution result on Azure

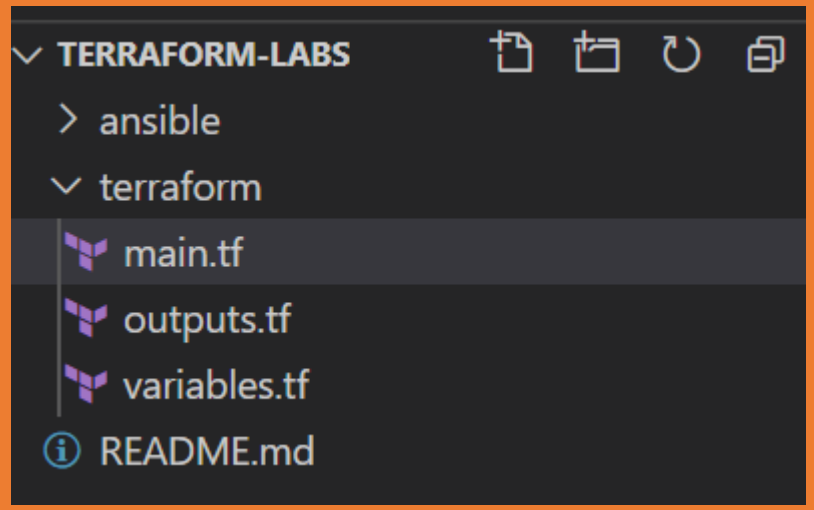


terraform: IaC Code

- Project Structure

  terraform-labs

  -terraform



terraform: IaC Code

- Variables.tf

- Defines all variables:

resource_group_name, location, tags
admin_username,ssh_key
storage_account_type
vm_hostname, vm_size, vm_os_offer,
vm_os_publisher, vm_os_sku,
vm_os_version

variables.tf

terraform > variables.tf

```
variable "resource_group_name" {
  description = "The name of the resource group in which the resources will b
  default     =  "SecondRG"
}


variable "tags" {
    default     = {
        env     = "Terraform azure"
    }
}

variable "location" {
    description = "Default Azure region"
    default     =   "West US"
}

variable "node_count" {
  description = "Specify the number of vm instances"
  default     = "3"
}

variable "admin_username" {
  description = "The admin username of the VM that will be deployed"
  default     = "azureuser"
}


variable "ssh_key" {
```

terraform: IaC Code

- Main.tf

- Defines all :

Provider, azure resource group,

Virtual network, subnet,

azure public ip, NSG, NIC,

Connect the security group to the

network interface,

storage account,

Azure VMs – 3 instances with cou

os_disk, image reference, admin s

Boot diagnostic

main.tf

terraform > main.tf

```
65  # **********************
66  # NIC Resource Creation
67  # **********************
68
69  resource "azurerm_network_interface" "myterraformnic" {
70      count                     = 3
71      name                      = "myNIC-${count.index}"
72      location                  = "${var.location}"
73      resource_group_name       = "${azurerm_resource_group.myterraformgroup.name}"
74
75      ip_configuration {
76          name                          = "myNicConfiguration-${count.index}"
77          subnet_id                     = "${azurerm_subnet.myterraformsubnet.id}"
78          private_ip_address_allocation = "Dynamic"
79          # public_ip_address_id          = azurerm_public_ip.myterraformpublicip.id
80          public_ip_address_id = "${length(azurerm_public_ip.myterraformpublicip.*.id) > 0 ? element(concat(azurerm_public_ip
81      }
82      tags = "${var.tags}"
83  }
84
85
86  # Connect the security group to the network interface
87  resource "azurerm_network_interface_security_group_association" "example" {
88      count                     = 3
89      network_interface_id      = azurerm_network_interface.myterraformnic[count.index].id
90      network_security_group_id = azurerm_network_security_group.myterraformnsg.id
91  }
```

terraform: IaC Code

- Outputs.tf

- Defines outputs:

Public IP address of 3 Azure VMs

outputs.tf

terraform > outputs.tf

```
output "ip-address-hostname" {
  value = "${
      formatlist(
        "%s:%s",
        azurerm_linux_virtual_machine.myterraformvm.*.name, azurerm_public_ip.myterraformpublicip.*.ip_address
      )
    }"
}

#  alternatively, Use  thefolllwing command from PowerShell to get the ipaddress
# az vm show --resource-group SecondRG  --name myvm-0 -d --query [publicIps] -o tsv
# az vm show --resource-group SecondRG  --name myvm-1 -d --query [publicIps] -o tsv
# az vm show --resource-group SecondRG  --name myvm-3 -d --query [publicIps] -o tsv
# output is ipaddres:

```

Task 1: README file

- https://github.com/kshartul/terraform-labs/blob/main/README.md

GSK: Task 2

- Task 2: Infrastructure setup in Azure/AWS/GCP (or local virtual box)

 - All VMs built using - image - ubuntu-xenial-16.04-amd64-server

- All VMs allow the users in the admin group (create test user1 and user2), to sudo without a password

- Webservers and load balancer running nginx

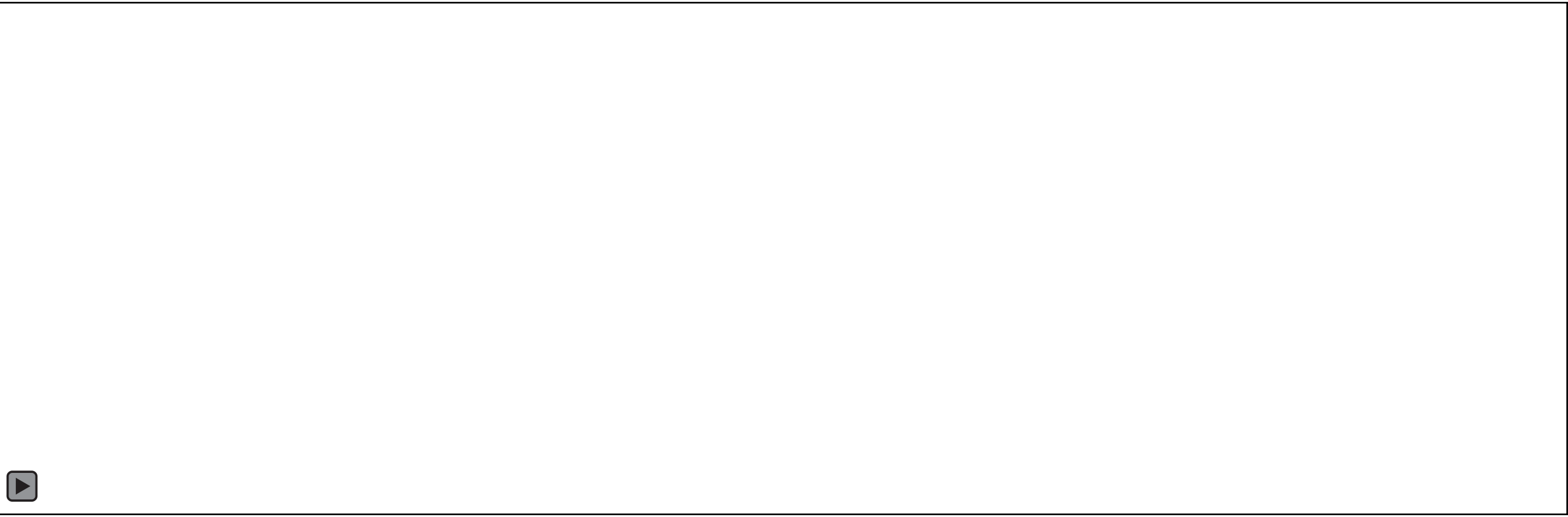
- Simple ‘Hello World’ application deployed to both webservers

- Automated tests to show that app is deployed correctly and nginx is load balancing correctly

Infrastructure as code !

Cloud is software defined ! And code defines it

IaC: Hover to find play video icon on left

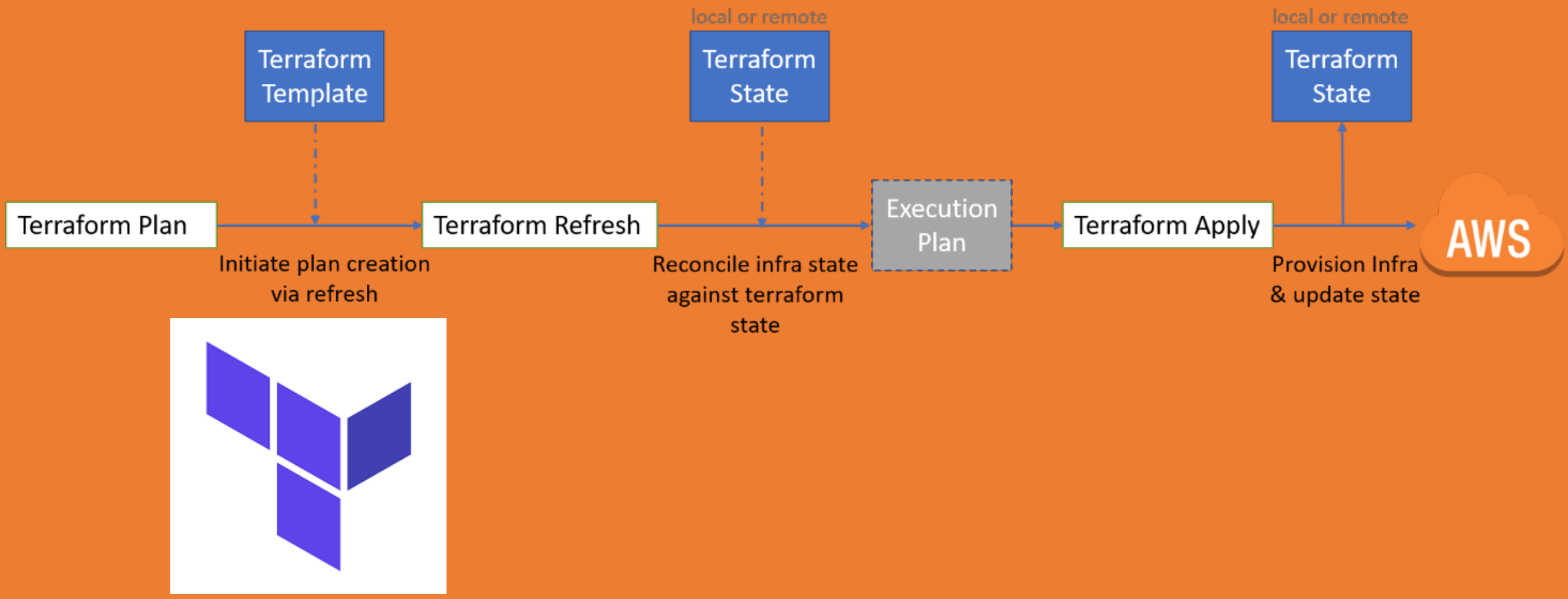


Why Terraform

- Terraform is a product to provision infrastructure and application resources across private cloud, public cloud, and external services using a common workflow
- Multi cloud
- Easy to describe json like format call HCF
- Supports for both on-prem and clouds

Provision Any Infrastructure For Any Application

Terraform: Declarative; Known Vs Actual state



Terraform: OPA policy on the Terraform plan

- OPA : Open Policy Agent Policies that test the changes Terraform is about to make before it makes them.

Save your terraform plan to binary and then to json format

```
terraform plan --out tfplan.binary
tfjson tfplan.binary > tfplan.json
```

The policy authorizes the plan when the score for the plan is below a threshold and there are no changes made to any IAM resources.

-Draft a “terraform.rego” file with reference https://www.openpolicyagent.org/docs/v0.11.0/terraform/

-evaluate the policy against that plan, you hand OPA the policy from terminal

```
opa eval --data terraform.rego --input tfplan.json "data.terraform.analysis.authz“
opa eval --data terraform.rego --input tfplan.json "data.terraform.analysis.score“ check given score
```

- Or Run OPA as a daemon and then evaluate policy ; start daemon opa run -s terraform.rego

Then from a terminal, use OPA’s HTTP API to evaluate the policy against the Terraform plans

```
curl localhost:8181/v0/data/terraform/analysis/authz -d @tfplan.json
```

Cloud deployment patterns

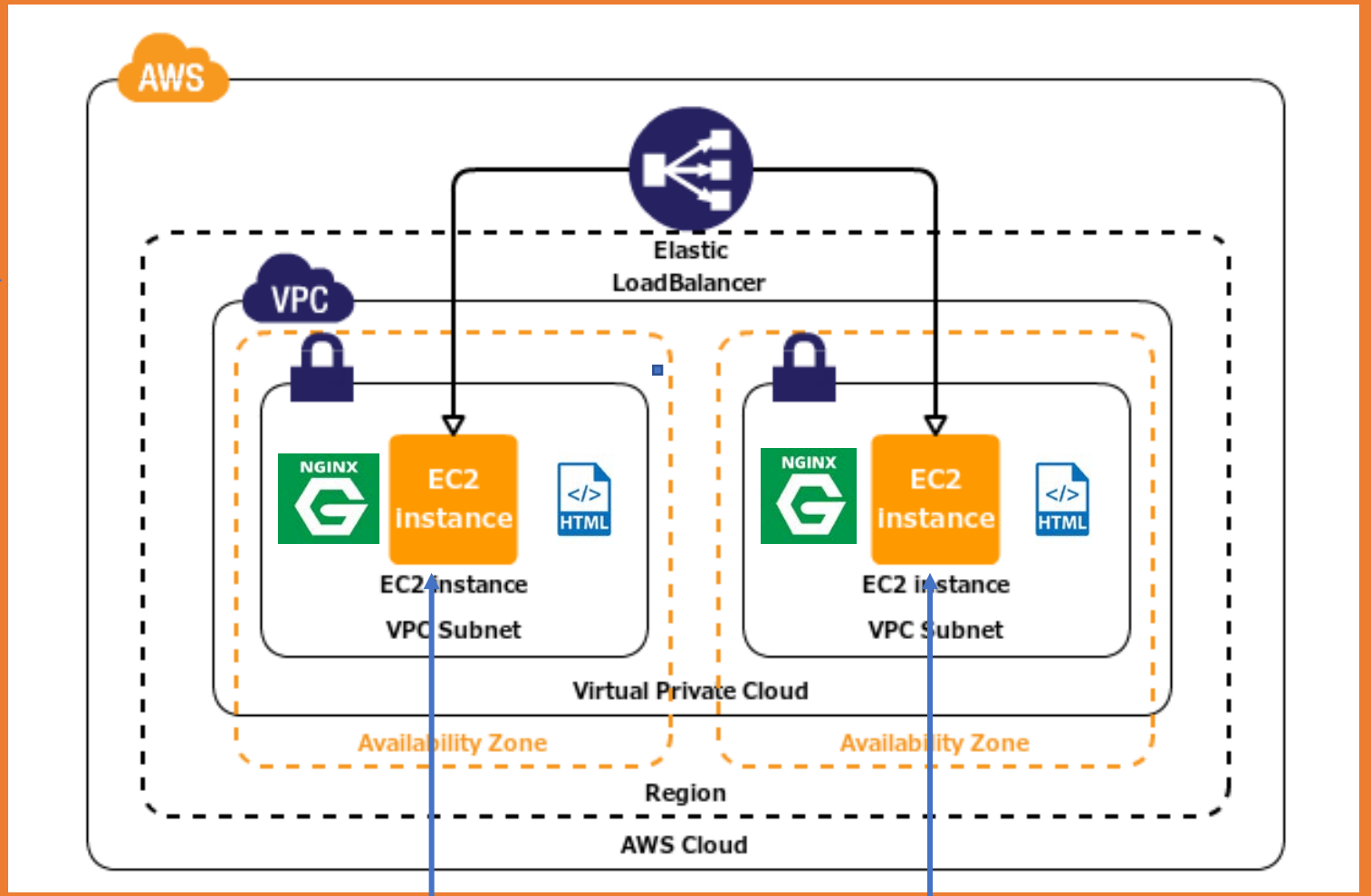
Mutable Infrastructure

- Capable of change
- Install the infrastructure and all updates are in place
- What you and most of us are used to
- Legacy software will be managed by this pattern

Immutable Infrastructure

- Not capable or susceptible of change
- Deploy infrastructure as per specification , change is a new infra deployment
- Newer pattern, cloud native
- Works best with new application patterns

Task 2: Overall Solution Approach



ANSIBLE

docker

KitchenCI

Ruby

Project Setup:

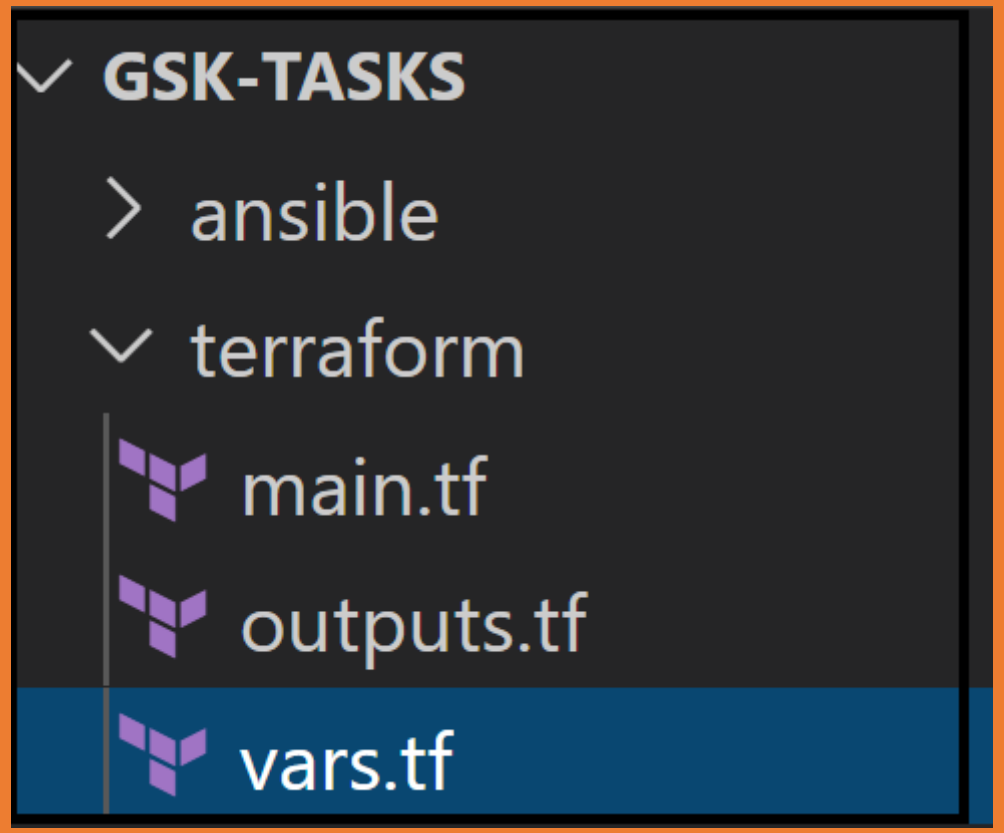
Folder structure

gsk-tasks

- - terraform
- - ansible

terraform:

- Main.tf
- Output.tf
- Vars.tf



terraform:

- Vars.tf
  - Defines all variables:
    aws region
    aws credentials
    key-name
    ssh-key path

vars.tf

terraform > vars.tf

```
1   variable "region"{
2     description = "AWS region"
3     default     = "us-east-1"
4   }
5
6   variable "shared_cred_file"{
7     default = "~/.aws/credentials"
8   }
9
10
11  variable "key_name"{
12    description = "Key name for SSHing into EC2"
13    default     = "deployer-key"
14  }
15
16  variable "ssh_key"{
17    description = "Path to the public key to be used for ssh access to the VM."
18    default = "~/.ssh/id_rsa.pub"
19  }
```

terraform:

- outputs.tf
  - Defines all outputs:

    ec2 pubic ip address

    elb dns name

outputs.tf

terraform > outputs.tf

```
1   output "instance_id1"{
2     value = ["${aws_instance.gsk-one.*.public_ip}"]
3
4   }
5
6   output "instance_id2"{
7   value = ["${aws_instance.gsk-two.*.public_ip}"]
8   }
9
10  output "elb_dns_name"{
11    value = "${aws_elb.example.dns_name}"
12  }
```

terraform:

- main.tf
  - Defines all :

  key-pair

  region

  aws ami

  -ubuntu 16.04 xenial-amd64

main.tf

```
provider "aws" {
 region = "${var.region}"
 shared_credentials_file = "${var.shared_cred_file}"
 profile = "default"
}
resource "aws_key_pair" "deployer"{
  key_name   = "deployer-key"
  public_key = "${file("${var.ssh_key}")}" # Please add the public key of th
 }

data "aws_ami" "ubuntu"{
   most_recent = true

   owners = ["099720109477"]

    filter {
        name   = "name"
        values = ["ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-*"
    }

    filter {
        name   = "virtualization-type"
```

terraform:

- main.tf
  - Defines all :
  2 ec2 instances

main.tf

```
# Creating EC2 instance
resource "aws_instance" "gsk-one"{
  ami                    = "${data.aws_ami.ubuntu.id}"
  key_name               = "${var.key_name}"
  vpc_security_group_ids = ["${aws_security_group.instance.id}"]
  source_dest_check      = false
  instance_type          = "t2.micro"

  tags = {
    Name = "gsk-testing"
  }
}

resource "aws_instance" "gsk-two"{
  ami                    = "${data.aws_ami.ubuntu.id}"
  key_name               = "${var.key_name}"
  vpc_security_group_ids = ["${aws_security_group.instance.id}"]
  source_dest_check      = false
  instance_type          = "t2.micro"
```

terraform:

- main.tf
  - Defines all :

  Network Security Group;

  ingress/outgress

  Virtual Private Network

main.tf

```
# Creating Security Group for EC2
resource "aws_security_group" "instance"{
  name = "terraform-gsk"

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
```

terraform:

- main.tf

  - Defines all :

  Elastic Load Balancer;

  health check, listener

main.tf

```
}

# Creating ELB
resource "aws_elb" "example"{
  name               = "terraform-elb-gsk"
  security_groups    = ["${aws_security_group.elb.id}"]
  availability_zones = "${data.aws_availability_zones.all.names}"

  health_check {
    healthy_threshold   = 2
    unhealthy_threshold = 2
    timeout             = 3
    interval            = 30
    target              = "TCP:22"
  }

  listener {
    lb_port           = 80
    lb_protocol       = "http"
    instance_port     = "80"
    instance_protocol = "http"
  }
```

terraform: execution result

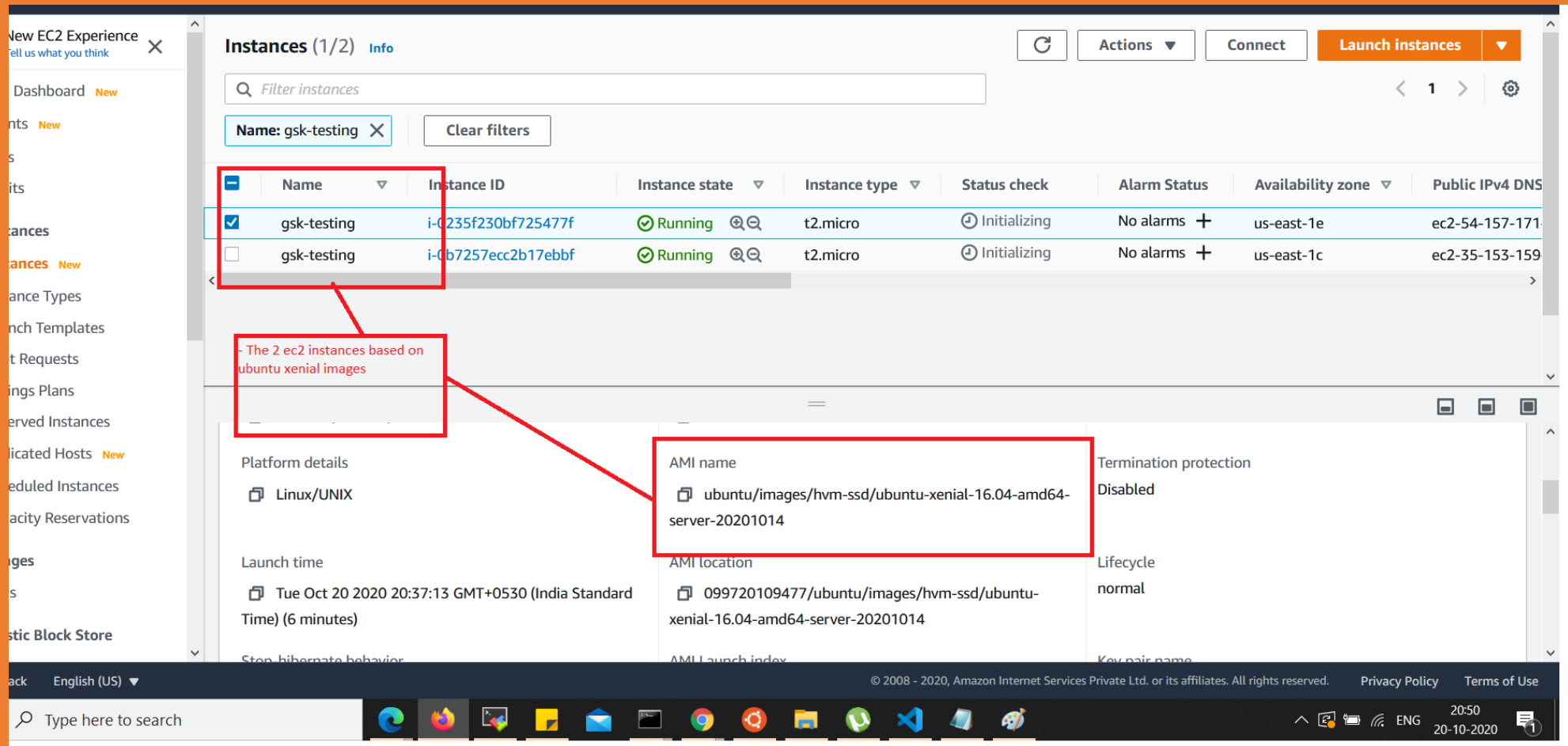
```
aws_instance.gsk-two: Creation complete after 35s [id=i-02551250b1725477f]
aws_instance.gsk-one: Creation complete after 36s [id=i-0b7257ecc2b17ebbf]
aws_security_group.elb: Still destroying... [id=sg-0fc6defd951f8f63a, 50s elapsed]
aws_security_group.elb: Still destroying... [id=sg-0fc6defd951f8f63a, 1m0s elapsed]

aws_security_group.elb: Still destroying... [id=sg-0fc6defd951f8f63a, 1m10s elapsed]
aws_security_group.elb: Still destroying... [id=sg-0fc6defd951f8f63a, 1m20s elapsed]
aws_security_group.elb: Destruction complete after 1m23s
aws_security_group.elb: Creating...
aws_security_group.elb: Creation complete after 10s [id=sg-012b4178c6b3bdffa]
aws_elb.example: Creating...
aws_elb.example: Still creating... [10s elapsed]
aws_elb.example: Creation complete after 19s [id=terraform-elb-gsk]
aws_elb_attachment.two: Creating...
aws_elb_attachment.one: Creating...
aws_elb_attachment.one: Creation complete after 1s [id=terraform-elb-gsk-20201020150857180100000001]
aws_elb_attachment.two: Creation complete after 1s [id=terraform-elb-gsk-20201020150857188200000002]

Apply complete! Resources: 8 added, 0 changed, 2 destroyed.

Outputs:

elb_dns_name = terraform-elb-gsk-1985368309.us-east-1.elb.amazonaws.com
instance_id1 = [
  [
    "35.153.159.204",
  ],
]
instance_id2 = [
  [
    "54.157.171.42",
  ],
]
shartul@Shartul-O14EB8K:/mnt/d/gsk-task/terraform$
shartul@Shartul-O14EB8K:/mnt/d/gsk-task/terraform$
```

- The classic load balancer which manages the traffic between the webserver running on these two ec2 intances

- The public ip address of 2 ec2 instances using ubuntu16.04-xenial-amd64-server

- The terraform output gives the result here in the console, can also be saved toa file with the -out flag

terraform: execution result



What is Ansible?

Modular

Many built-in modules, or you can write your own

Agent-less

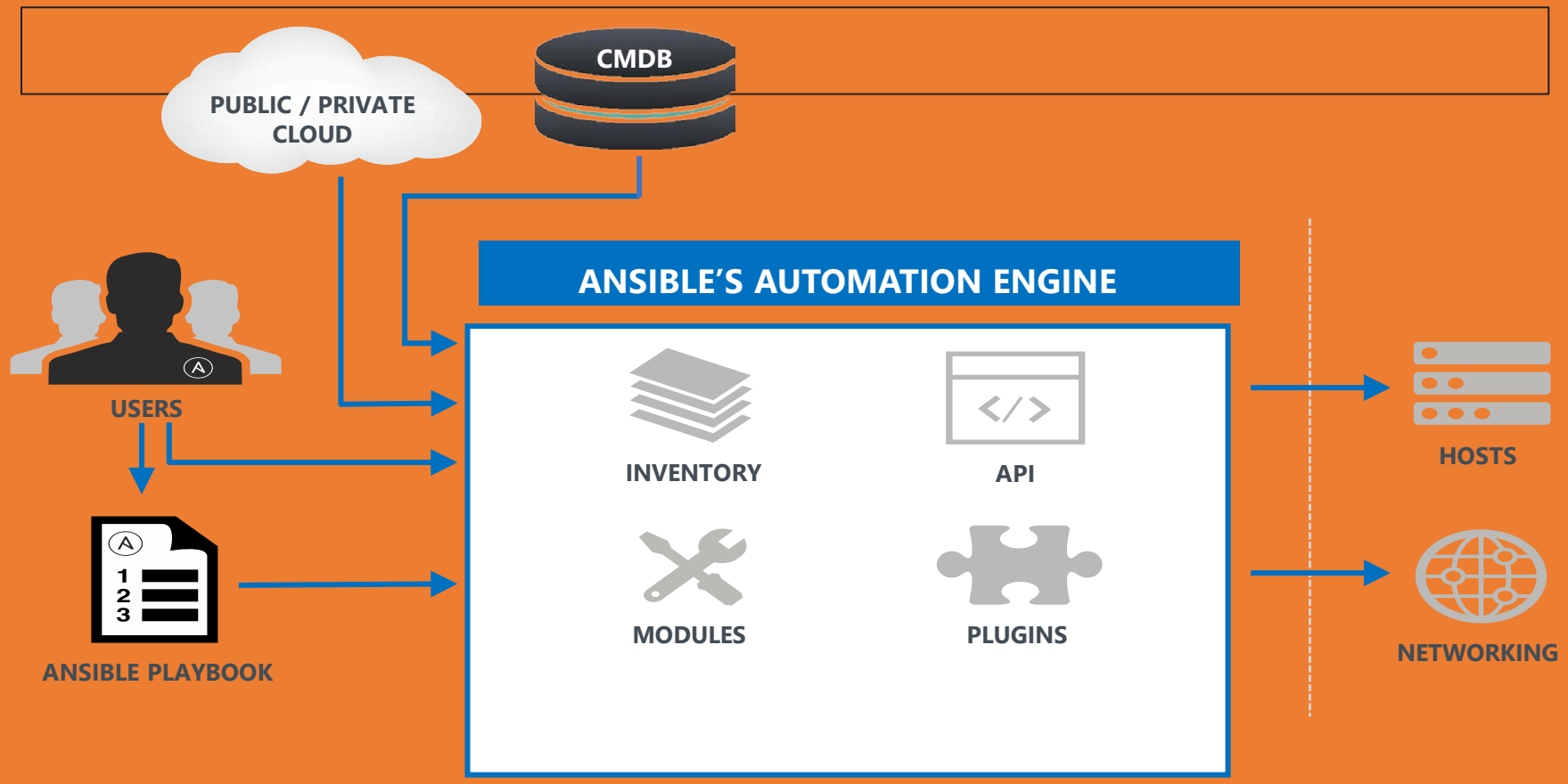
Your Ansible controller will connect to hosts to run

SSH-based

Connect to your hosts with SSH

Keys (recommended), passwords, or Kerberos (Windows is supported)

How Ansible Works?



ansible:

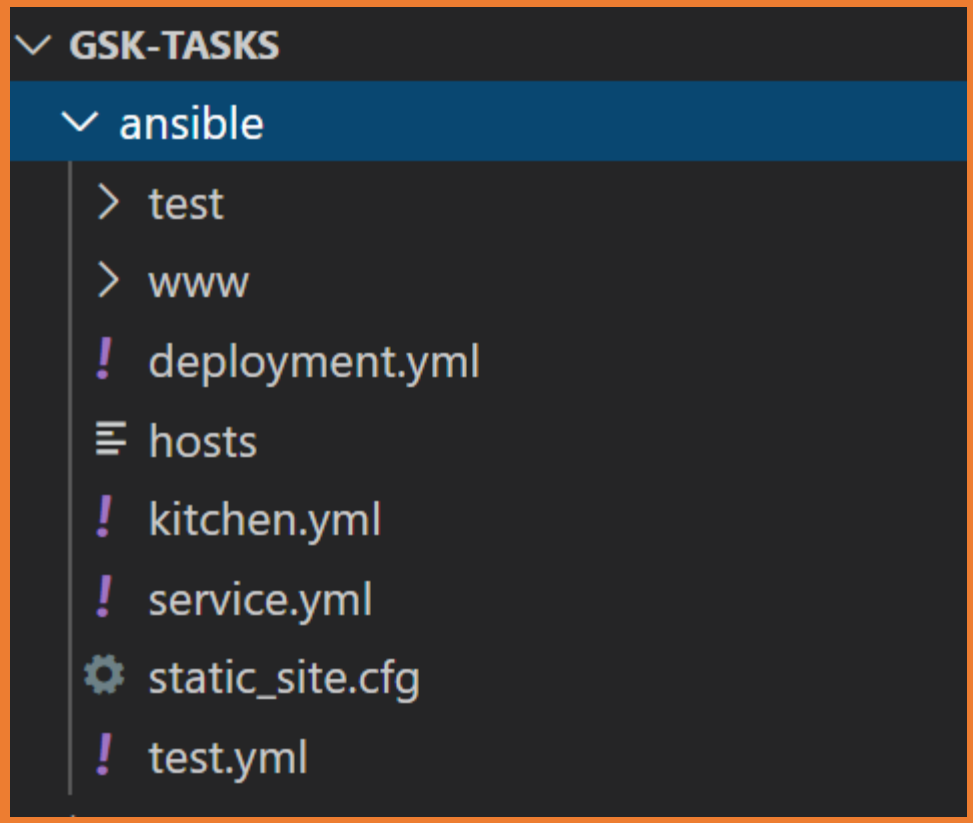
Folders:

-test

-www

Files:

- Host
- Deployment.yml
- Service.yml
- Static_site.cfg
- Test.yml



ansible:

File:

- Host

2 ec2 ipaddress

variable

ubuntu user

public key; key-pair

python3 interpreter

hosts

ansible > hosts

```
1   [servers]
2   35.153.159.204
3   54.157.171.42
4
5
6   [servers:vars]
7   ansible_user=ubuntu
8   ansible_ssh_private_key_file= ~/.ssh/id_rsa
9   ansible_python_interpreter=/usr/bin/python3
10
```

ansible:

Files:

- Deployment.yml

-Upgrade and Install packages
docker, python
-Pull nginx docker image
-From www to var
Copy ansible logo and index file
-Build the container from nginx image
-Run the container
-Expose the container to 80

deployment.yml

AnsibleMetrics

ansible > deployment.yml > ...

```yaml
---
- name: Install docker, pull nginx image, run nginx container with sample c
  hosts: servers
  remote_user: ubuntu
  become: yes
  vars:
    ansible_python_interpreter: /usr/bin/python3
    container_name: nginx
    container_image: nginx
    container_image_tag: latest
  gather_facts: no
  pre_tasks:
    - name: 'install python'
      raw: 'sudo apt-get -y update && sudo apt-get -y upgrade && sudo apt-g
  tasks:
    - name: Install aptitude using apt
      apt: name=aptitude state=latest update_cache=yes force_apt_get=yes

    - name: Install required system packages
      apt: name={{ item }} state=latest update_cache=yes
```

```yaml
- name: Add Docker GPG apt Key
  apt_key:
    url: https://download.docker.com/linux/ubu
    state: present

- name: Add Docker Repository
  apt_repository:
    repo: deb https://download.docker.com/linu
    state: present

- name: Update apt and install docker-ce
  apt: update_cache=yes name=docker-ce state=

- name: Install Docker Module for Python
  pip:
    name: docker

- name: Pull default Docker image
  docker_image:
    source: pull
    name: "{{ container_image }}"
    tag: "{{ container_image_tag }}"
```

```yaml
- name: Copy content directory for webserver
  copy:
    src: ./www
    dest: /var
    mode: '0755'
    owner: www-data
    group: www-data

- name: Start Nginx container
  docker_container:
    name: "{{ container_name }}"
    image: "{{ container_image }}"
    volumes:
      - "/var/www:/usr/share/nginx/html:ro"
    published_ports:
      - "80:80"
    state: started
```

ansible:

Files:

- Static_site.cfg

nginx configuration file

- Test.yml

rest api automated test with ansible

to verify the ansible logo

static_site.cfg

ansible > static_site.cfg

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /home/foo/static-site;
        server_name _;
        location / {
                try_files $uri $uri/ =404;
        }
}
```

test.yml

ansible > test.yml > ...

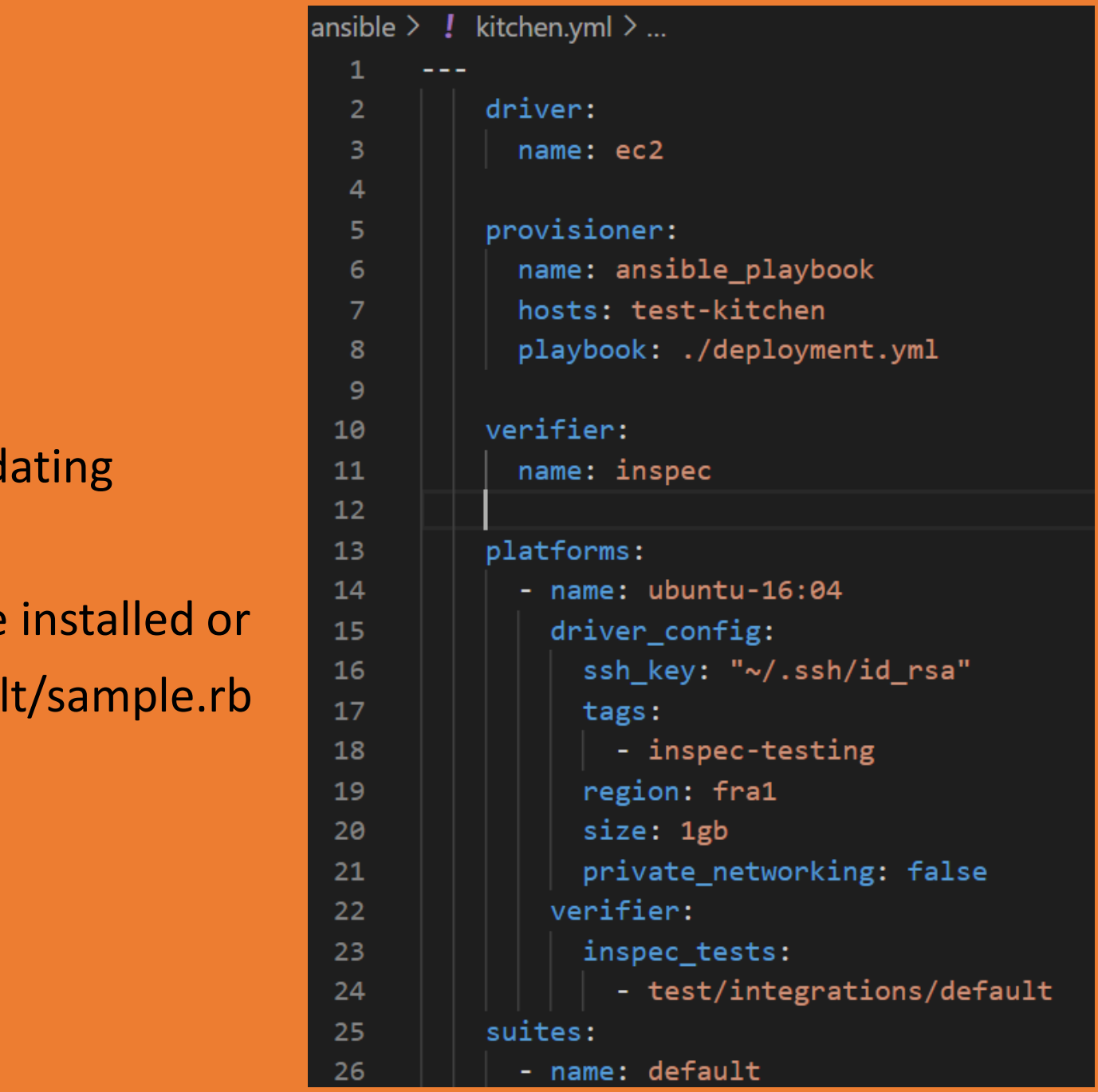
```
---
- name: validate ec2 instance-01 deployment
  uri:
    url: http://35.153.159.204:8080/
    method: POST
    body: "{{ lookup('file','ansible-logo.jpg') }}"
    body_format: json
    headers:
    Content-Type: "application/json"
- name: validate ec2 instance-01 deployment
  uri:
    url: http://54.157.171.42:8080/
    method: POST
    body: "{{ lookup('file','ansible-logo.jpg') }}"
    body_format: json
    headers:
    Content-Type: "application/json"
```

ansible:

Files:

- Kitchen.yml

  - Test kitchen setup for validating ansible deployment
  - Validation of nginx package installed or not test/integration/default/sample.rb file



ansible:

Folders:

-test

kitchen test;

to validate nginx package

in ruby file

-www

ansible logo in jpg

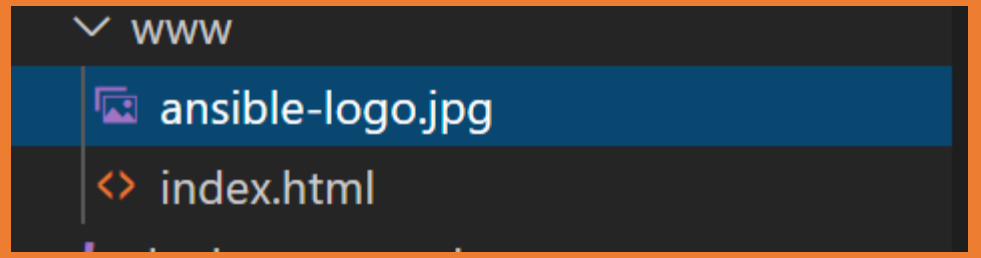
index.html

hello world app

sample.rb

ansible > test > integrations > default > sample.rb

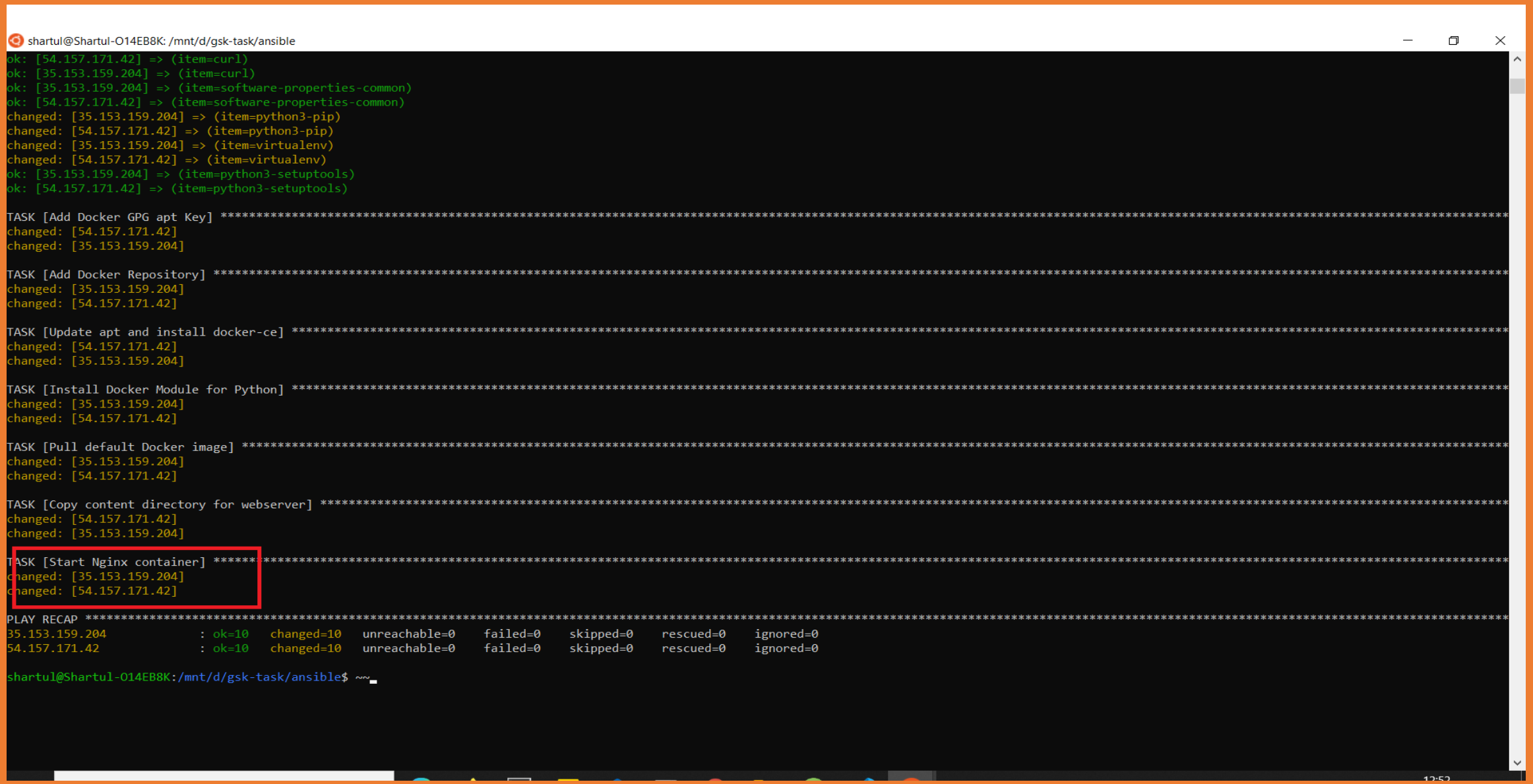
```ruby
describe package('nginx') do
  it { should be_installed }
end

```



ansible: execution result Part 1

```
shartul@Shartul-O14EB8K: /mnt/d/gsk-task/ansible

shartul@Shartul-O14EB8K:/mnt/d/gsk-task/ansible$ ansible-playbook -i hosts deployment.yml

PLAY [Install docker, pull nginx image, run nginx container with sample content] ***********************************************************************************************************************************

TASK [install python] ***********************************************************************************************************************************
changed: [54.157.171.42]
changed: [35.153.159.204]

TASK [Install aptitude using apt] ***********************************************************************************************************************************
changed: [35.153.159.204]
changed: [54.157.171.42]

TASK [Install required system packages] ***********************************************************************************************************************************
ok: [54.157.171.42] => (item=apt-transport-https)
ok: [35.153.159.204] => (item=apt-transport-https)
ok: [54.157.171.42] => (item=ca-certificates)
ok: [35.153.159.204] => (item=ca-certificates)
ok: [54.157.171.42] => (item=curl)
ok: [35.153.159.204] => (item=curl)
ok: [35.153.159.204] => (item=software-properties-common)
ok: [54.157.171.42] => (item=software-properties-common)
changed: [35.153.159.204] => (item=python3-pip)
changed: [54.157.171.42] => (item=python3-pip)
changed: [35.153.159.204] => (item=virtualenv)
changed: [54.157.171.42] => (item=virtualenv)
ok: [35.153.159.204] => (item=python3-setuptools)
ok: [54.157.171.42] => (item=python3-setuptools)

TASK [Add Docker GPG apt Key] ***********************************************************************************************************************************
changed: [54.157.171.42]
changed: [35.153.159.204]

TASK [Add Docker Repository] ***********************************************************************************************************************************
changed: [35.153.159.204]
changed: [54.157.171.42]

TASK [Update apt and install docker-ce] ***********************************************************************************************************************************
```

ansible: execution result Part 2



Reference: Github

- Task 1: https://github.com/kshartul/terraform-labs

- Task2: https://github.com/kshartul/gsk-tasks

Summary

- Infrastructure as code is the future !
- Multi cloud tools like Terraform and Ansible help make this easier
- There will be a mix of mutable and immutable deployments
- Immutable will have an upper hand

Thank you