IT 445 – Capstone Implementation

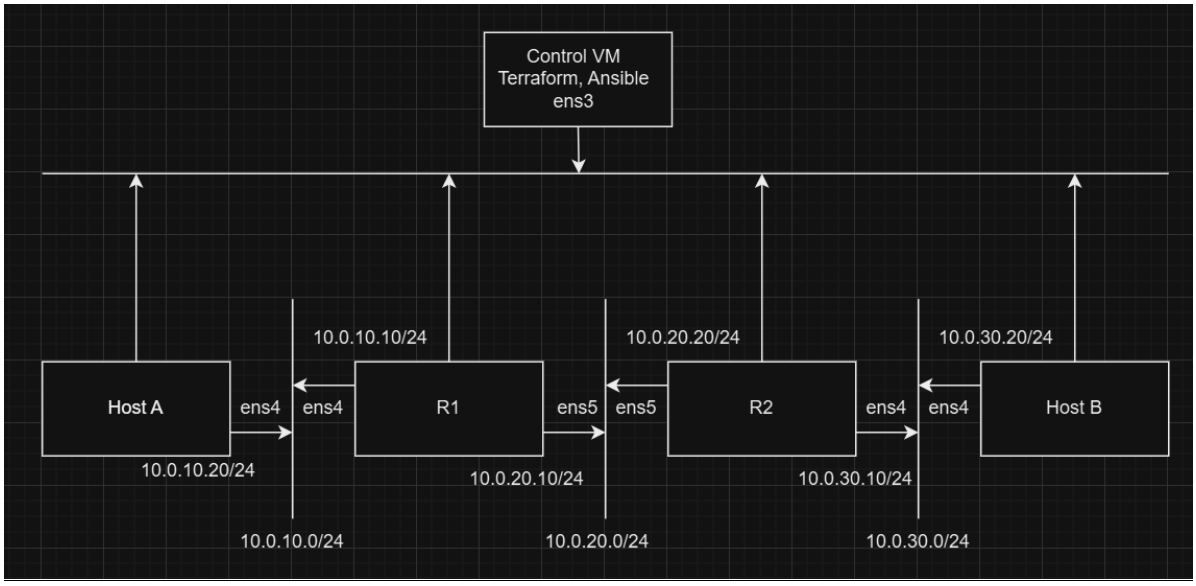
Lab 02 – Cloud Management, Automation and Configuration

1. **Learning Objectives:**
   - How does Terraform work?
   - How to develop a network infrastructure
   - How does ansible galaxy work?
2. **Equipment/software:**
   - Each member should have access to
     - VMWare Workstation
     - Ubuntu Desktop Control VM
     - Openstack Deployment
3. **Exercises**

   MAKE SURE TO PROVIDE NETWORK DIAGRAMs AND TABLEs CONTAINING INFORMATION ABOUT THE MACHINES INVOLVED (SUCH AS, IP ADDRESSES, MAC ADDRESSES, ETC.)

   NOTE THAT HAVING TO POST WIRESHARK PCAP/PCAPNG FILES IS NOT A SUBSITUTE FOR HAVING TO PROVIDE SCREEN SHOTS OF WIRESHARK CAPTURES AS EVIDENCE IN SUPPORT OF YOUR NARRATIVES AND ANSWERS.

   3.1 Exercise 1: Preparation (it should be assigned to the students before class)



Step 1: Cloud Infrastructure Development

We will be creating and setting up the infrastructure needed for Lab 1 in IT 333. The difference between this and the last lab is the tool being used to create the infrastructure, and the platform that the vms are being created on. Four VMs, 2 Desktop VMs and 2 Server VMs will be created and set up using Terraform, a software tool used to set up the infrastructure of cloud networks, and different tools being installed onto these vms by using Ansible.

Ensure that the control vm being used is up to date and has the necessary tools installed:

sudo apt-get update && sudo apt-get install -y gnupg software-properties- common

Install the HashiCorp GPG Key:

wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg > /dev/null

Verfiy the Key:

gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint

Add the official HashiCorp repository to your system:

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

Update the vm:

sudo apt update

Install Terraform:

sudo apt-get install terraform

Now that Terraform is installed onto the host machine, it is time to set up the credentials. These are necessary to authenticate the user that is trying to create the instances onto that specific openstack user. There are a few ways to go about this, with it either being added in the terraform file, or through an application credential [1].

An application credential gives you a way to authenticate the user without needing to embed any user account credentials onto any configuration files. This gives a seamless authentication process when creating the instances. To create an application credential, do the following:

Log into Openstack Horizon as a user

Go to Identity > Application Credential > Create Application Credential

Put "Terraform" under the name column

Leave every other box blank

Don't click on any role, it will auto assign both roles (member and reader)

Check the unrestricted box

Click "Create Application Credential"

This will give you ID, name, and secret of the application credential. Save these values, as they won't pop up again (**losing them will result in you having to recreate the application credential**)

Q1: What would be another way to authenticate the user without using the application credential?

Click on the "Download openrc file" and "Download clouds. yaml" buttons to download the files. These files will be the way that Openstack authenticates you as the authorized user of the Openstack account. A sample of what these files will look like can be seen in Appendix A and B with the ID and secret being different for each person.

Change the export OS_AUTH_URL=http://mykolla0.net:5000 to the url and port that is being used for your specific project as shown in Appendix B.

Q2: Why do we have to change the url and port? What would happen if we kept it the same?

Now that we got the User Authentication set up, we will now construct a simple configuration file, main.tf, that creates a single vm instance. This will be a test to see if it is possible for us to create a vm on openstack using an existing network. The configuration of the main.tf file can be found in Appendix C.

Once the main.tf file is completed, run the following four commands:

source openrc.sh

Terraform init

Terraform plan

Terraform apply

The "source openrc.sh" command adds the contents of the openrc.sh file into the configuration of Terraform on the control machine.

Q3: Explain the contents of the openrc.sh file. How does it differ from the clouds.yaml file.

Q4: Explain the role and difference between the terraform init, plan and apply using example outputs/results of these commands.

If successful, you should see the newly created vm in the instances tab in Openstack Horizon.

Now that that is successful and we can create a vm on openstack, lets replicate the environment for IT333 Lab1 Step2 by following the contents in Appendix D

After that's completed run the same three commands again:

Terraform init

Terraform plan

Terraform apply

If successful, you should see 4 created vms on the Openstack Horizon instance tab, and the newly created networks, subnets, and ports when looking at the networks tab. Each vm should have the ip addresses that were specified in the main.tf file with one public ip address.

Q5: Provide the mapping of the following sockets 134.126.152.136:8585, 134.126.152.136:8686, and 134.126.152.136:8787 into the internal sockets which are mykolla0.net:xxxx, mykolla0.net:xxxx, mykolla0.net:xxxx

Q6: Provide clear evidence of the outcome/results on openstack network topology. Provide the ip route show results on each instance.

Step 2: Automation Configuration

Now that we have the infrastructure set up with the 4 vms being created with the necessary networks, lets now set up the vms to have to ability to be automatically configured using Ansible.

Add a floating ip to each vm by doing the following in the instance tab:

Actions > Associate Floating IP > "Pick respective IP address" > Associate

Make sure that the user account has been allocated 4 floating ips.

If using the same control vm from the last lab, then there's no need to install Ansible. If a new vm is being used, then install Ansible by using the same steps from the last lab.

Q1: Why does the floating ip need to be associated to the instances? Which interface will the floating ip be associated with?

Q2: Why do we have to add the port number to the command? What would happen if it wasn't added?

If a ssh key has not been generated on the control vm, then do so:

ssh-keygen

Now we have to copy the public key to each vm respectively:

ssh-copy-id -p (port number) (username of vm)@(ip address of vm)

If an error pops up stating that "REMOTE HOST IDENTIFICATION HAS CHANGED", then copy and paste the command the error says into the command line, run it, and try again. This should resolve the error.

Create instances.ini file that contains the ip addresses of target machines

sudo vim instances.ini

Use the content of Appendix E to add the hosts to the instances.ini file.

Now that the file has been set up, Ansible has been installed and the ssh public key has been installed during the creation of the instances, lets utilize the playbook we wrote in the last lab to test that we can install packages onto these vms. If on a new vm, the file can be found in Appendix F.

Run the playbook:

ansible-playbook -i instances.ini nginx.yml

Q3: Explain the purpose of the inventory.ini file. Can playbooks be ran without the usage of the inventory.ini file?

The playbook will generate an error if you are using the same one from Lab 1, with the error being due to the instances needing to be updated. So a task has been added to the nginx.yml file that updates all of the instances listed in the instances.yml file. Now that the playbook has been modified to fix the error that was happening in the last lab, with it now updating the vms that the yml file is being ran on before installing the nginx package.

Q4: Why is the updating of the vms needed? Why did that error pop up last lab?

Check to see if nginx is officially installed onto the vm by running a command on the targeted vm:

sudo systemctl status nginx

You should see the package being up and running.

Realistically, Ansible playbooks aren't written like this and are generally more simplified, even for simple playbooks like this. Ansible galaxy is a platform where you can access preset packages and install them onto your target machines. These come with the setting already pre-determined and set up by a third party and are uploaded to a hub where anyone can access them. Let's use ansible galaxy to install nginx.

Q5: If wanting to install another package, such as apache, how would you go about finding the right package. Explain the different ways that can be used.

Install the preset package:

ansible-galaxy install geerlingguy.nginx

Create a galaxy_nginx.yml that installs nginx using this package as seen in Appendix G.

Run the playbook:

ansible-playbook -i instances.ini galaxy_nginx.yml

If successful, nginx should be reinstalled onto the 4 vms and up and running.

Q5: Create your own ansible galaxy playbook that installs a different package, such as sql, python, etc. Show the yml file created and add it to the report.

Step 3: Task Automation

While what we have done up to this point is effective, we can streamline the process for some of the tasks implemented, and new tasks that will be implemented, in the main.tf file. This will reduce the amount of manual tasks being done during the creation of the instances, with the playbook being ran during the running Terraform.

Let's start with the Port Security. This option is enabled automatically when the port is created in the main.tf file, but it can be turned off during its creation. We will need it to be off to be able have connectivity between all four of the vms.

Under the created ports in the main.tf file in Appendix L, add the following line:

port_security_enabled = false

This will create the port without the option enabled, or if the port is already created, will go and uncheck the option on each port that has that statement.

Run the commands:

Terraform init

Terraform plan

Terraform apply

If successful, the port security option should be unchecked on each port.

Q1: What other options can be modified on the ports created?

Next we will go and set up the creation of the the ssh keygen, which can be added as a task in the main.tf file. Appendix I shows the task that will be added in the main.tf file, under the provider "openstack" task. Once that is added, add thr following line to each created instance to add the key to the vms:

key_pair = openstack_compute_keypair_v2.test-keypair.name

This should then attach the created private key to each vm where the line is added in.

Run the commands:

Terraform init

Terraform plan

Terraform apply

If successful, a key pair should be created and attached to each created vm.

Next, we will set up the main.tf file to run an ansible playbook during the execution of Terraform. Add the contents of Appendix J to the end of the main.tf file. This will check to make sure that the vms were created successfully before running the playbook, with it stopping if there was an error with the vms being created.

Q2: Explain the purpose of the "null resource" and "local exec" resource block? How does each block work?

Execute the main.tf file to see the playbook get installed.

Now that the execution is successful, let's replicate the playbooks created in Lab 1, with us adding the respective routes to the vms, enabling ip forwarding on R1 and R2, and testing the pings between all of the vms. In one playbook, named Lab2.yml, add each playbook that was created in Lab1 into the Lab2.yml file with the proper modifications to the ip_routes.yml file, with the contents of the Lab2.yml being found in Appendix K if needed. **(changes will have to be made due to change in the ip addresses for each vm and name of interfaces)**

Q3: What is the difference between the routes added in Lab 1 and the routes that were added in Lab 2?

Run the commands:

Terraform init

Terraform plan

Terraform apply

If successful, the routes should be added to the vms, the ip forwarding should be enabled on R1 and R2, and the pings should be successful on each VM.

Q4: Construct a task in the main.tf file that automates the allocation of the floating ip address.

Step 4: Complete Automation

Now that we've seen these tasks individually become automated within the main.tf file, lets complete the full running of the file from scratch.

Clear any semblance of the created contents from the main.tf file from the OpenStack user account. This includes the networks, ports , subnets, and instances. Clear the application credentials that were created for the lab as well.

Once that is done, recreate the application credentials in the same format that it was done in Step 1, with the modifications from Step 1 and Appendix A and B being done as well.

Once the application credentials have been added to the control vm, run Terraform using the contents of Appendix L, with all of the updated actions being added to the file, and the ansible playbook from Appendix k being added to the end of the main.tf file to be ran at the end of the creation of the instances **(this could take some time).**

Terraform init

Terraform plan

Terraform apply

If successful, the entirety of the network should be created, with the ansible playbook being executed and applied to each instance targeted as well.

Appendix A: openrc.sh file

export OS_AUTH_TYPE=v3applicationcredential

export OS_AUTH_URL=http://134.126.152.136:8383/v3

export OS_IDENTITY_API_VERSION=3

unset OS_REGION_NAME

export OS_INTERFACE=public

export OS_APPLICATION_CREDENTIAL_ID=d159adc6e48a46268c7a446015aae1f3

export OS_APPLICATION_CREDENTIAL_SECRET=_ZPRRGfEWoZMAxcYl-eR5Q65mXAIw98tDb1WQLUOWQ9N2dkWISFcl9EDZBo8_rVdcE-rmkZL4-Xtpk9TN8RmQg

Appendix B: clouds.yaml

clouds:

osp_admin:

auth:

auth_url: http://134.126.152.136:8383

application_credential_id: "d159adc6e48a46268c7a446015aae1f3"

application_credential_secret: "_ZPRRGfEWoZMAxcYl-eR5Q65mXAIw98tDb1WQLUOWQ9N2dkWISFcl9EDZBo8_rVdcE-rmkZL4-Xtpk9TN8RmQg"

region_name: "RegionOne"

interface: "public"

identity_api_version: 3

auth_type: "v3applicationcredential"

Appendix C: main.tf with one cirros vm

terraform {

```
 required_providers {
  openstack = {
   source = "terraform-provider-openstack/openstack"
  }
 }
}

provider "openstack" {
 cloud = "osp_admin"

 endpoint_overrides = {
  image = "http://134.126.152.136:8585/v2.15/"
  compute = "http://134.126.152.136:8686/v2.1/"
  network = "http://134.126.152.136:8787/v2.0/"
 }


}


resource "openstack_compute_instance_v2" "server" {
 name       = "ABC"
 image_id    = "eae3fb54-1a84-4c15-ac8b-cb51fafd266b"
 flavor_id = "1"
 security_groups = ["default"]
```

```
  network {
    uuid = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
  }
}
```

Appendix D: main.tf with the four vms from IT 333 Lab1 Step2

```
terraform {
  required_providers {
    openstack = {
      source = "terraform-provider-openstack/openstack"
    }
  }
}

provider "openstack" {
  cloud = "osp_admin"

  endpoint_overrides = {
    image   = "http://134.126.152.136:8585/v2.15/"
    compute = "http://134.126.152.136:8686/v2.1/"
    network = "http://134.126.152.136:8787/v2.0/"
  }
}

resource "openstack_networking_network_v2" "Lab2-10" {
```

```
 name = "Lab2-10"
}

resource "openstack_networking_network_v2" "Lab2-20" {
 name = "Lab2-20"
}

resource "openstack_networking_network_v2" "Lab2-30" {
 name = "Lab2-30"
}

resource "openstack_networking_network_v2" "vmnet3" {
 name = "vmnet3"
}


resource "openstack_networking_subnet_v2" "subnet1" {
 name          = "subnet1"
 network_id    = openstack_networking_network_v2.Lab2-10.id
 cidr          = "10.0.10.0/24"
 ip_version    = 4
 gateway_ip    = "10.0.10.1"
}

resource "openstack_networking_subnet_v2" "subnet_2" {
 name          = "subnet_2"
```

```
 network_id     = openstack_networking_network_v2.Lab2-20.id
 cidr           = "10.0.20.0/24"
 ip_version     = 4
 gateway_ip     = "10.0.20.1"
}

resource "openstack_networking_subnet_v2" "subnet_3" {
 name           = "subnet_3"
 network_id     = openstack_networking_network_v2.Lab2-30.id
 cidr           = "10.0.30.0/24"
 ip_version     = 4
 gateway_ip     = "10.0.30.1"
}

resource "openstack_networking_port_v2" "port_1" {
 name       = "port_1"
 network_id  = openstack_networking_network_v2.Lab2-10.id
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet1.id
  ip_address = "10.0.10.20"
 }
}

resource "openstack_networking_port_v2" "port_2" {
 name       = "port_2"
 network_id  = openstack_networking_network_v2.Lab2-10.id
```

```
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet1.id
  ip_address = "10.0.10.10"
 }
}

resource "openstack_networking_port_v2" "port_3" {
 name       = "port_3"
 network_id  = openstack_networking_network_v2.Lab2-20.id
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_2.id
  ip_address = "10.0.20.10"
 }
}

resource "openstack_networking_port_v2" "port_4" {
 name       = "port_4"
 network_id  = openstack_networking_network_v2.Lab2-20.id
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_2.id
  ip_address = "10.0.20.20"
 }
}

resource "openstack_networking_port_v2" "port_5" {
 name       = "port_5"
```

```
 network_id  = openstack_networking_network_v2.Lab2-30.id
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_3.id
  ip_address = "10.0.30.10"
 }
}

resource "openstack_networking_port_v2" "port_6" {
 name       = "port_6"
 network_id  = openstack_networking_network_v2.Lab2-30.id
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_3.id
  ip_address = "10.0.30.20"
 }
}

resource "openstack_compute_instance_v2" "HostA" {
 name       = "HostA"
 image_id    = "606bf3aa-c091-4402-8104-e14543c66d78"
 flavor_id   = "3"
 security_groups = ["default"]

 network {
  uuid = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
 }
```

```
 network {
  port = openstack_networking_port_v2.port_1.id
 }

}


resource "openstack_compute_instance_v2" "R1" {
 name       = "R1"
 image_id    = "71379c8d-9f5e-4d08-ac95-cc3ff82c25e6"
 flavor_id   = "2-3"
 security_groups = ["default"]

 network {
  uuid = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
 }

 network {
  port = openstack_networking_port_v2.port_2.id
 }

 network {
  port = openstack_networking_port_v2.port_3.id
 }
}
```

```
resource "openstack_compute_instance_v2" "R2" {
 name       = "R2"
 image_id    = "71379c8d-9f5e-4d08-ac95-cc3ff82c25e6"
 flavor_id   = "2-3"
 security_groups = ["default"]

 network {
  uuid = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
 }

 network {
  port = openstack_networking_port_v2.port_4.id
 }

 network {
  port = openstack_networking_port_v2.port_5.id
 }
}

resource "openstack_compute_instance_v2" "HostB" {
 name       = "HostB"
 image_id    = "606bf3aa-c091-4402-8104-e14543c66d78"
 flavor_id   = "3"
 security_groups = ["default"]

 network {
```

```
   uuid = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
  }

  network {
   port = openstack_networking_port_v2.port_6.id
  }
}
```

Appendix E: instances.ini file

```
HostB ansible_host=134.126.152.136 ansible_port=6083 ansible_user=checkout
HostA ansible_host=134.126.152.136 ansible_port=6084 ansible_user=checkout
R1 ansible_host=134.126.152.136 ansible_port=6085 ansible_user=checkout
R2 ansible_host=134.126.152.136 ansible_port=6086 ansible_user=checkout
```

Appendix F: nginx.yml file

```
---
- name: Install nginx
  hosts: all
  become: yes
  tasks:

    - name: Update apt package index
      apt:
        update_cache: yes
```

```
  - name: Install nginx
    apt:
      name: nginx
      state: present
```

Appendix G: galaxy_nginx.yml

```
- name: Install and configure Nginx
  hosts: all
  become: yes
  roles:
    - geerlingguy.nginx
```

Appendix H: del_nginx.yml

```
- name: Uninstall Nginx
  hosts: all
  become: yes
  tasks:
    - name: Remove Nginx
      apt:
        name: nginx
        state: absent
```

Appendix I: SSH Keygen task

```
resource "openstack_compute_keypair_v2" "test-keypair" {
 name      = "my-keypair"
 public_key = file("~/.ssh/id_rsa.pub")
}
```

Appendix J: Task to add ansible playbook

```
resource "null_resource" "run_ansible" {
 depends_on = [
  openstack_compute_instance_v2.HostA,
  openstack_compute_instance_v2.R1,
  openstack_compute_instance_v2.R2,
  openstack_compute_instance_v2.HostB
 ]

 provisioner "local-exec" {
  command = "ansible-playbook -i instances.ini nginx.yml"
 }

}
```

Appendix K: Lab2.yml

```
---
- name: main yml file
 hosts: all
 become: yes
```

```
 tasks:

  - name: Update apt package index
    apt:
      update_cache: yes

  - name: Install nginx
    apt:
      name: nginx
      state: present

  - name: change routes for HostA
    when: inventory_hostname == 'HostA'
    shell: |
      sudo ip route add 10.0.20.0/24 via 10.0.10.10 dev ens4
      sudo ip route add 10.0.30.0/24 via 10.0.10.10 dev ens4

  - name: Add routes for R1
    when: inventory_hostname == 'R1'
    shell: |
      sudo ip route add 10.0.30.0/24 via 10.0.20.20 dev ens5

  - name: Add routes for R2
    when: inventory_hostname == 'R2'
    shell: |
      sudo ip route add 10.0.10.0/24 via 10.0.20.10 dev ens4
```

```
  - name: change routes for HostB
    when: inventory_hostname == 'HostB'
    shell: |
      sudo ip route add 10.0.10.0/24 via 10.0.30.10 dev ens4
      sudo ip route add 10.0.20.0/24 via 10.0.30.10 dev ens4



- name: Enable IP Forwarding on R1 and R2
  hosts: R1,R2
  become: yes
  tasks:

  - name: Ensure IP forwarding is enabled on boot
    lineinfile:
      path: /etc/sysctl.conf
      regexp: '^net.ipv4.ip_forward='
      line: 'net.ipv4.ip_forward=1'
      state: present

  - name: Apply sysctl changes
    shell: |
      sysctl -p

- name: Check connectivity between VMs
  hosts: all
```

```
 become: yes
 tasks:

   - name: Ping HostA from all VMs
     shell: ping -c 2 10.0.10.20
     ignore_errors: yes
     changed_when: false

   - name: Ping HostB from all VMs
     shell: ping -c 2 10.0.30.20
     ignore_errors: yes
     changed_when: false

   - name: Ping R1 eth1 from all VMs
     shell: ping -c 2 10.0.10.10
     ignore_errors: yes
     changed_when: false

   - name: Ping R1 eth2 from all VMs
     shell: ping -c 2 10.0.20.10
     ignore_errors: yes
     changed_when: false

   - name: Ping R2 eth1 from all VMs
     shell: ping -c 2 10.0.20.20
     ignore_errors: yes
```

```
    changed_when: false

  - name: Ping R2 eth2 from all VMs
    shell: ping -c 2 10.0.30.10
    ignore_errors: yes
    changed_when: false
```

Appendix L: Final main.tf file

```
terraform {
 required_providers {
  openstack = {
   source = "terraform-provider-openstack/openstack"
  }
 }
}

provider "openstack" {
 cloud = "osp_admin"

 endpoint_overrides = {
  image   = "http://134.126.152.136:8585/v2.15/"
  compute = "http://134.126.152.136:8686/v2.1/"
  network = "http://134.126.152.136:8787/v2.0/"
 }
}
```

```
resource "openstack_compute_keypair_v2" "test-keypair" {
 name      = "my-keypair"
 public_key = file("~/.ssh/id_rsa.pub")
}

resource "openstack_networking_network_v2" "Lab2-10" {
 name = "Lab2-10"
}

resource "openstack_networking_network_v2" "Lab2-20" {
 name = "Lab2-20"
}

resource "openstack_networking_network_v2" "Lab2-30" {
 name = "Lab2-30"
}

resource "openstack_networking_network_v2" "vmnet3" {
 name = "vmnet3"
}

resource "openstack_networking_subnet_v2" "subnet_1" {
 name          = "subnet_1"
 network_id     = openstack_networking_network_v2.vmnet3.id
 cidr          = "10.0.10.0/24"
 ip_version     = 4
```

```
 gateway_ip     = "10.0.10.1"
}

resource "openstack_networking_subnet_v2" "subnet_2" {
 name          = "subnet_2"
 network_id     = openstack_networking_network_v2.Lab2-20.id
 cidr          = "10.0.20.0/24"
 ip_version     = 4
 gateway_ip     = "10.0.20.1"
}

resource "openstack_networking_subnet_v2" "subnet_3" {
 name          = "subnet_3"
 network_id     = openstack_networking_network_v2.Lab2-30.id
 cidr          = "10.0.30.0/24"
 ip_version     = 4
 gateway_ip     = "10.0.30.1"
}

resource "openstack_networking_port_v2" "port_1" {
 name       = "port_1"
 network_id  = openstack_networking_network_v2.Lab2-10.id
 port_security_enabled = false
 fixed_ip {
   subnet_id  = openstack_networking_subnet_v2.subnet1.id
   ip_address = "10.0.10.20"
```

```
 }
}

resource "openstack_networking_port_v2" "port_2" {
 name       = "port_2"
 network_id  = openstack_networking_network_v2.Lab2-10.id
 port_security_enabled = false
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet1.id
  ip_address = "10.0.10.10"
 }
}

resource "openstack_networking_port_v2" "port_3" {
 name       = "port_3"
 network_id  = openstack_networking_network_v2.Lab2-20.id
 port_security_enabled = false
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_2.id
  ip_address = "10.0.20.10"
 }
}

resource "openstack_networking_port_v2" "port_4" {
 name       = "port_4"
 network_id  = openstack_networking_network_v2.Lab2-20.id
```

```
 port_security_enabled = false
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_2.id
  ip_address = "10.0.20.20"
 }
}

resource "openstack_networking_port_v2" "port_5" {
 name       = "port_5"
 network_id  = openstack_networking_network_v2.Lab2-30.id
 port_security_enabled = false
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_3.id
  ip_address = "10.0.30.10"
 }
}

resource "openstack_networking_port_v2" "port_6" {
 name       = "port_6"
 network_id  = openstack_networking_network_v2.Lab2-30.id
 port_security_enabled = false
 fixed_ip {
  subnet_id  = openstack_networking_subnet_v2.subnet_3.id
  ip_address = "10.0.30.20"
 }
}
```

```
resource "openstack_networking_port_v2" "public_1" {
 name      = "public_1"
 network_id = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
}

resource "openstack_networking_port_v2" "public_2" {
 name      = "public_2"
 network_id = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
}

resource "openstack_networking_port_v2" "public_3" {
 name      = "public_3"
 network_id = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
}

resource "openstack_networking_port_v2" "public_4" {
 name      = "public_4"
 network_id = "42a44f2d-b72f-41f0-87a2-53e8170223f8"
}

resource "openstack_networking_floatingip_associate_v2" "fip_assoc_1" {
 floating_ip = "192.168.1.34"
 port_id     = openstack_networking_port_v2.public_1.id
}
```

```
resource "openstack_networking_floatingip_associate_v2" "fip_assoc_2" {
 floating_ip = "192.168.1.35"
 port_id     = openstack_networking_port_v2.public_4.id
}

resource "openstack_networking_floatingip_associate_v2" "fip_assoc_3" {
 floating_ip = "192.168.1.36"
 port_id     = openstack_networking_port_v2.public_2.id
}

resource "openstack_networking_floatingip_associate_v2" "fip_assoc_4" {
 floating_ip = "192.168.1.37"
 port_id     = openstack_networking_port_v2.public_3.id
}

resource "openstack_compute_instance_v2" "HostA" {
 name       = "HostA"
 image_id    = "606bf3aa-c091-4402-8104-e14543c66d78"
 flavor_id   = "3"
 security_groups = ["default"]
 key_pair    = openstack_compute_keypair_v2.test-keypair.name

 network {
   port = openstack_networking_port_v2.public_1.id
 }
```

```
 network {
  port = openstack_networking_port_v2.port_1.id
 }
}

resource "openstack_compute_instance_v2" "R1" {
 name       = "R1"
 image_id    = "71379c8d-9f5e-4d08-ac95-cc3ff82c25e6"
 flavor_id   = "2-3"
 security_groups = ["default"]
 key_pair    = openstack_compute_keypair_v2.test-keypair.name

 network {
  port = openstack_networking_port_v2.public_2.id
 }

 network {
  port = openstack_networking_port_v2.port_2.id
 }

 network {
  port = openstack_networking_port_v2.port_3.id
 }
}

resource "openstack_compute_instance_v2" "R2" {
```

```
 name       = "R2"
 image_id    = "71379c8d-9f5e-4d08-ac95-cc3ff82c25e6"
 flavor_id   = "2-3"
 security_groups = ["default"]
 key_pair    = openstack_compute_keypair_v2.test-keypair.name

 network {
  port = openstack_networking_port_v2.public_3.id
 }

 network {
  port = openstack_networking_port_v2.port_4.id
 }

 network {
  port = openstack_networking_port_v2.port_5.id
 }
}

resource "openstack_compute_instance_v2" "HostB" {
 name       = "HostB"
 image_id    = "606bf3aa-c091-4402-8104-e14543c66d78"
 flavor_id   = "3"
 security_groups = ["default"]
 key_pair    = openstack_compute_keypair_v2.test-keypair.name
```

```
  network {
    port = openstack_networking_port_v2.public_4.id
  }

  network {
    port = openstack_networking_port_v2.port_6.id
  }
}

locals {
  get_port = {
    "HostA" = 6083
    "HostB" = 6084
    "R1"    = 6085
    "R2"    = 6086
  }

  get_ip = {
    "HostA" = "134.126.152.136"
    "HostB" = "134.126.152.136"
    "R1"    = "134.126.152.136"
    "R2"    = "134.126.152.136"
  }
}

resource "null_resource" "wait_for_ssh" {
```

```
 for_each = toset(["HostA", "HostB", "R1", "R2"])

 triggers = {
  always_run = timestamp()
 }

 depends_on = [
  openstack_compute_instance_v2.HostA,
  openstack_compute_instance_v2.HostB,
  openstack_compute_instance_v2.R1,
  openstack_compute_instance_v2.R2,
 ]

 provisioner "local-exec" {
  command = <<EOT
   /bin/bash -c '
   for i in {1..30}; do
    nc -z ${local.get_ip[each.key]} ${local.get_port[each.key]} && exit 0 || sleep 5;
   done;
   echo "SSH not ready for ${local.get_ip[each.key]}:${local.get_port[each.key]}" && exit 1
   '
  EOT
 }
}

resource "null_resource" "add_host_keys" {
```

```
 for_each = toset(["HostA", "HostB", "R1", "R2"])

 triggers = {
  always_run = timestamp()
 }

 depends_on = [null_resource.wait_for_ssh]

 provisioner "local-exec" {
  command = <<EOT
  /bin/bash -c 'ssh-keyscan -p ${local.get_port[each.key]} ${local.get_ip[each.key]} >>
~/.ssh/known_hosts'
  EOT
 }
}



resource "null_resource" "run_ansible" {
 triggers = {
  always_run = timestamp()
 }

 depends_on = [
  null_resource.add_host_keys,
 ]
```

```
  provisioner "local-exec" {
    command = "ansible-playbook -i instances.ini Lab2.yml"
  }
}
```