2228-CSE-5333-001 Cloud Computing

Lab 2

Name: Ranjith Gopal Reddy
Student Id: 1002028255

1. **Create a Compute Engine virtual machine (VM) instance, security group, and ssh keys using Google Cloud CLI and Python**

Python script to create and add the firewall to the VM,

In the below script if you take a closer look at the main function highlighted in yellow speaks about the tags assigned to the VM,

global tag_name
tag_name = 'webserver'

---

instance.tags = {
"items": [
tag_name
]

---

firewall_rule.target_tags = [tag_name]

```
import os
import re
import sys
from typing import Any, List
import warnings

from google.api_core.extended_operation import ExtendedOperation
from google.cloud import compute_v1
from google.cloud import translate_v3beta1



def get_image_from_family(project: str, family: str) -> compute_v1.Image:
    image_client = compute_v1.ImagesClient()
    # List of public operating system (OS) images:
https://cloud.google.com/compute/docs/images/os-details
```

```
    newest_image = image_client.get_from_family(project=project, family=family)
    return newest_image


def disk_from_image(
    disk_type: str,
    disk_size_gb: int,
    boot: bool,
    source_image: str,
    auto_delete: bool = True,
) -> compute_v1.AttachedDisk:

    boot_disk = compute_v1.AttachedDisk()
    initialize_params = compute_v1.AttachedDiskInitializeParams()
    initialize_params.source_image = source_image
    initialize_params.disk_size_gb = disk_size_gb
    initialize_params.disk_type = disk_type
    boot_disk.initialize_params = initialize_params
    # Remember to set auto_delete to True if you want the disk to be deleted when you
delete
    # your VM instance.
    boot_disk.auto_delete = auto_delete
    boot_disk.boot = boot
    return boot_disk


def wait_for_extended_operation(
    operation: ExtendedOperation, verbose_name: str = "operation", timeout: int = 300
) -> Any:

    result = operation.result(timeout=timeout)

    if operation.error_code:
        print(
            f"Error during {verbose_name}: [Code: {operation.error_code}]:
{operation.error_message}",
            file=sys.stderr,
            flush=True,
        )
        print(f"Operation ID: {operation.name}", file=sys.stderr, flush=True)
        raise operation.exception() or RuntimeError(operation.error_message)

    if operation.warnings:
        print(f"Warnings during {verbose_name}:\n", file=sys.stderr, flush=True)
        for warning in operation.warnings:
            print(f" - {warning.code}: {warning.message}", file=sys.stderr, flush=True)

    return result


def create_instance(
```

```
    project_id: str,
    zone: str,
    instance_name: str,
    disks: List[compute_v1.AttachedDisk],
    machine_type: str = "n1-standard-1",
    network_link: str = "global/networks/default",
    subnetwork_link: str = None,
    internal_ip: str = None,
    external_access: bool = False,
    external_ipv4: str = None,
    accelerators: List[compute_v1.AcceleratorConfig] = None,
    preemptible: bool = False,
    spot: bool = False,
    instance_termination_action: str = "STOP",
    custom_hostname: str = None,
    delete_protection: bool = False,
        target_tag: str = None
) -> compute_v1.Instance:

    instance_client = compute_v1.InstancesClient()

    tags = compute_v1.Tags()


    # Use the network interface provided in the network_link argument.
    network_interface = compute_v1.NetworkInterface()
    network_interface.name = network_link
    if subnetwork_link:
        network_interface.subnetwork = subnetwork_link

    if internal_ip:
        network_interface.network_i_p = internal_ip

    if external_access:
        access = compute_v1.AccessConfig()
        access.type_ = compute_v1.AccessConfig.Type.ONE_TO_ONE_NAT.name
        access.name = "External NAT"
        access.network_tier = access.NetworkTier.PREMIUM.name
        if external_ipv4:
            access.nat_i_p = external_ipv4
        network_interface.access_configs = [access]

    # Collect information into the Instance object.
    instance = compute_v1.Instance()
    instance.network_interfaces = [network_interface]
    instance.name = instance_name
    instance.tags = {
      "items": [
        tag_name
      ]
    }
    instance.disks = disks
```

```
    if re.match(r"^zones/[a-z\d\-]+/machineTypes/[a-z\d\-]+$", machine_type):
        instance.machine_type = machine_type
    else:
        instance.machine_type = f"zones/{zone}/machineTypes/{machine_type}"

    if accelerators:
        instance.guest_accelerators = accelerators

    if preemptible:
        # Set the preemptible setting
        warnings.warn(
            "Preemptible VMs are being replaced by Spot VMs.", DeprecationWarning
        )
        instance.scheduling = compute_v1.Scheduling()
        instance.scheduling.preemptible = True

    if spot:
        # Set the Spot VM setting
        instance.scheduling = compute_v1.Scheduling()
        instance.scheduling.provisioning_model = (
            compute_v1.Scheduling.ProvisioningModel.SPOT.name
        )
        instance.scheduling.instance_termination_action = instance_termination_action

    if custom_hostname is not None:
        # Set the custom hostname for the instance
        instance.hostname = custom_hostname

    if delete_protection:
        # Set the delete protection bit
        instance.deletion_protection = True

    # Prepare the request to insert an instance.
    request = compute_v1.InsertInstanceRequest()
    request.zone = zone
    request.project = project_id
    request.instance_resource = instance

    # Wait for the create operation to complete.
    print(f"Creating the {instance_name} instance in {zone}...")

    operation = instance_client.insert(request=request)

    wait_for_extended_operation(operation, "instance creation")

    print(f"Instance {instance_name} created.")
    return instance_client.get(project=project_id, zone=zone, instance=instance_name)

def create_firewall_rule(
    project_id: str, firewall_rule_name: str, network: str = "global/networks/default"
) -> compute_v1.Firewall:
```

```
    firewall_rule = compute_v1.Firewall()
    firewall_rule.name = firewall_rule_name
    firewall_rule.direction = "INGRESS"

    allowed_ports = compute_v1.Allowed()
    allowed_ports.I_p_protocol = "tcp"
    allowed_ports.ports = ["80", "443"]

    firewall_rule.allowed = [allowed_ports]
    firewall_rule.source_ranges = ["0.0.0.0/0"]
    firewall_rule.network = network
    firewall_rule.description = "Allowing TCP traffic on port 80 and 443 from Internet."

    firewall_rule.target_tags = [tag_name]

    # Note that the default value of priority for the firewall API is 1000.
    # If you check the value of `firewall_rule.priority` at this point it
    # will be equal to 0, however it is not treated as "set" by the library and thus
    # the default will be applied to the new rule. If you want to create a rule that
    # has priority == 0, you need to explicitly set it so:
    # TODO: Uncomment to set the priority to 0
    # firewall_rule.priority = 0

    firewall_client = compute_v1.FirewallsClient()
    operation = firewall_client.insert(
        project=project_id, firewall_resource=firewall_rule
    )

    wait_for_extended_operation(operation, "firewall rule creation")

    return firewall_client.get(project=project_id, firewall=firewall_rule_name)

if __name__ == '__main__':
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] =
'C:\\Users\\GopalReddyRanjith\\Downloads\\rgr-06-14de99dea2e2.json'

    googlebot = translate_v3beta1.TranslationServiceClient()

    global tag_name
    tag_name = 'webserver'

    disk_image = disk_from_image('zones/us-central1-a/diskTypes/pd-ssd', 10, True,
                                 'projects/ubuntu-os-cloud/global/images/ubuntu-1804-
bionic-arm64-v20220901', True)
    compute_vm_out = create_instance('rgr-06', 'us-central1-a', 'lab2', [disk_image])
    print(compute_vm_out)

    firewall_rule_out = create_firewall_rule('rgr-06', 'lab2')
    print(firewall_rule_out)
```

Output:

C:\Users\GopalReddyRanjith\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\GopalReddyRanjith\PycharmProjects\pythonProject\main.py

Creating the lab2 instance in us-central1-a...

Instance lab2 created.

cpu_platform: "Intel Haswell"

creation_timestamp: "2022-09-28T18:02:16.520-07:00"

deletion_protection: false

disks {

architecture: "ARM64"

auto_delete: true

boot: true

device_name: "persistent-disk-0"

disk_size_gb: 10

guest_os_features {

type_: "VIRTIO_SCSI_MULTIQUEUE"

}

guest_os_features {

type_: "UEFI_COMPATIBLE"

}

guest_os_features {

type_: "GVNIC"

}

index: 0

interface: "SCSI"

kind: "compute#attachedDisk"

licenses: "https://www.googleapis.com/compute/v1/projects/ubuntu-os-cloud/global/licenses/ubuntu-1804-lts"

mode: "READ_WRITE"

shielded_instance_initial_state {

dbxs {

content: "xxxxxxxx"

file_type: "BIN"

}

}

source: "https://www.googleapis.com/compute/v1/projects/rgr-06/zones/us-central1-a/disks/lab2"

type_: "PERSISTENT"

}

fingerprint: "LxdyDoIiwlM="

id: 5190622517615218807

kind: "compute#instance"

label_fingerprint: "42WmSpB8rSM="

last_start_timestamp: "2022-09-28T18:02:22.428-07:00"

machine_type: "https://www.googleapis.com/compute/v1/projects/rgr-06/zones/us-central1-a/machineTypes/n1-standard-1"

metadata {

fingerprint: "FFhFC7Oxmlg="

kind: "compute#metadata"

}

name: "lab2"

network_interfaces {

fingerprint: "O0U0H6_g2Qk="

kind: "compute#networkInterface"

name: "nic0"

network: "https://www.googleapis.com/compute/v1/projects/rgr-06/global/networks/default"

network_i_p: "10.128.0.16"

stack_type: "IPV4_ONLY"

subnetwork: "https://www.googleapis.com/compute/v1/projects/rgr-06/regions/us-central1/subnetworks/default"

}

scheduling {

automatic_restart: true

on_host_maintenance: "MIGRATE"

preemptible: false

provisioning_model: "STANDARD"

}

self_link: "https://www.googleapis.com/compute/v1/projects/rgr-06/zones/us-central1-a/instances/lab2"

shielded_instance_config {

enable_integrity_monitoring: true

enable_secure_boot: false

enable_vtpm: true

}

shielded_instance_integrity_policy {

update_auto_learn_policy: true

}

start_restricted: false

status: "RUNNING"

tags {

fingerprint: "_TImyF0dyaI="

items: "webserver"

}

zone: "https://www.googleapis.com/compute/v1/projects/rgr-06/zones/us-central1-a"

allowed {

I_p_protocol: "tcp"

ports: "80"

ports: "443"

}

creation_timestamp: "2022-09-28T18:02:35.047-07:00"

description: "Allowing TCP traffic on port 80 and 443 from Internet."

direction: "INGRESS"

disabled: false

id: 3746272879085965380

kind: "compute#firewall"

log_config {

enable: false

}

name: "lab2"

network: "https://www.googleapis.com/compute/v1/projects/rgr-06/global/networks/default"

priority: 1000

self_link: "https://www.googleapis.com/compute/v1/projects/rgr-06/global/firewalls/lab2"

source_ranges: "0.0.0.0/0"

target_tags: "webserver"

Process finished with exit code 0

Generate SSH key using the command below on the command prompt,

ssh-keygen

```
C:\Users\GopalReddyRanjith>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\GopalReddyRanjith/.ssh/id_rsa):
Created directory 'C:\Users\GopalReddyRanjith/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\GopalReddyRanjith/.ssh/id_rsa.
Your public key has been saved in C:\Users\GopalReddyRanjith/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:ogQgp5Ye+nH34jGyoOYsHbPozPybDhxbQlXjTVCCXcs azuread\gopalreddyranjith@OIT-SL-23294353
The key's randomart image is:
+---[RSA 3072]----+
|o ..+=++         |
|.+o...* .        |
|.=.  . E         |
|= ..             |
|.+..o o S        |
|..B+ o o         |
| *o+o + .        |
|Bo+..+ +         |
|*Bo=o .          |
+----[SHA256]-----+

C:\Users\GopalReddyRanjith>
```

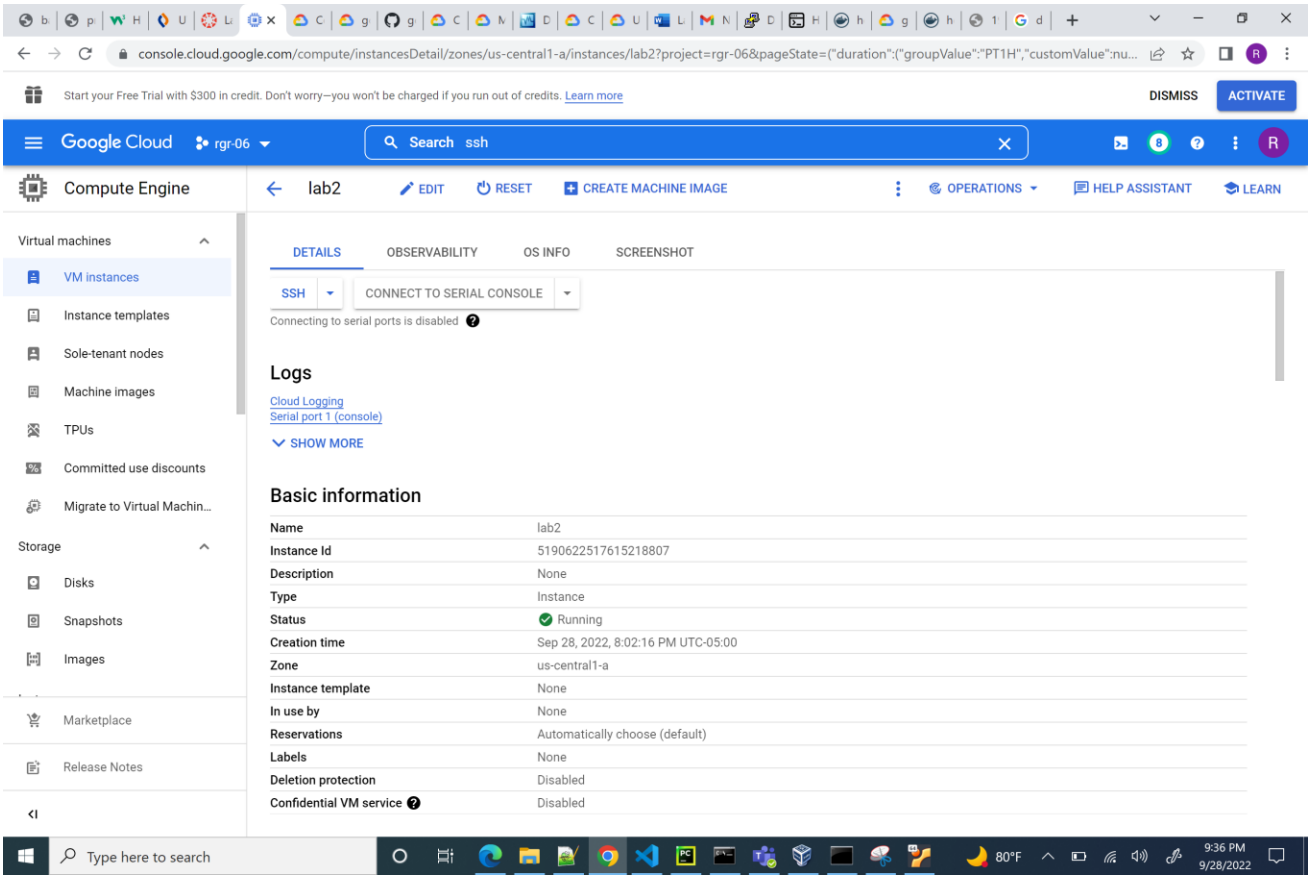
After generating the ssh key, add the ssh key to the VM using below command,

gcloud compute instances add-metadata lab2 --metadata-from-file ssh-keys=C:\Users\GopalReddyRanjith\.ssh\id_rsa.pub

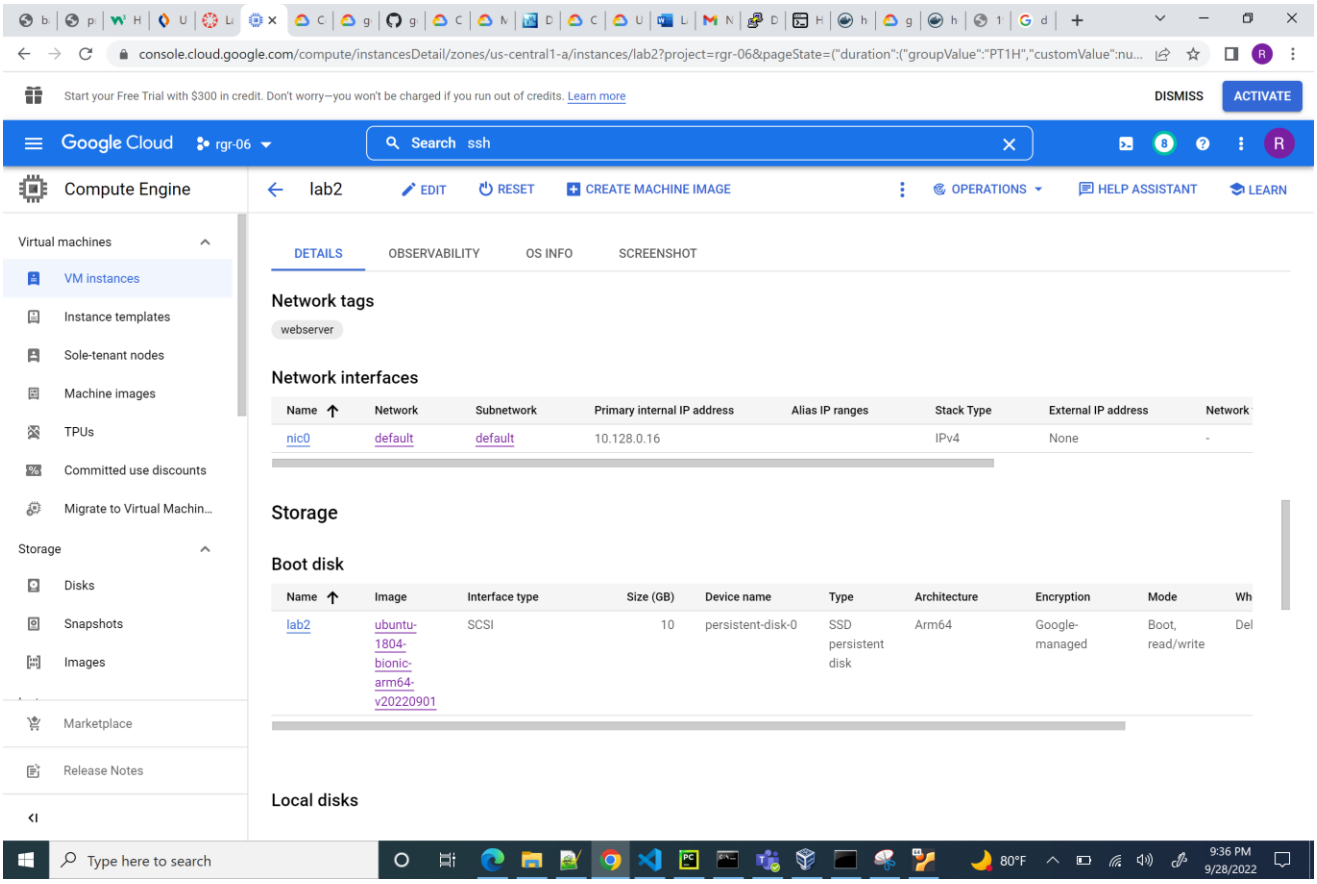
```
C:\Users\GopalReddyRanjith\AppData\Local\Google\Cloud SDK>gcloud compute instances add-metadata lab2 --metadata-from-file ssh-keys=C:\Users\GopalReddyRanjith\.ssh\id_rsa.pub
No zone specified. Using zone [us-central1-a] for instance: [lab2].
WARNING: The following key(s) are missing the <username> at the front
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDXyZleooIT5f9E/n3Db2udNDAqCZ1JTLMcf4gZDsz8inGMK+ot0g5r50SlK2QNucbg1BaQshYJZbiiRaH5A5h5NcXJAhJ4iuw3JiXN9m7D3JT2fKNm5+CaMkgg3hMHCtxatkkbULGDDEbd
dhK9mU5mYIZLV4/FP+yc2WBsoyBGM7YVmShd92s43l51IDUnHYO72Nhvo7pIU7VKmDxWVo25oCZwBxprPO3gjg+X7W3EBd3ddCfeooOspUpmsvyuVb74+OJtf3ndsRFR/98AzlvOGnQqSDoMvubNr1IPIkH3D+T2wq5Ejo8YQ8/2SksyU72A
/ihRgIO/F0GgSEkVQzBVs3B65i8NxA2ZLaQXJhOA7emxh3l6tFj4XG4btxiTVbhepLLfn6lJsy2s0aJyUsvJ0TZRPyBbq+yMzCBhuDBwWVi0jCgnWPGiSI5DPi9LChZmTG2Kvwg+nVd9JMf1nGyhg1nQkZkizTOMrOItSVecnFTJMOQvbsHf
ZQv0q149uts= azuread\gopalreddyranjith@OIT-SL-23294353

Format ssh keys following https://cloud.google.com/compute/docs/instances/adding-removing-ssh-keys
Updated [https://www.googleapis.com/compute/v1/projects/rgr-06/zones/us-central1-a/instances/lab2].
```

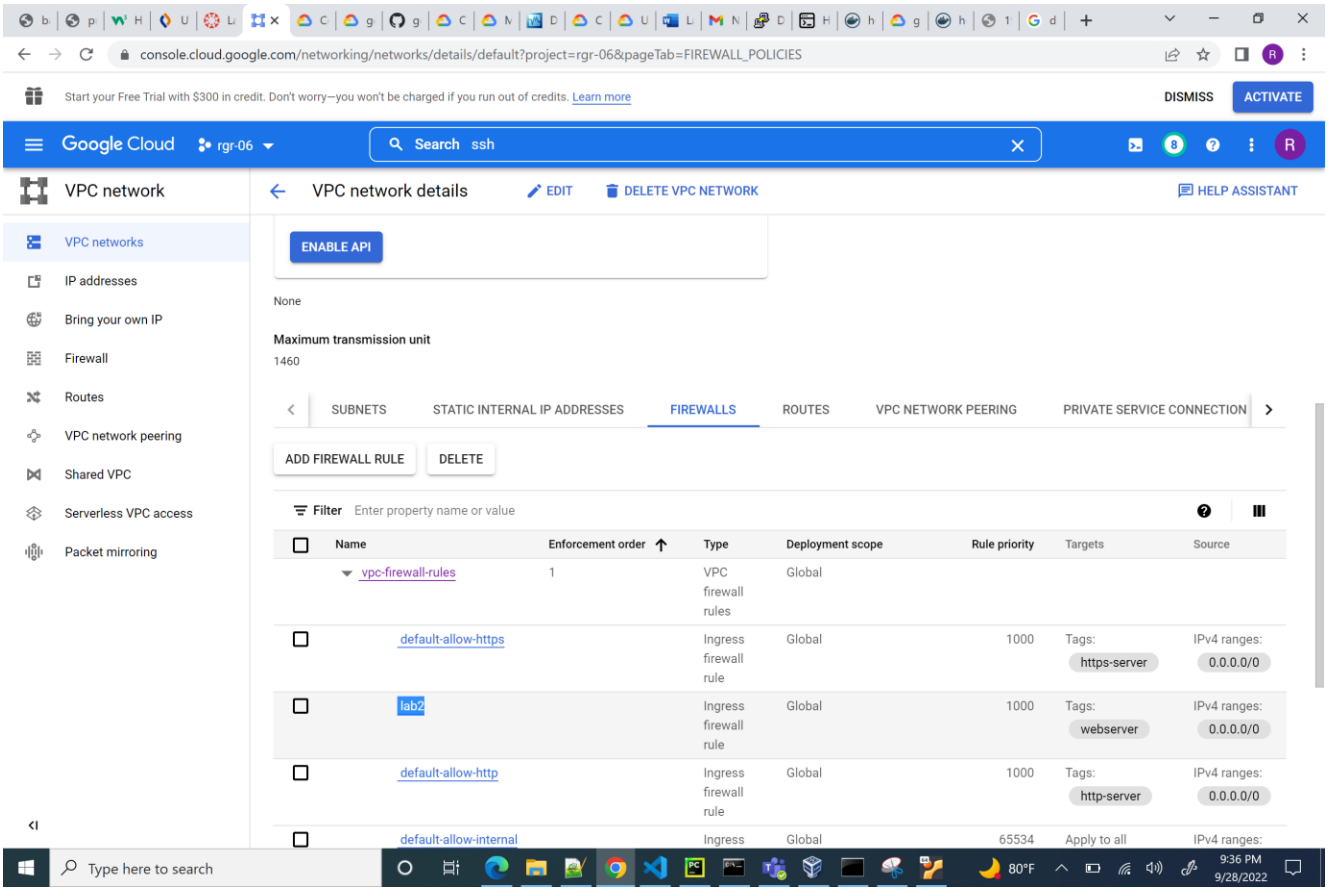
After the steps are executed, we take a look at the GCP console to see that **lab2** VM is created



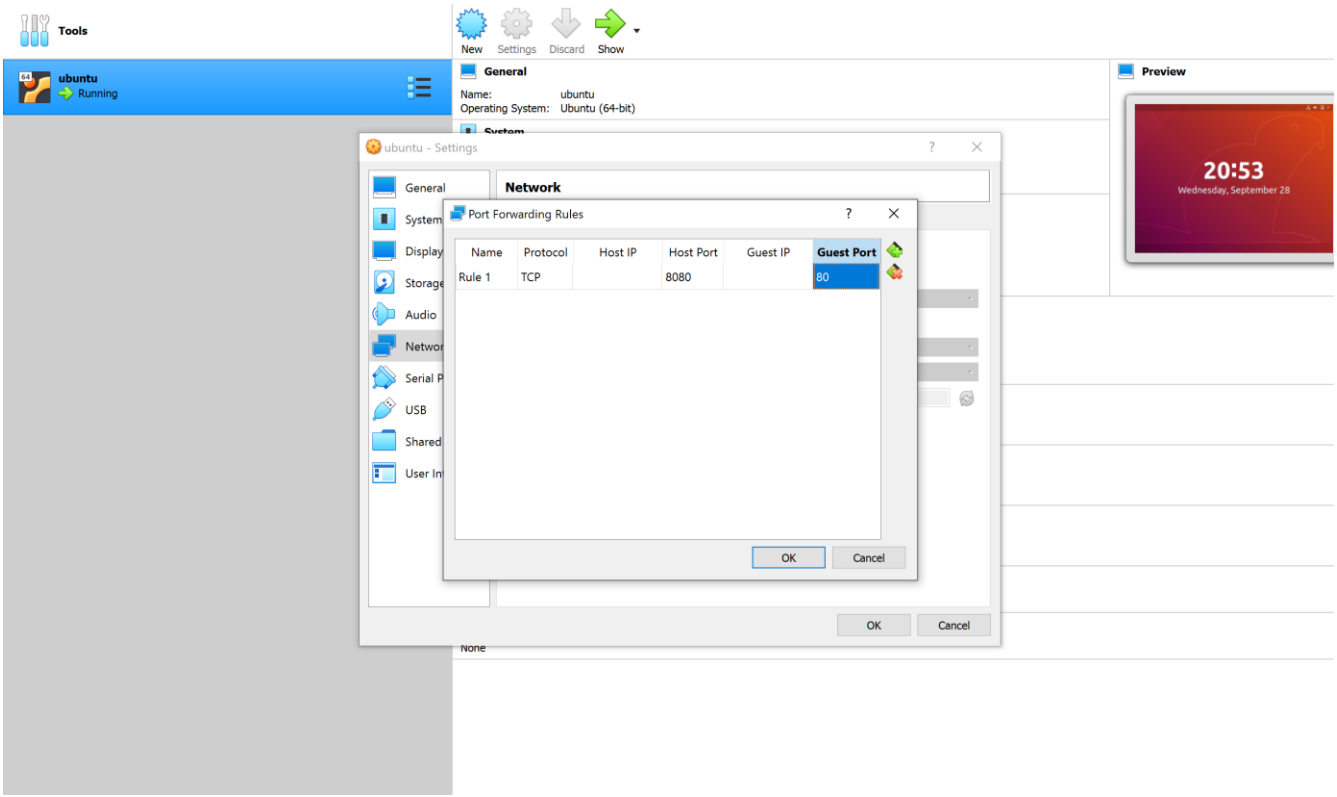
If we take a look at the networking side of the VM, we can find that default VPC is configured with the network tag as webserver,



Under the default VPC, upon navigating to the firewall section we can find that lab2 firewall rule is created with the tag *webserver*,



2. Configure VirtualBox to allow for inbound IP traffic



3. **Install and configure Docker and run a hello world application that can be called from the host machine**

Install docker on the VM using the below command,

sudo apt install docker.io

Run the hello-world application on docker using the command,

Step1: create index.html

<h1>Welcome to cloud computing, Lab2</h1>

Step2: create Dockerfile in the same directory as index.html with steps

FROM nginx:latest

COPY ./index.html /usr/share/nginx/html/index.html

Step3: run the docker image using the docker file

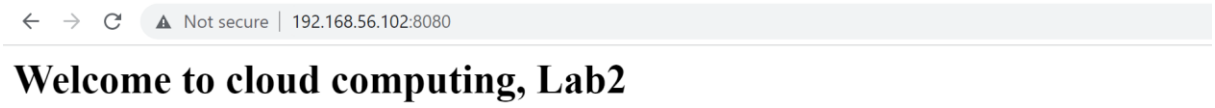
```
docker build -t webserver .
```

sudo docker run -it --rm -d -p 8080:80 --name web webserver

Check if the hello-world using the below command,

```
ubuntu@ubuntu-VirtualBox:~$ echo '<h1>Welcome to cloud computing, Lab2</h1>' > index.html
ubuntu@ubuntu-VirtualBox:~$ echo '<h1>Welcome to cloud computing, Lab2</h1>' > index.html^C
ubuntu@ubuntu-VirtualBox:~$ ^C
ubuntu@ubuntu-VirtualBox:~$ ls
Desktop  Documents  Downloads  examples.desktop  index.html  Music  Pictures  Public  Templates  Videos
ubuntu@ubuntu-VirtualBox:~$ vi index.html
ubuntu@ubuntu-VirtualBox:~$ touch Dockerfile
ubuntu@ubuntu-VirtualBox:~$ cat Doc
Dockerfile  Documents/
ubuntu@ubuntu-VirtualBox:~$ cat Dockerfile
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
ubuntu@ubuntu-VirtualBox:~$ sudo docker build -t webserver .
Sending build context to Docker daemon  90.73MB
Step 1/2 : FROM nginx:latest
 ---> 2d389e545974
Step 2/2 : COPY ./index.html /usr/share/nginx/html/index.html
 ---> 8d0dc137d4bb
Successfully built 8d0dc137d4bb
Successfully tagged webserver:latest
ubuntu@ubuntu-VirtualBox:~$ docker run -it --rm -d -p 8080:80 --name web webserver
docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containe
rs/create?name=web": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
ubuntu@ubuntu-VirtualBox:~$ sudo docker run -it --rm -d -p 8080:80 --name web webserver
c78439d514ae911b927a85d200732009a11b9c122394e727fd72dfe94dae43de
ubuntu@ubuntu-VirtualBox:~$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                   NAMES
c78439d514ae   webserver   "/docker-entrypoint.…"   35 seconds ago   Up 34 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   web
ubuntu@ubuntu-VirtualBox:~$
```

Step4: From the host machine, call the virtualbox vm ip with port 8080, as it is configured to hello-world, to load the index.html page



Welcome to cloud computing, Lab2

To find VM, Ip use command *ifconfig*

```
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::5b63:d1aa:a64b:ff74  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:db:b9:75  txqueuelen 1000  (Ethernet)
        RX packets 4423  bytes 381420 (381.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4317  bytes 637315 (637.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```