Team Members: Aadya Maurya and Esha Patel

We intend to configure a multi-site VPN for our project. We will use Wireguard to set this up. We are planning to use this article as a guide to set it up:
https://www.digitalocean.com/community/tutorials/how-to-set-up-wireguard-on-ubuntu-20-04
We also skipped all optional parts.

First, we downloaded the ubuntu 20.04 image from this link to use as wireguard server and peers.
https://www.linuxvmimages.com/images/ubuntu-2004/#system-details--login-information

Documentation:

1. We changed the network adapter to NAT for the ubuntu 20.04 vm.
2. Updated and installed wireguard on the server.
3. Generated a public and private keypair for the server

```
ubuntu@ubunu2004:~$ wg genkey | sudo tee /etc/wireguard/private.key
ULwlSAAr70Q6XLseQ9TQvrL50W4yUuV+RoWAEDGWUXk=
ubuntu@ubunu2004:~$ sudo chmod go= /etc/wireguard/private.key
ubuntu@ubunu2004:~$ ^C
ubuntu@ubunu2004:~$ sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee /etc/wireguard/public.key
45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
```

4. Changed permissions so that only the root user can access the private key. Command used: sudo chmod go= /etc/wireguard/private.key
5. We chose the IPv4 range that we wanted to use: 10.8.0.0/24
6. Created wireguard server configuration file called wg0.conf. It includes:

```
[Interface]
PrivateKey = ULwlSAAr70Q6XLseQ9TQvrL50W4yUuV+RoWAEDGWUXk=
Address = 10.8.0.1/24, fd0d:86fa:c3bc::1/64
ListenPort = 51820
SaveConfig = true

PostUp = ufw route allow in on wg0 out on enp0s3
PostUp = iptables -t nat -I POSTROUTING -o enp0s3 -j MASQUERADE
PreDown = ufw route delete allow in on wg0 out on enp0s3
PreDown = iptables -t nat -D POSTROUTING -o enp0s3 -j MASQUERADE
```

7. Configured IP forwarding
    a. First, we configured the sysctl.conf so it includes this line: net.ipv4.ip_forward=1
    b. This will allow the wireGuard Server to forward incoming traffic from the virtual VPN device to others and then to the public internet.

```
ubuntu@ubunu2004:~$ sudo sysctl -p
net.ipv4.ip_forward = 1
```

8. Added firewall rules to make sure that traffic is routed properly. We enabled masquerading. wg0.conf file:

```
[Interface]
PrivateKey = ULwlSAAr70Q6XLseQ9TQvrL50W4yUuV+RoWAEDGWUXk=
Address = 10.8.0.1/24, fd0d:86fa:c3bc::1/64
ListenPort = 51820
SaveConfig = true

PostUp = ufw route allow in on wg0 out on enp0s3
PostUp = iptables -t nat -I POSTROUTING -o enp0s3 -j MASQUERADE
PreDown = ufw route delete allow in on wg0 out on enp0s3
PreDown = iptables -t nat -D POSTROUTING -o enp0s3 -j MASQUERADE
```

We ran sudo ufw status command to check if the rules are in place:

```
ubuntu@ubunu2004:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
51820/udp                  ALLOW       Anywhere
OpenSSH                    ALLOW       Anywhere
51820/udp (v6)             ALLOW       Anywhere (v6)
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

9. Started the wireguard server and made sure it was running properly. First we enabled the wg-quick service, then we started the service and checked the status.

```
ubuntu@ubunu2004:~$ sudo systemctl enable wg-quick@wg0.service
Created symlink /etc/systemd/system/multi-user.target.wants/wg-quick@wg0.service → /lib/systemd/system/wg-quick@.service.
ubuntu@ubunu2004:~$ sudo systemctl start wg-quick@wg0.service
ubuntu@ubunu2004:~$ sudo systemctl status wg-quick@wg0.service
● wg-quick@wg0.service - WireGuard via wg-quick(8) for wg0
     Loaded: loaded (/lib/systemd/system/wg-quick@.service; enabled; vendor pre>
     Active: active (exited) since Sat 2024-04-20 01:14:43 IST; 1min 36s ago
       Docs: man:wg-quick(8)
             man:wg(8)
             https://www.wireguard.com/
             https://www.wireguard.com/quickstart/
             https://git.zx2c4.com/wireguard-tools/about/src/man/wg-quick.8
             https://git.zx2c4.com/wireguard-tools/about/src/man/wg.8
    Process: 4688 ExecStart=/usr/bin/wg-quick up wg0 (code=exited, status=0/SUC>
   Main PID: 4688 (code=exited, status=0/SUCCESS)

Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip link add wg0 type wireguard
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] wg setconf wg0 /dev/fd/63
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip -4 address add 10.8.0.1/24 dev>
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip -6 address add fd0d:86fa:c3bc:>
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ip link set mtu 1420 up dev wg0
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] ufw route allow in on wg0 out on >
Apr 20 01:14:43 ubunu2004 wg-quick[4739]: Rule added
Apr 20 01:14:43 ubunu2004 wg-quick[4739]: Rule added (v6)
Apr 20 01:14:43 ubunu2004 wg-quick[4688]: [#] iptables -t nat -I POSTROUTING -o>
Apr 20 01:14:43 ubunu2004 systemd[1]: Finished WireGuard via wg-quick(8) for wg>
```

10. On our peer 1 VM, we updated and installed wireguard.
11. Generated public and private key pair for the peer vm.

```
ubuntu@ubunu2004:~$ wg genkey | sudo tee /etc/wireguard/priva
te.key
8EKKFpUA8Rxy1sIFaNw5t48nSEnpr6NWxRgDWwStJ1w=
ubuntu@ubunu2004:~$ sudo chmod go= /etc/wireguard/private.key
ubuntu@ubunu2004:~$ sudo cat /etc/wireguard/private.key | wg
pubkey | sudo tee /etc/wireguard/public.key
AFX4y2o3hcfpsuulVf8VB8ndRySpDhccxER/JLVNqQ0=
```

12. Created configuration file wg0.conf for the peer 1. We added the private key and allowed IPs inside it.

```
  GNU nano 4.8        /etc/wireguard/wg0.conf
[Interface]
PrivateKey = 8EKKFpUA8Rxy1sIFaNw5t48nSEnpr6NWxRgDWwStJ1w=
Address = 10.8.0.2/24

#PostUp = ip rule add table 200 from 10.0.2.4
#PostUp = ip route add table 200 default via 10.0.2.1
#PreDown = ip rule delete table 200 from 10.0.2.4
#PreDown = ip route delete table 200 default via 10.0.2.1

[Peer]
PublicKey = 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
AllowedIPs = 10.8.0.0/24
Endpoint = 10.0.2.15:51820
```

13. Add peer's public key to the wireguard server so that we can connect and route traffic over the VPN. We used the sudo wg set wg0 peer command to set the peer 1 up.
14. Connect peer to the VPN tunnel. We installed resolvconf and then ran 'sudo wg-quick up wg0' to start the tunnel.

```
ubuntu@ubunu2004:~$ sudo wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.8.0.2/24 dev wg0
[#] ip link set mtu 1420 up dev wg0
```

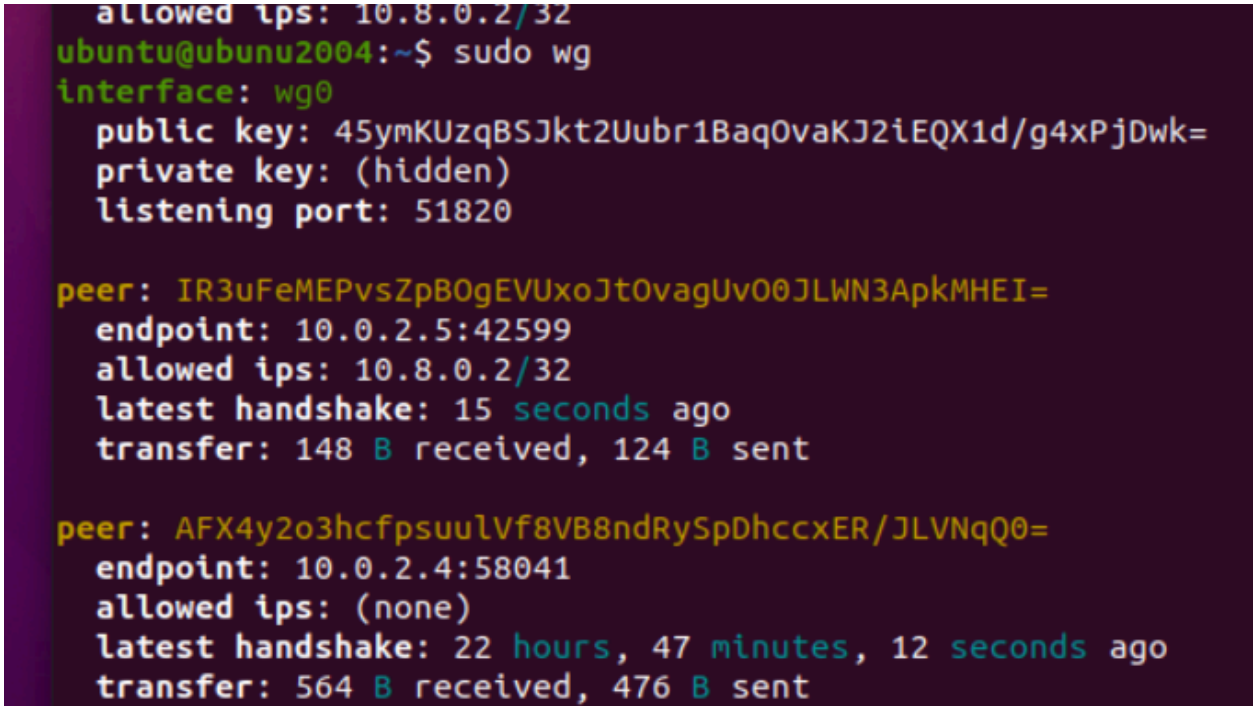
We then checked the status of the tunnel on the peer 1 using the wg show command.

```
ubuntu@ubunu2004:~$ sudo wg show
interface: wg0
  public key: AFX4y2o3hcfpsuulVf8VB8ndRySpDhccxER/JLVNqQ0=
  private key: (hidden)
  listening port: 58041

peer: 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
  endpoint: 10.0.2.15:51820
  allowed ips: 10.8.0.0/24
  latest handshake: 22 hours, 37 minutes, 24 seconds ago
  transfer: 476 B received, 564 B sent
```

15. We re-did all of the steps in peer 2's VM. Here is the wg show command on peer 2:

```
ubuntu@ubunu2004:~$ sudo wg show
interface: wg0
  public key: IR3uFeMEPvsZpBOgEVUxoJtOvagUvO0JLWN3ApkMHEI=
  private key: (hidden)
  listening port: 42599

peer: 45ymKUzqBSJkt2Uubr1BaqOvaKJ2iEQX1d/g4xPjDwk=
  endpoint: 10.0.2.15:51820
  allowed ips: 10.8.0.0/24
```

16. Here is the sudo wg command running on the wireguard server showing both peer's public keys:



Reflection:
One of the main issues we dealt with was the ip configuration. Since we used the same VM image and tutorial for each VM, we had the same IP address for all of them. This caused issues with the network and we didn't get the handshake after connecting. After changing the ip addresses, it worked. We also had to change the adapter from NAT to NAT network for all VMs.