Open VPN Server

**Prerequisite :**

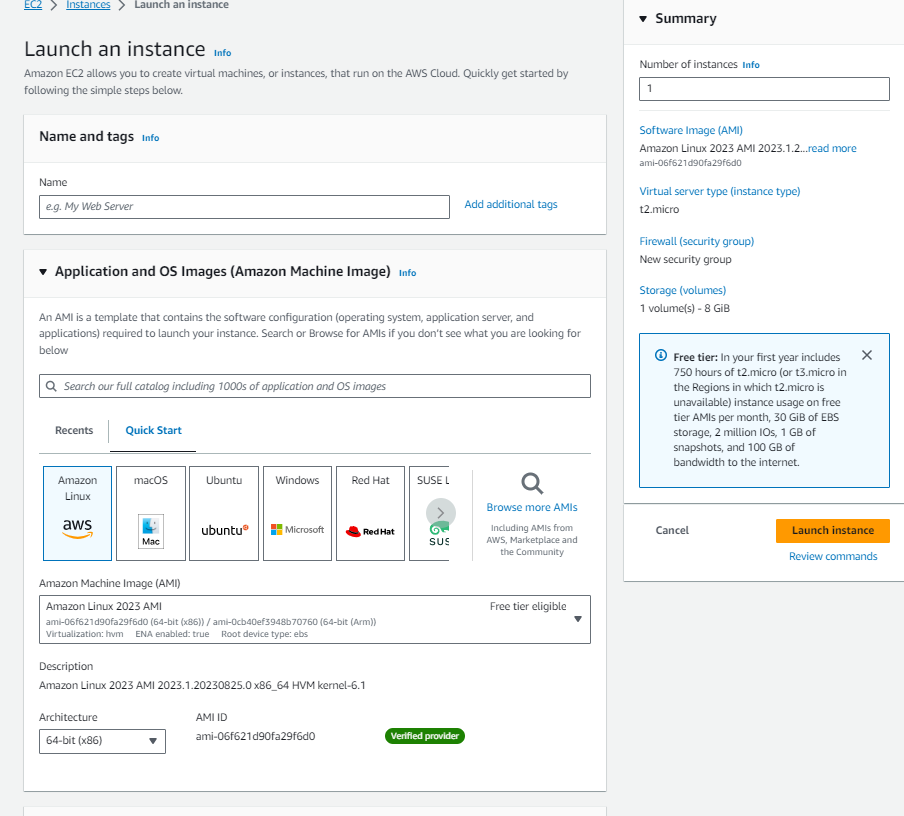
* AWS console access.
* Already configure VPN and subnets associated with it. If not, please go through with the AWS VPN Documentation file to do so.

**Setup and Launch EC2 Instance :**

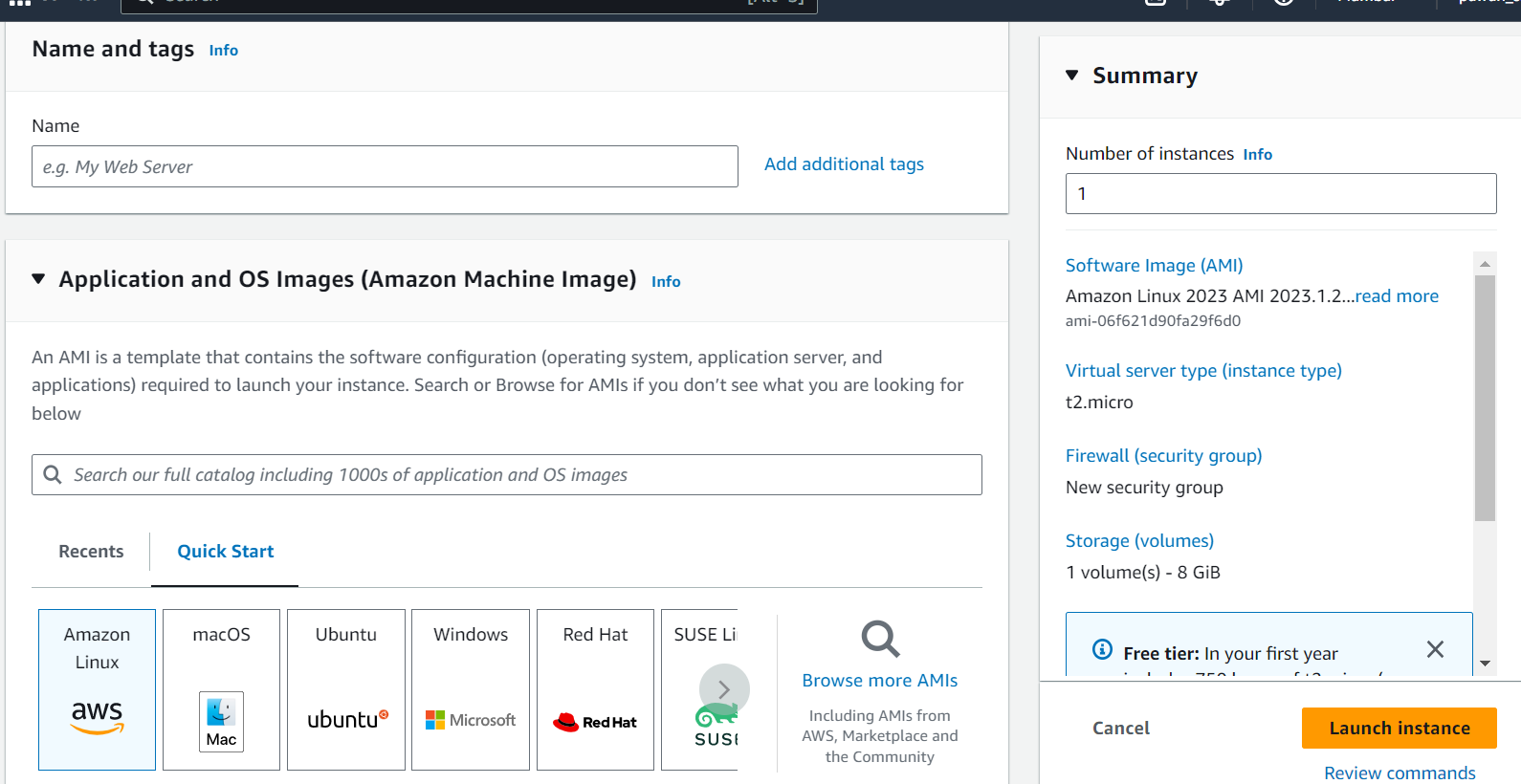
Installing any server, first we need is a EC2 machine running a public subnet so that anyone can reach the VPN server running on that machine. So, let’s move forward with creating an EC2 instance in our public subnet.

Steps:

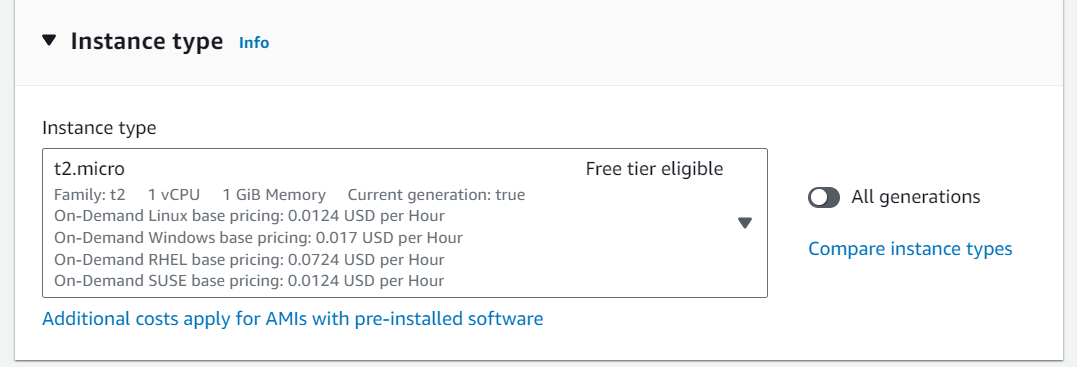
1. Login to your AWS console.
2. Search for EC2 instances.
3. Launch Instance by clicking on ‘Launch Instance’ and it will land to the page similar to following-



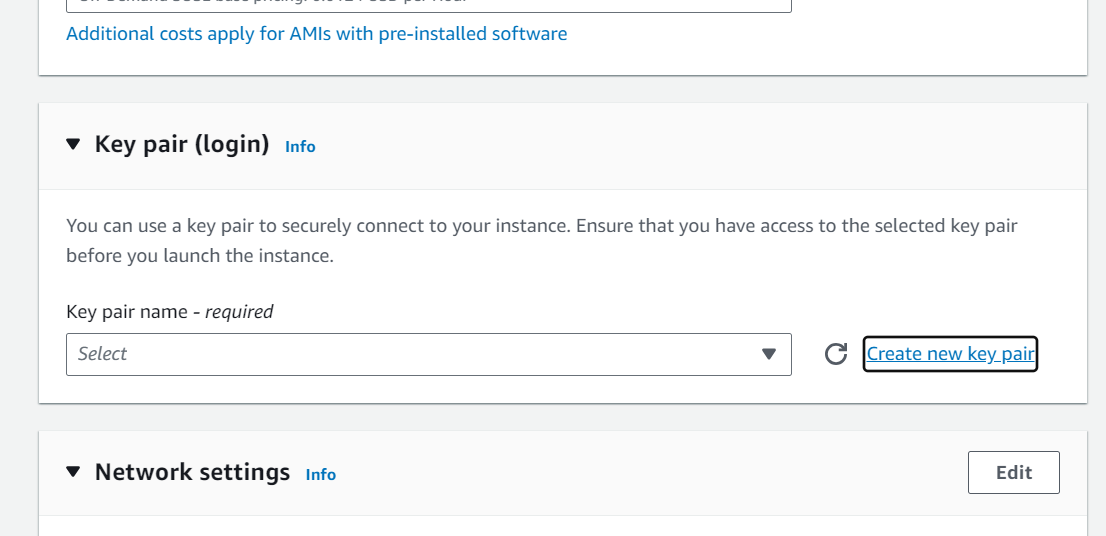
1. Fill the details as below –
   1. Give a name to the machine, we name it as ‘VPN\_server’.
   2. Choose application OS Images like ubuntu, windows or any thing else. We choose ubuntu for now.



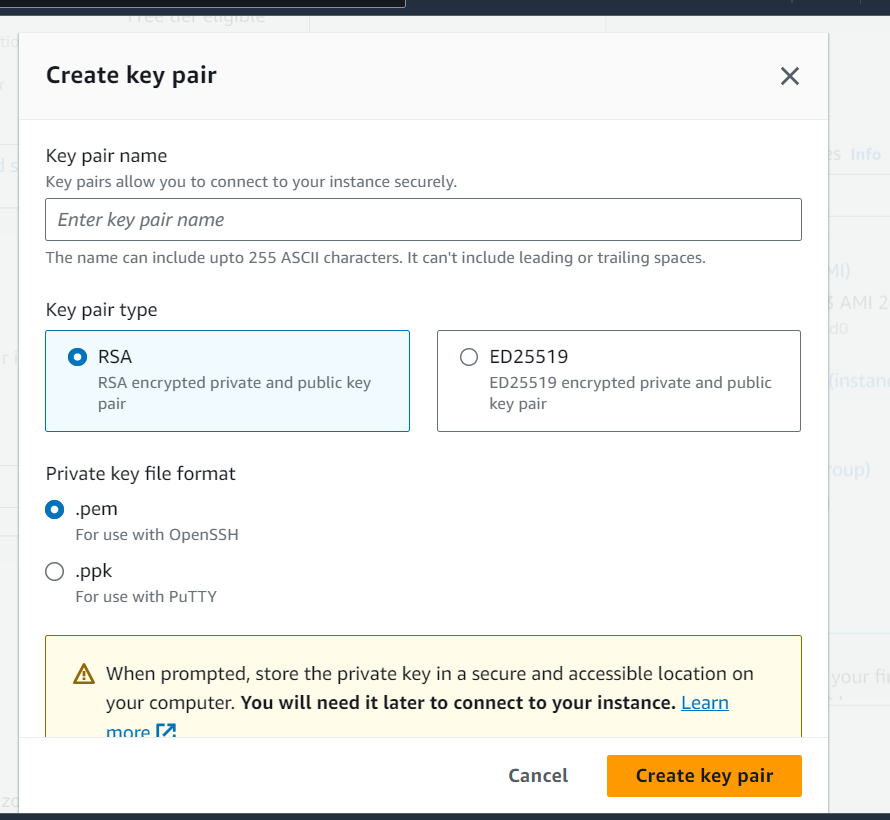
1. Now choose Instance type as par your need, we are choosing free tier ‘t2.micro’.



1. If you have already generated a key pair then choose one or else create a new key pair. We’ll create a new key pair.

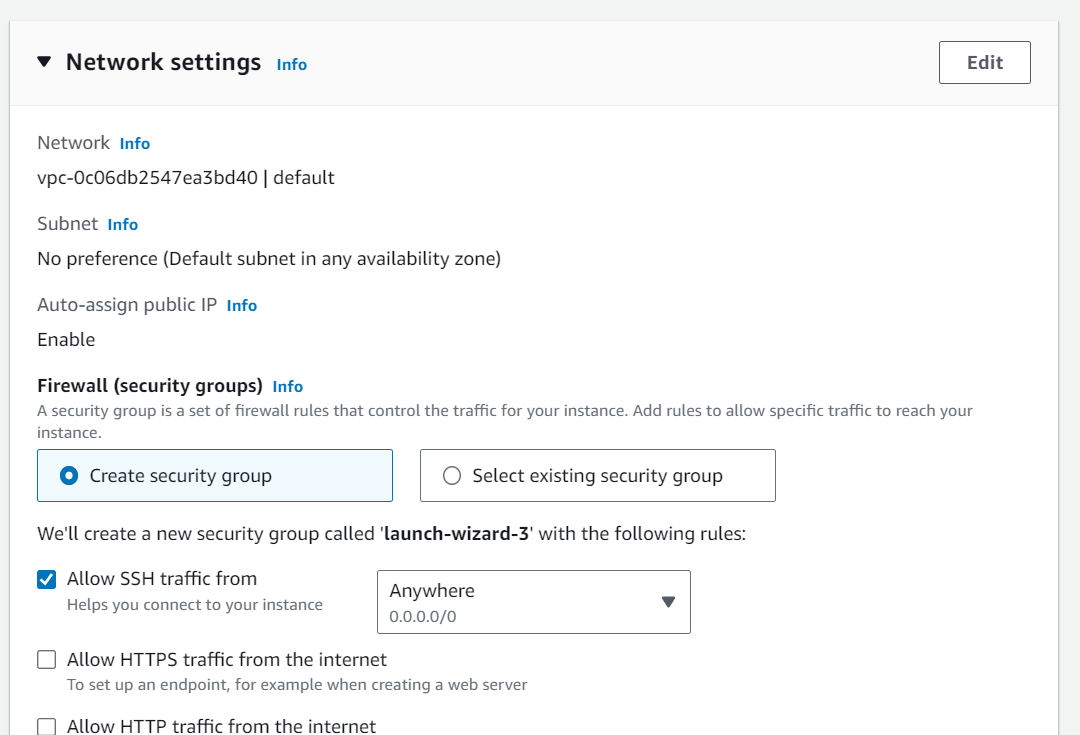


1. Click on ‘create new key pair’ and it will pop up a model dialog box as below-

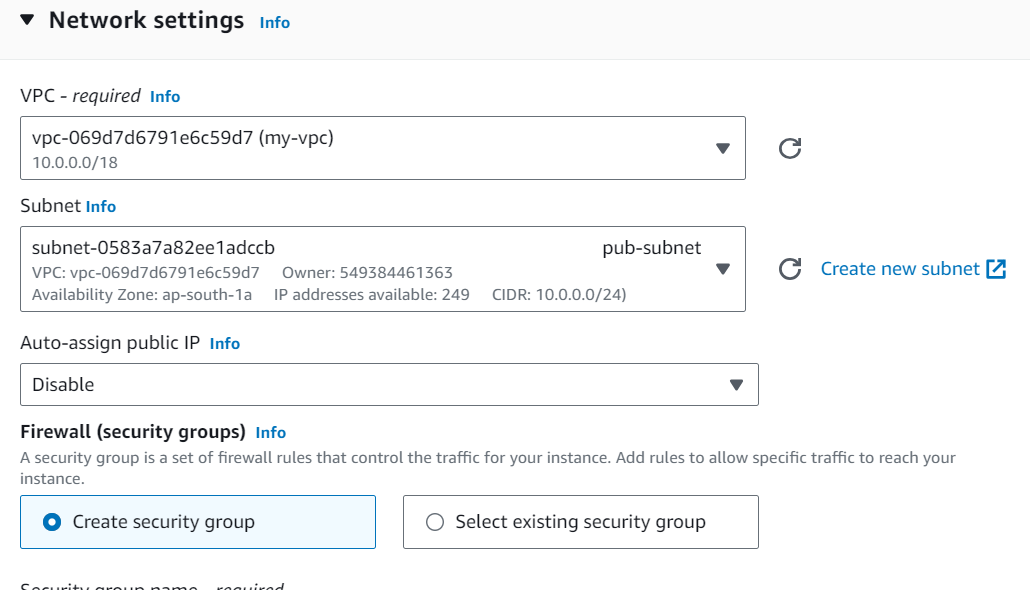


1. Give a name to your key pair and hit create key pair.
2. Please save the key downloaded as it is important for login to the machine.

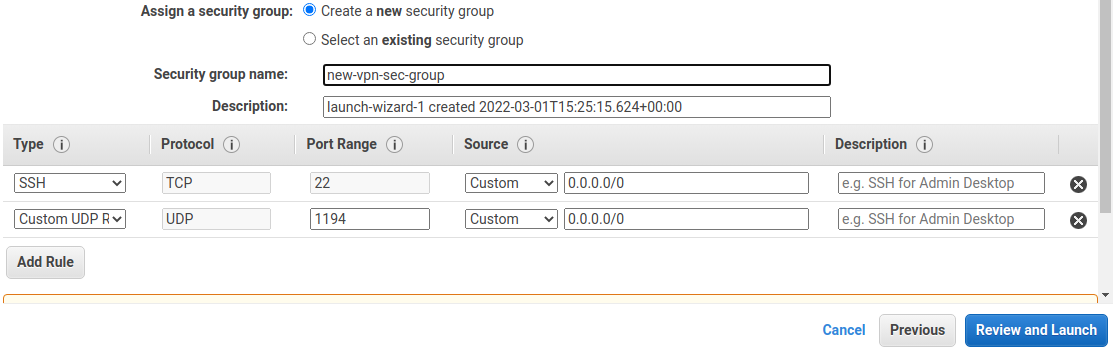
1. Now in ‘Network Setting' section click on ‘Edit’.



1. Now edit following details inside it-
   1. Choose your VPN and the public subnet so that the machine can be accessible from out side but it can communicate with other subnet inside your VPN.



1. Now under Firewall, select an existing security group or  create a security group and add some inbound security group rules to it.



Note- you can add or remove security group rules by using ‘Add security group rule’ and ‘Remove’ buttons under this section as in above fig.

1. Now, we can scale up our storage if we require more storage. Here we’ll proceed with free tier availability.
2. So, we are done with setting up now click on ‘Launch Instance’ to launch the instance with these settings. It will take some time to install and set up the EC2 machine, once done it will run the instance.

1. Now, the Ec2 machine ‘VPN\_server’ is up and running, but we need to assign it a public IP so that we can access it from anywhere. We need an elastic IP to associate with ‘VPN\_server’, so that the IP wouldn't change after a restart of the machine.

Follow these steps to do so-

1. From the left pane of EC2 dashboard.
2. Go to Elastic IP under Network & Security.
3. Create Elastic IP.
4. Associate that Elastic IP to ‘VPN\_server’

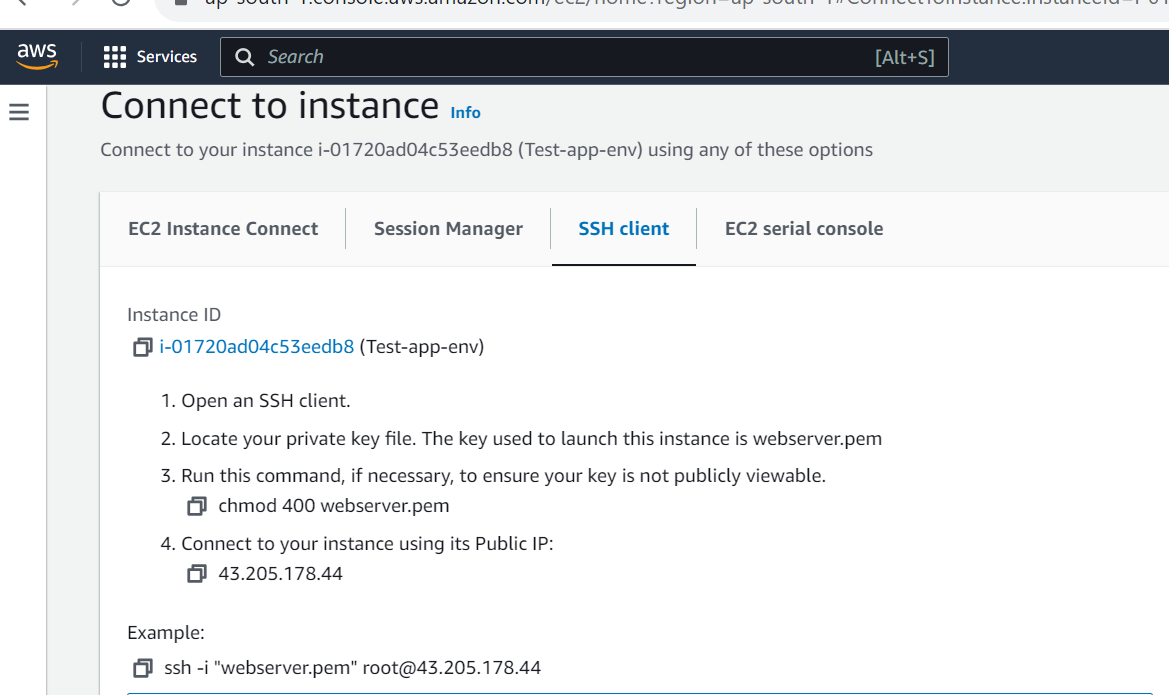
**Installing and settings up ‘OpenVPN Server’ :**

Till now, we have created an EC2 instance within our VPN’s public subnet, which is already running i.e., stage 1. Now we will move to our second stage, which is installing an open VPN server and configuring server-side settings to it. The steps involved in stage 2 are as follows-

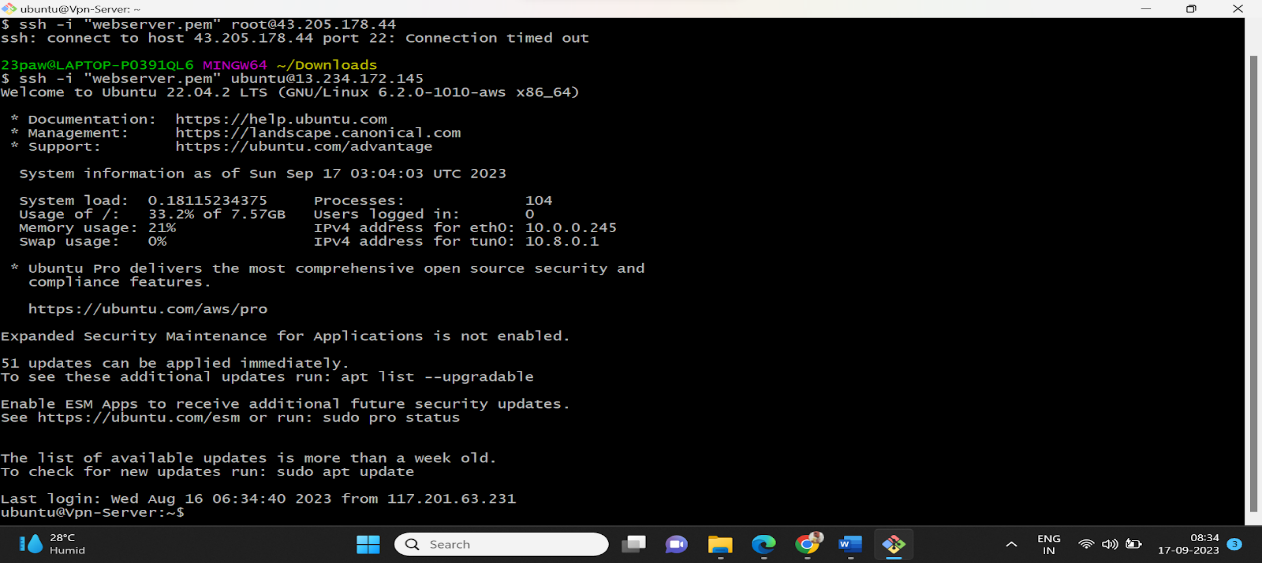
1. Access the instance ‘VPN\_Server’ through ssh from your local machine.

Note- you can get the details from your AWS console ->

* EC2
* running instances
* select the instance in our case ‘VPN\_Server’
* click on ‘connect’ from top right.
* Then select SSH Client tab



Now copy the ssh command and paste it on your local machine cmd and hit enter.



Now we are connected with our AWS machine. So we will move to our next stage to install Open VPN server on the machine and configure that to serve our purpose. Most of the configuration and settings will do on the Aws machine for which I will use ‘# Server Side’ and for local machine ‘# Local Machine’. So let’s start-

1. **Modify the Hostname**

You might also want to update the hostname to be something more readable than the default assigned by AWS. Use the following command to edit the hostname file and change it to something useful.

# Server Side

sudo nano /etc/hostname

I have given the name ‘test-vpn’. Save and exit, Ctrl+x followed by y and hit Enter. Now we need to reboot the machine by following command-

# Server Side

#sudo reboot

1. **Installing OpenVPN and Easy RSA**

We need to download and install following software-

1. OpenVPN server
2. Easy-RSA for generating certification and keys.
3. net-tools for ease of identifying our IP address and default interface.

So, for installing all these tools we will use following commands-

# Server Side

sudo apt-get update

sudo apt-get install openvpn easy-rsa net-tools

1. **Configuring the OpenVPN Server**

OpenVPN comes with some example files for us to build on, which is better to start completely from scratch.

# Server Side

cd /usr/share/doc/openvpn/examples/sample-config-files/

Now we are gone to create our server configuration, with the help of sample ‘server.conf’ file provided by OpenVPN. So just copy and paste the file at ‘/etc/OpenVpn/’ directory by following commands –

# Server Side

sudo gunzip -c server.conf.gz > ~/server.conf

sudo cp ~/server.conf /etc/openvpn/server.conf

Note- ‘gunzip’ command is to unzip the zipped file, if it is unzip already ignore that commands and directly move to ‘cp’ commands.

Now, we need to configure our server configuration file. So lets move to the ‘/etc/openvpn/’ and create a dir to store all certificates and keys for server side.

# Server Side

cd /etc/openvpn/

sudo mkdir server

sudo nano server.conf

Our server.cof file will open in nano editor and we see it full of comments as we copied it from samples. So, we need to edit some lines and uncomment a few to configure our server as per our need.

The very first thing is we need to add a prefix dir ‘server/’ to server.crt, server.key, ca.crt dh .pem as following-

# Any X509 key management system can be used.

# OpenVPN can also use a PKCS #12 formatted key file

# (see "pkcs12" directive in man page).

ca server/ca.crt

cert server/server.crt

key server/server.key  # This file should be kept secret

# Diffie hellman parameters.

# Generate your own with:

#   openssl dhparam -out dh2048.pem 2048

dh server/dh.pem

We want to be sure that all our client’s traffic goes through the OpenVPN server, so uncomment ‘**push “redirect-gateway def1 bypass-dhcp” ’** as below-

# If enabled, this directive will configure

# all clients to redirect their default

# network gateway through the VPN, causing

# all IP traffic such as web browsing and

# and DNS lookups to go through the VPN

# (The OpenVPN server machine may need to NAT

# or bridge the TUN/TAP interface to the internet

# in order for this to work properly).

push "redirect-gateway def1 bypass-dhcp”

We want client’s DNS requests to use the VPN tunnel, so they can’t be monitored. Uncomment either one of or both:

‘ push “dhcp-option DNS 208.67.222.222” ‘

‘ push “dhcp-option DNS 208.67.220.220” ‘

We can optionally change the DNS to use another provider as well. I have chosen Google’s i.e ‘8.8.8.8’ here-

# Certain Windows-specific network settings

# can be pushed to clients, such as DNS

# or WINS server addresses.  CAVEAT:

# http://openvpn.net/faq.html#dhcpcaveats

# The addresses below refer to the public

# DNS servers provided by opendns.com.

push "dhcp-option DNS 8.8.8.8"

;push "dhcp-option DNS 208.67.220.220"

Now to run OpenVPN server with safe permission uncomment or set ( user nobody and group nogroup) following –

# It's a good idea to reduce the OpenVPN

# daemon's privileges after initialization.

#

# You can uncomment this out on

# non-Windows systems.

user nobody

group nogroup

Now, we are almost done with our server configuration, so we need to save the server.conf file by CTRL + x and press y to confirm and hit enter.

1. **Packet Forwarding and Firewall-**

We need to turn on packet forwarding by following steps-

# Server Side

sudo nano /proc/sys/net/ipv4/ip\_forward

In this file, we need to change 0 to 1. Now save and exit from the editor by by CTRL + x and press y to confirm and hit enter.

Now, we’ll enable IP packet forwarding on boot by editing ‘sysctl.cnf’ file. So lets do it-

# Server Side

sudo nano /etc/sysctl.conf

uncomment following  line-

# Uncomment the next line to enable packet forwarding for IPv4

net.ipv4.ip\_forward=1

Now, save and exit the file by CTRL + x and press y to confirm and hit enter.

Moving to next, let’s check the firewall ufw(Universal Firewall) status i.e. if it is installed and active.

# Server Side

ufw –version

If we get the result of above command, same as below it means it is installed.

ufw 0.36

Copyright 2008-2015 Canonical Ltd.

Now, we check if it is active or not-

# Server Side

sudo ufw status

Status: inactive

It says, inactive. OK then, let’s set some rules. These will be the same as what we did in the AWS security group. You could argue that this step with **ufw** isn’t necessary and that we can just lean on AWS, but it’s quick for me to show here and you may not be using AWS to provide your compute.

# Server Side

sudo ufw allow ssh

sudo ufw allow 1194/udp

# Server Side

sudo nano /etc/default/ufw

In this file, we need to change “DEFAULT\_FORWARD\_POLICY” from “DROP” to “ACCEPT” as below-

# Set the default forward policy to ACCEPT, DROP or REJECT.  Please note that

# if you change this you will most likely want to adjust your rules

DEFAULT\_FORWARD\_POLICY="ACCEPT"

Now, save and exit the file by CTRL + x and press y to confirm and hit enter.

For the next step we need to identify the interface we are using for the internet.

# Server Side

Ifconfig

The return should look something like below, we can see the local loop back **lo** to home (127.0.0.1) and our internet facing interface **eth0** on 172.31.86.140.

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001

        inet 172.31.86.140  netmask 255.255.240.0  broadcast 172.31.95.255

        inet6 fe80::10e7:f8ff:fe89:49bb  prefixlen 64  scopeid 0x20<link>

        ether 12:e7:f8:89:49:bb  txqueuelen 1000  (Ethernet)

        RX packets 190651  bytes 269842353 (269.8 MB)

        RX errors 0  dropped 0  overruns 0  frame 0

        TX packets 28300  bytes 2598273 (2.5 MB)

        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536

        inet 127.0.0.1  netmask 255.0.0.0

        inet6 ::1  prefixlen 128  scopeid 0x10<host>

        loop  txqueuelen 1000  (Local Loopback)

        RX packets 282  bytes 26634 (26.6 KB)

        RX errors 0  dropped 0  overruns 0  frame 0

        TX packets 282  bytes 26634 (26.6 KB)

        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Now we need to add that to the **ufw** before.rules config file.

# Server Side

sudo nano /etc/ufw/before.rules

Add the following somewhere near the top of the file:

**\*nat:**

**POSTROUTING ACCEPT [0.0]**

**-A POSTROUTING -s 10.8.0.0/8 -o eth0 -j MASQUERADE**

**COMMIT**

#

# rules.before

#

# Rules that should be run before the ufw command line added rules. Custom

# rules should be added to one of these chains:

#   ufw-before-input

#   ufw-before-output

#   ufw-before-forward

#

\*nat

:POSTROUTING ACCEPT [0.0]

-A POSTROUTING -s 10.8.0.0/8 -o eth0 -j MASQUERADE

COMMIT

Now, save and exit the file by CTRL + x and press y to confirm and hit enter. Now we can enable ufw.

# Server Side

sudo ufw enable

Press y and hit enter.

If we are still have an active ssh connection to our ‘VPN\_server’, it means we are good to go. Better we check the status of ufw-

# Server Side

sudo ufw status

Status: active

To                         Action      From

--                         ------      ----

22/tcp                     ALLOW       Anywhere

1194/udp                   ALLOW       Anywhere

22/tcp (v6)                ALLOW       Anywhere (v6)

1194/udp (v6)              ALLOW       Anywhere (v6)

Now, we are done enabling IP packet forwarding and Firewall. Let's move to the next stage.

**Setting up the Server Side keys and certificates-**

We are going to take all the build scripts that installed with Easy RSA and put them into the same directory as OpenVPN for convenience.

# Server Side

sudo cp -r /usr/share/easy-rsa /etc/openvpn

We need to do some setup in the Easy RSA directory. Namely creating a vars file from the provided example. This will store our default values for a range of variables that are asked for when using Easy RSA.

# Server Side

sudo cp /etc/openvpn/easy-rsa/vars.example /etc/openvpn/easy-rsa/vars

sudo nano /etc/openvpn/easy-rsa/vars

Edit Country, Province, City, Org, Email and OU as required.

# Organizational fields (used with 'org' mode and ignored in 'cn\_only' mode.)

# These are the default values for fields which will be placed in the

# certificate.  Don't leave any of these fields blank, although interactively

# you may omit any specific field by typing the "." symbol (not valid for

# email.)

set\_var EASYRSA\_REQ\_COUNTRY     "US"

set\_var EASYRSA\_REQ\_PROVINCE    "California"

set\_var EASYRSA\_REQ\_CITY        "San Francisco"

set\_var EASYRSA\_REQ\_ORG         "Copyleft Certificate Co"

set\_var EASYRSA\_REQ\_EMAIL       "me@example.net"

set\_var EASYRSA\_REQ\_OU          "My Organizational Unit"

Now, save and exit the file by CTRL + x and press y to confirm and hit enter.

Now, we will generate all certs and keys for the server.

# Server Side

cd /etc/openvpn/easy-rsa

sudo ./easyrsa clean-all

sudo ./easyrsa init-pki

sudo ./easyrsa build-ca server nopass

I will use ‘server’  and ‘client1’ for U**NIQUE\_SERVER\_SHORT\_NAME** and **UNIQUE\_CLIENT\_SHORT\_NAME**

# Server Side

# replace your req filename with UNIQUE\_SERVER\_SHORT\_NAME for me it is ‘server’.

sudo ./easyrsa gen-req server nopass

sudo ./easyrsa sign-req server server

Press Y and hit Enter. Now we can verify our certificates.

# Server Side

sudo openssl verify -CAfile pki/ca.crt pki/issued/server.crt

We also need to generate a 2048 bit Diffie-Hellman pem file.

# Server Side

sudo ./easyrsa gen-dh

Next we need to move all these generated files somewhere that OpenVPN can get at them. Note that for me, **“server”** was my **UNIQUE\_SHORT\_FILE\_NAME** and the commands shown here reflect that in the filenames but not in the OpenVPN sub-directory.

# Server Side

sudo cp pki/ca.crt /etc/openvpn/server/.

sudo cp pki/issued/server.crt /etc/openvpn/server/.

sudo cp pki/private/server.key /etc/openvpn/server/.

sudo cp pki/dh.pem /etc/openvpn/server/.

We are also going to use TLS for additional security and this means we need an additional key.

# Server Side

cd /etc/openvpn

sudo openvpn --genkey --secret ta.key

Now we can start the server and check it’s status.

# Server Side

service openvpn start

service openvpn status

Looks good.

● openvpn.service - OpenVPN service

Loaded: loaded (/lib/systemd/system/openvpn.service; enabled; vendor preset: enabled)

Active: active (exited) since Sat 2022-03-12 13:19:56 UTC; 19min ago

Process: 555331 ExecStart=/bin/true (code=exited, status=0/SUCCESS)

Main PID: 555331 (code=exited, status=0/SUCCESS)

For more information and debugging we can check the main log file created by the process.

# Server Side

sudo watch tail /var/log/openvpn/openvpn.log

Looks as though we are up and running, now lets provision a client and get connected.

Sat Mar 12 13:19:58 2022 Socket Buffers: R=[212992->212992] S=[212992->212992]

Sat Mar 12 13:19:58 2022 UDPv4 link local (bound): [AF\_INET][undef]:1194

Sat Mar 12 13:19:58 2022 UDPv4 link remote: [AF\_UNSPEC]

Sat Mar 12 13:19:58 2022 GID set to nogroup

Sat Mar 12 13:19:58 2022 UID set to nobody

Sat Mar 12 13:19:58 2022 MULTI: multi\_init called, r=256 v=256

Sat Mar 12 13:19:58 2022 IFCONFIG POOL: base=10.8.0.4 size=62, ipv6=0

Sat Mar 12 13:19:58 2022 IFCONFIG POOL LIST

Sat Mar 12 13:19:58 2022 Initialization Sequence Completed

**Generating Client Keys-**

Clients also need keys and certificates generated. In the command set below change **UNIQUE\_CLIENT\_SHORT\_NAME** to the name of your client, for me ‘client1’.

# Server Side

cd /etc/openvpn/easy-rsa

sudo ./easyrsa gen-req client1 nopass

sudo ./easyrsa sign-req client client1

Press Y and hit enter.

We can now verify our generated and signed certificates using **openssl**.

# Server Side

sudo openssl verify -CAfile pki/ca.crt pki/issued/client1.crt

Now we need to pass over these certificates and keys over to OpenVPN.

# Server Side

sudo cp pki/ca.crt /etc/openvpn/client/.

sudo cp pki/issued/client1.crt /etc/openvpn/client/.

sudo cp pki/private/client1.key /etc/openvpn/client/.

So we are done with generating and registering keys and certs for servers and clients.But we need all these keys and certs for client at client side as well, this is why we need a .ovpn file to connect the VPN server with our local VPN client. So, our next step is to create a ‘.ovpn’ file for the client side.

**Creating .ovpn file for Client-**

We can use an OpenVPN example file as the basis of .ovpn file for us. We’ll copy this example and edit it as a template to re-use for whenever we choose to make subsequent clients.

# Server Side

cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf ~/template.ovpn

The clients will need to know the server’s public IP, we can get the IP from the AWS management console (or from the note I’m sure you made earlier), it’s the Elastic IP that we assigned to our instance. We need to add this and change some other settings in our **template.ovpn** file.

# Server Side

cd ~

nano template.ovpn

Find remote and replace my-server-1 with the public IP address of your server. For me this ends up looking like this.

# The hostname/IP and port of the server.

# You can have multiple remote entries

# to load balance between the servers.

remote 34.193.47.15 1194

;remote my-server-2 1194

Remove the leading semi-colon to uncomment out “user nobody” and “group nogroup” to allow operation with downgraded priveleges.

# Downgrade privileges after initialization (non-Windows only)

user nobody

group nogroup

We also need to comment out the directives that point to the ca, cert and key, as we are going to add those to this file.

# SSL/TLS parms.

# See the server config file for more

# description. It's best to use

# a separate .crt/.key file pair

# for each client. A single ca

# file can be used for all clients.

;ca ca.crt

;cert client.crt

;key client.key

It’s the same story for the directive that points to the tls-auth key, again we are going to put this key in to this file

# If a tls-auth key is used on the server

# then every client must also have the key.

;tls-auth ta.key 1

Now, save and exit the file by CTRL + x and press y to confirm and hit enter.

Let’s create a directory within our home directory and give it the same name as our client, which for me was “client”. This will be where we store copies of our client’s keys and certificates.

Let’s make the directory for this specific client, in our home directory. My client is called “client” so I’ve called my directory client, I’d advise you use whatever your client’s name is here.

# Server Side

mkdir ~/client

Now we need to fill it with all the relevant files for our client to connect. Finally we’ll use the ls command inside the directory and just check that we’ve got everything together.

# Server Side

cd /etc/openvpn/client

sudo cp ca.crt client.crt client.key ~/client

cd ~/client

sudo cp /etc/openvpn/ta.key ~/client

cp ~/template.ovpn ~/client/client1.ovpn

ls

Looks like we do.

ca.crt client.crt client.key client1.ovpn ta.key

Now we can append each of these other files to our .ovpn file, along with some ML style declarations.

For the CA certificate.

# Server Side

echo "<ca>" >> client1.ovpn

sudo cat ca.crt >> client1.ovpn

echo "</ca>" >> client1.ovpn

For the client certificate.

# Server Side

echo "<cert>" >> client1.ovpn

sudo cat client1.crt >> client1.ovpn

echo "</cert>" >> client1.ovpn

For the client key.

# Server Side

echo "<key>" >> client1.ovpn

sudo cat client1.key >> client1.ovpn

echo "</key>" >> client1.ovpn

For the TLS key. Note that this also requires an additional directive to set the key direction.

# Server Side

echo "key-direction 1" >> client1.ovpn

echo "<tls-auth>" >> client1.ovpn

sudo cat ta.key >> client1.ovpn

echo "</tls-auth>" >> client1.ovpn

Client Setup

Downloading .Ovpn file to local machine-

Here I’m assuming your client is also a Linux based system running Network Manager, but there some alternatives mentioned at the end of this section.

With terminal access on the client machine, securely copy client.ovpn to your client device. I’d recommend using scp to copy the file, replacing “user” for your user name on the sever and your server’s public IP for “serverIP”.

# Client Side

scp -i "my.pem" user@serverIP:client/client.ovpn .

The above command will download the client1.ovpn file to our local machine at the current location. This file will be used for openvpn network manager or client side application to connect the VPN server.

-----------------------        Thanks & Regards       --------------------------