OpenSSH

opeSSH by default use port 22 to connect with server. It need the Ip Address to connect with the server with local machine.

CONNECTING TO REMOTE SERVER BY SSH

Ssh client only allow you to connect with other machine rather let people connect with your local machine.
Ssh username@ Ip address will let you connect with the server as long as the server is running openSSH sever components and server allow incoming traffic via port 22 we should manage to connect.
When we connect with server using ssh it always keep the fingerprint on local machine in .ssh/ known_host file.

How we connect with the server we can see that in /var/log directory and auth.log file. Sometimes if we have any issue to connect with the server we can see that in server auth log file.

How the server knew that I am allowed to connect. There are various ways of like password based authentication or key based authentication. Password based authentication, it will ask for the password if we want to connect with the server and the server already knew the password as when we are creating the server we are giving username, password and it is storing the pass and match it when we want to connect remotely.

Key based auth, we need to generate key in our local machine and put the public key in the authorised key section on the server.

CONFIGURING THE OPENSSH CLIENT

Remembering an ip address is always tough, so make life easier we can configure the ssh file so that we can login to the remote server very easily. Real world we are using DNS but still we can make it simple by configuring openSSH client.

First we need to create file in .ssh directory name config!
Then we need to modify the file. Here is simple example bellow how we can do that.
Host myServer
    Hostname 172.0.0.1 (IP address of the server that I want to connect)
    Port 22 (which is default port for connecting the server from local machine using ssh)
    User ubuntu (is the name of the server)

What happened here is, as the file is ssh directory what ssh client will see there is a file exists and it will gonna load it to the memory. What it basically does is it will going to have is an entry for myServer with ip and other related address.

We can have multiple entry of different server following the same pattern.

USING PUBLIC/PRIVATE KEYS

Password based connection with server is not secure. As password can be guess or break by hacker so it is not wise to use password base login into a server or vm!
So key based login is better and secure but there is downfall with that as if we don't have the key it will still ask for the password and allow password based login which is not good. We can turn it off, so it is good option to do.
How to create ssh key! Its very simple! We can use another binary named ssh keygen.
We need to type ssh-keygen

It will gonna ask for where to save the file in the .ssh directory and file name id_rsa. We need to be very careful about that as it is important not have any other id_rsa file in the same directory. If it has it will gonna rewrite the existing file with new one and if we used the previous key to access any server by using the previous key we can not login again.
Then comes the pass phrase, which is another layer of the security. If someone has the key but don't know the passphrase they cant manage to login the server. It is better to use pass phrase. Then it will generate two keys one private and one public.

Public key should be store in the server authorised key file. Then we can login to the server without using any public key.
If we type ssh -v user@ip address it will gonna show all thing how it connect with the server.

The easiest way to send local machine public key to the server is using command rather copy and paste of public key.

Ssh-copy-id -I ./.ssh/id_rsa.pub ubuntu@198.27.28.9
Here ssh-copy-id is the dedicated command for moving ssh key to the server and -I is using for input files next we use the path of the files and finally we give server name and ip address.

MANAGING SSH KEYS

It is very common that we can use our local machine to login multiple server and if we use the same public key for every server, it is very dangerous if the key fall to some hackers hand.
It is wise to use separate key for separate server.

But there is an issue if .ssh directory has lot of keys it will check everyone and try to match it but if there are too many keys there is a chance getting logged out of the server.

When we generate keys for individual server we can use different type of keys like ed25519. These are more secure then typical rsa key. The command for generating these type of key is
Ssh-keygen -t ed25519 -C "name of the server that is gonna used"

It is always good option to rename the key file name to specific server so that easily recognisable.
Next question ? We have different key for different server. How we use the specific key to login specific server?
Yes we can login using the specific key. The command for this is ssh -I ~/.ssh/name of the private key file then username@ipaddress

SSH AGENT

Though we are using different key for log into different server if we use passphrase every time
We need to give the pass phrase for different server which is a hassle? How to get rid of that. The answer is ssh agent. Ssh agent will help us to cache the key to the memory. Remember this will help til the terminal is open. If its closed we need to save it to the cache again.

GUI based linux distros usually has ssh agent running but non gui based server didnot have that so to check either ssh agent running or not we can use the following command:
ps aux | grep ssh-agent

If its not running we can use that following command to start ssh agent:
EVAL $(SSH-AGENT -S)

While ssh agent is running, we can add our private key for specific server to cache memory.
The command for it is: ssh-add ~/.ssh/name of the private key

Next time, if we want to login to the server we will not ask for passphrase any more until I close the terminal.

CONFIGURE THE SERVER COMPONENT OF OPENSSH

We have so far learn all about client side SSH. What about server side ssh? Most linux distros has sshd (SSH Demon) in build and they are running. We can check it by - which sshd

To check wether it is running or not we can use - systemctl status sshd. The thing is if its not running we are not able to connect with the server.

Systemctl is a command line utility which is based on systemd. It is a default init system. Systemd is a system and service manager. With that we can start/stop/ enable/ disable/status/restart service.
To enable ssh we can use - systemctl enable ssh command.

When we login to server side where our key is stored? If not how they match it?
Yah they are stored in cd/etc/ssh directory ! There we can see all the fingerprints public and private key stored there. They are very Important because one simple reason if we delete that we cant login the server any more.

Here one thing need to remind is that, when we want build an image of server that I want to deploy or clone into other machine all the related ssh key also went with it. In that case we need to change the ssh key and give new one. If not local machine will get confused.

On that ssh directory we can see two config file one is ssh config and another one is sshd config. Ssh config is actually client side config file which we can create. Previously we saw that how to create it.

Sshd config file is the server side config file. This file will listen for server service. We can nano that file we can see there are lot of settings what we can change.
For example we can change the port number, as default ssh always listen for port 22 but we can change it.
One more option is permit root login. As usually if you are single user you can use it. If you are not single user and there are more user it is very safe to made this root permit login to NO.

Another one is password based auth! Which we always need to set as no. because if we do not set it no there is possible chance of hacking is always there. So after setting up ssh key auth next step should disable password based auth.

If we change port we need to use command like : ssh -p 2222 user@172.1.1.1