Firewall & SSH

- Firewall-Hostlevel, Network level
- Why we want to gain access to the remote shell of a computer over the network?
- How does SSH works? what makes the SSH secured?
- How does encryption works?
    - symmetric encryption
    - asymmertic encryption
    - hashing
- How does the SSH Connection would be established with a remote server?
- How does SSH works?
- How to setup ssh access for a linux machine?
- How to ssh onto the remote machine?
- How to configure passwordless authentication to the remote server?
- How to add public key to a user allowing the ssh access?
- How to copy the generated public key on to the remote user authorized_keys file?
- How to copy the public key onto the remote user?

==========**********==========

Firewalls are used for enforcing traffic restrictions allowing your software programs to be accessible over the network. Most of the operating systems will ship firewall software as part of their install

The Firewalls can be scoped at 2 places

1. Host level

In a personal computer or single user systems we use host-level firewall for protecting the applications not to be accessed from the outside world, unless permitted

2. Network level

Within an organization, instead of configuring the firewall at the individual host level which is very difficult, we can setup on firewall at the network level which will protect any of the computers connected to the network not to be exposed or accessed from outside world.

The Firewalls can be either a software applications or can be hardware devices as well.
In an enterprise organization, people setup hardware firewalls for more performance and network through put than s/w firewalls

We apply traffic restrictions on the software applications to be accessed by the others by configuring various different rules like
1. Allow any traffic to a program running on a specific port
2. Allow all http traffic
etc

Ubuntu operating system provides iptables as a default firewall software as part of its distribution, through which we can manage/apply traffic restrictions. working with iptables requires great amount of knowledge on networking and protocols concepts, so many of the people often find very complex to work with iptables.

To help people to manage and configuring iptables/firewall in ubuntu operating system, the ubuntu has distributed a software called **ufw**. ufw stands for "**un-complicated firewall**". it acts as an interface through which we can manage the ubuntu firewall system.

By default in ubuntu 18.x operating system ufw is installed and distributed. if you are using previous versions of ubuntu operating system then you need to manually install ufw software package
sudo apt install ufw

#1. by default ubuntu firewall is disabled, so no traffic restrictions are applied to the programs running on your computer.
To enforce traffic restrictions we need to enable the firewall software
sudo ufw enable

when we enable firewall or ufw above the default behaviour is
1. deny all the inbound traffic
2. allow all the outbound traffic

#2. How to know the status of the firewall or ufw?
sudo ufw status

We can enforce traffic restrictions at 3 levels
1. source ip address
2. portno
3. protocol

#3 how to see what traffic restrictions are enforced through the firewall
sudo ufw status verbose

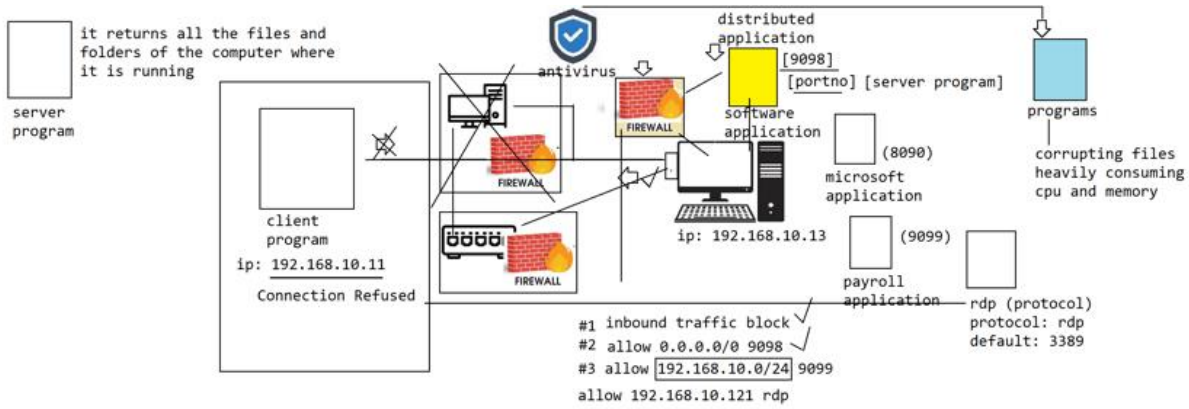
#4. how to allow/deny traffic to the programs from a specific sourceip address
sudo ufw allow from sourceip
sudo ufw deny from sourceip

#5. how to allow/deny traffic restrictions based on protocol
sudo ufw allow protocol
sudo ufw deny protocol
eg..
sudo ufw allow http

#6. how to allow/deny access based on portno
sudo ufw allow portno
sudo ufw deny portno

#7. how to allow/deny traffic to a range of portno
sudo ufw allow 8080:8088/tcp

#8. How to see the list of rules with numbers applied
sudo ufw status numbered



By default on ubuntu machines the ufw is disabled, so no traffic restrictions are applied.
allowing/denying traffic from a sourceip address
sudo ufw allow/deny from sourceip

allowing/denying traffic from a protocol
sudo ufw allow/deny protocol

sudo ufw delete rule#

SSH (Secure Shell)

SSH Stands for secure shell, it is used for gaining the access to the shell of a remote computer.

- ✓ **Why we want to gain access to the remote shell of a computer over the network?**

In an organization they can be lot of computer being setup and used by the associates/employees of the organization. The system administrator has to monitor, install or troubleshoot these computers used by the associates.
The administrator inorder to manage these computer has to go physically and visit each computer one over the other to troubleshoot or do any activity on them. It would be very difficult to reach physically to a machine to carry the operation and is an time consuming job

Sometimes we wanted an administrator to control the machines over a different goegraphical position as well
only way to achieve remote access to a computer without appearing physically at the machine is through SSH.
SSH is widely used in gaining access to the cloud machines of a cloud provider.

Before SSH there used to be a protocol called telnet through which we can gain access to the remote computer shell. but the problem with telnet is all the information exchanged between the client/server computer are plain/text and anyone can steal the information.

telnet is not secured in exchanging the data between the computers and no more being used replacing telnet SSH has been introduced

SSH is used for:
accessing the remote computer shell in a secured way, so that we can perform administrative activities on the remote computer like installing/uninstalling software, manage configurations, controlling the process etc can be done on the remote computer

- ✓ **How does SSH works? what makes the SSH secured?**

SSH to ensure the information exchanged between the client/server computer are secured it uses cryptographic technics
one form of cryptography is encryption/decryption, encryption/decryption are used for writing the information in another format so that people cannot understand it. SSH protocol uses ecryption/decryption for exchanging the data secured between 2 parties

- ✓ **How does encryption works?**

Encryption and decryption works based on an encryption key. using the encryption key we encrypt/transform the data into another format so that no one can understand the data.
SSH mainly works based on encryption key, the better the key is the information will be more secured. So always consider the key to be as strong key so that the key itself cannot be cracked first-in place so that no one can decrypt it other than the person who poses it.

To ensure the encryptions keys are strong enough so that people cannot break the data, there are key generation algorithms are available, which will derive the keys based on some mathematical calculator which are seems to be unique and strong to be cracked
There are many key generation algorithms are available
1. RSA
2. SHA
3. AES
4. DES
etc

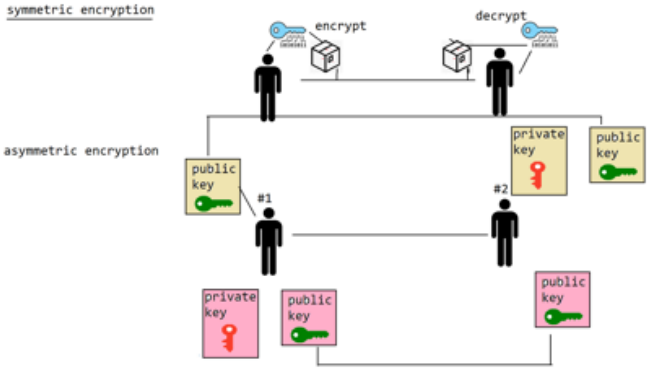
SSH uses 3 types of encryption technics in securely exchanging the data between the computers.
1. symmetric encryption
A single encryption key is used for encrypting/decrypting the data between both the parties for exchanging the data
2. assymetric encryption
There are 2 different keys are used for encryption/decryption
1. public key = The public key is used for encrypting the data
2. private key = The data that is encrypted using public key can be decryptable only by using the private key
As there are 2 keys being used it is called assymetric encryption

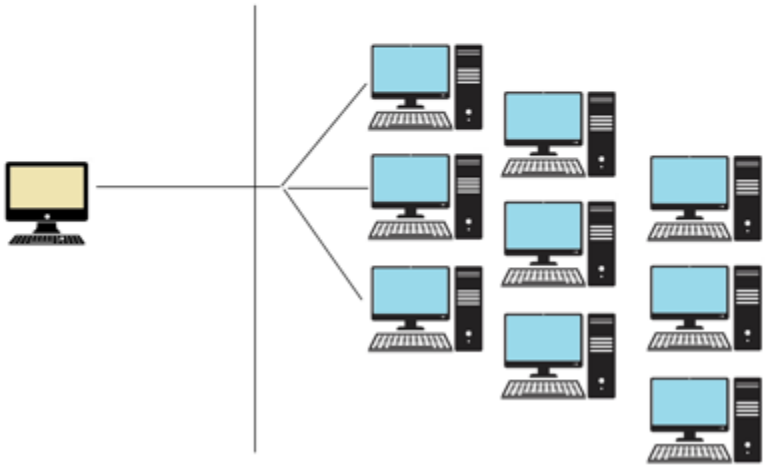
The key generation algorithms uses mathematical technics in generating the public/private key pair, such a way
#1. from one key we cannot derive the another one
public key, we cannot compute the private key for that public key
#2. The data encrypted using public key can be decryptable only through using the private key
To ensure the security always the private key should be kept private.

3. hashing

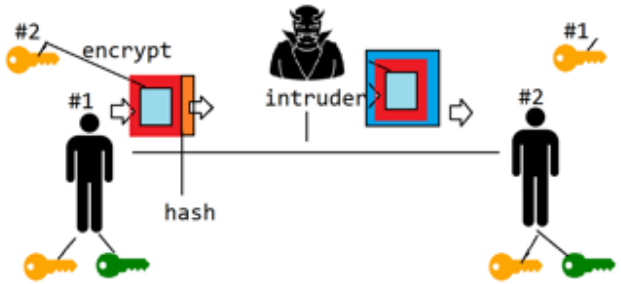
hashing is a technic that is used for verifying the integrity of the data.
The hash is a special mathematical function where always an
hash(data) = unique value
always hash(samedata) = results in same unique value. but by using hash value we can compute original data
so we can easily identify at the received end, whether original message that is send by the sender has been tampered or not by recomputing the hash value on received data matching with original hashvalue

The SSH works on 2 stage process
2 Computers will establish SSH connection/communication in 2 stages.
#1. Establishing and exchanging a session key to exchange information between the computers in a encrypted/secured form
#2. Authenticating to gain access to the remote computer



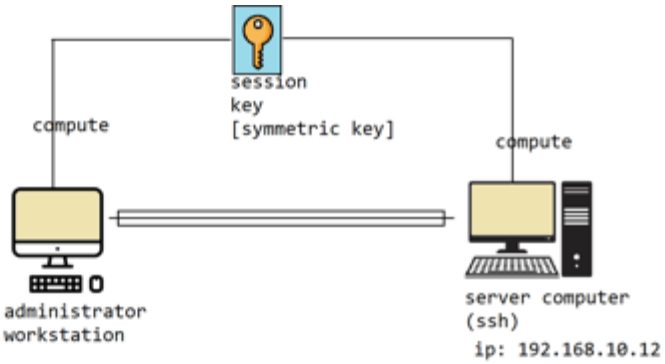


asymmetric encryption



hashing is a function
a = hfx(a) = 1001
hfx(a) = 1001
hfx(1001) = ~~a~~

#1



#2

- ✓ **How does SSH works?**

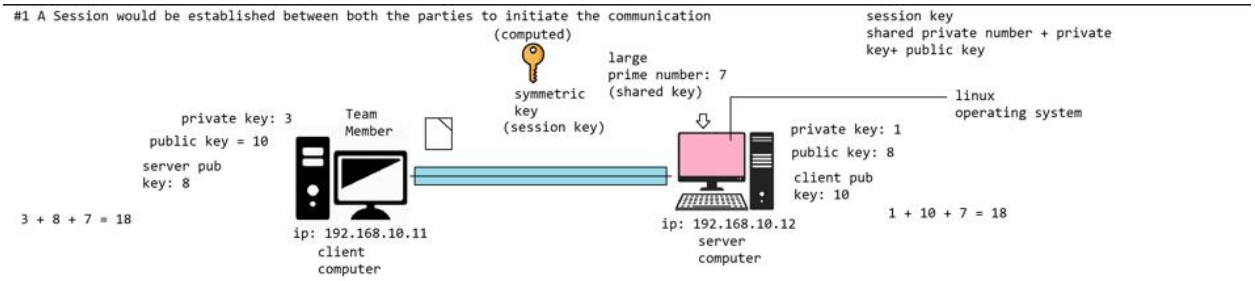
SSH communication takes place between the computers in 2 stages
1. establishing the session by generating an session key for exchanging the data
2. authenticating the remote user in granting the access to the remote machine

- ✓ **What is SSH, why do we use it?**

SSH stands for Secure Shell, it is used for accessing the remote shell of a linux computer securely, so that we can perform administrative activities like install/uninstalling software packages, configuration management etc on the machines remotely

- ✓ **How does the SSH Connection would be established with a remote server?**

There are 2 stages in which an SSH connection would be established to an remote server
1. A session would be established between both the parties to communicate with each other
2. authenticate the remote/client computer to authorize access to the server



How does the session would be established between both the parties in initiate the communication?
#1. The client computer sends an request to the server asking for starting an SSH session for communicating with each other, then the server would accept the request

During the session establishment, both parties agrees upon an symmetric key to make the communication secured. The key would not be generated by one part and will not be shared with other, since during the key exchange process anyone can steal the key and makes the communication insecured
Both the parties will compute the symmetric key individual and arrives to the same key based on a sequence of operations.
#1. Both the parties agree on large prime number, which will serve as a seed value
#2. Both the parties agree on an encryption algorithm like AES, RSA etc which will be used for encrypting the data
#3. Independently, each party comes up with another prime number which is kept secret from the other party (we can consider it as a private key)
#4. generated private key, encryption algorithm and shared prime number are used to generate a public key that is derived from the private key
#5. both the public keys are exchanged by the parties
#6. upon receiving the public key of others
    1. by using the private prime number (own private key)
    2. public key of other person
    3. shared prime number
    generate a session key, where both the parties would endup in computing the same session key through this process.
#7. by using this session key both parties would encrypt the data and communicate with each other.

#2.
upon establishing the session by both the parties, the server challenges the client for authentication

The client inorder to establish the communication, he has to send the username/password of the user using which he wanted to connect the server.
This username/password would be encrypted using shared session key (symmetric key) and would be send to the server.
The server upon receiving the credentails would decrypt them and validate the username/password provided is valid or not and allows the client to communicate with the server and perform operation on behalf of the user

The above technic of authenticating the client with the server is called "passwordAuthentication" technic.
With the above technic of authenticating the client with the server and gaining the access has few dis-advantages:-

1. exchanging the username/password over the network is a security breach,because no matter whether we encrypted or not, still there is a chance someone might break it
2. by allowing the client to connect to the server using username/password will make the client gain permanent access to our computer.
if we want to revoke the access to the client after certain amount of time, it would be possible or may lead to difficulties.
How to over the above problem with passwordAuthentication.

That is where we need to use Asymmetric keys for authentication.

- ✓ **How does SSH works?**

The SSH connection would be established between the client and the server in 2 stages
1. The session will be established between both the parties by computing a shared session key for end-to-end encryption
2. authenticating the client to allow access to the server machine

#1. The session will be established between both the parties by computing an shared session key
The client sends the request to the server initiate the session, upon the server accepts the requests, the below process would begin to compute a shared session key for encryption
1. both parties will agree upon and generates an large prime number which is an shared key
2. both parties agrees upon an encryption algorithm to encrypt the data
3. each party independently generates an prime number which is kept as an private key
4. each party generates independently their own public key with a combination of shared key, private key and algorithm
5. exchange their public keys with each other
6. upon exchanging the public key, compute a session key with either of their public keys with their private key and shared key which will results in the same session key by both the parties.
7. use the computed session key for encrypting the data during the communication

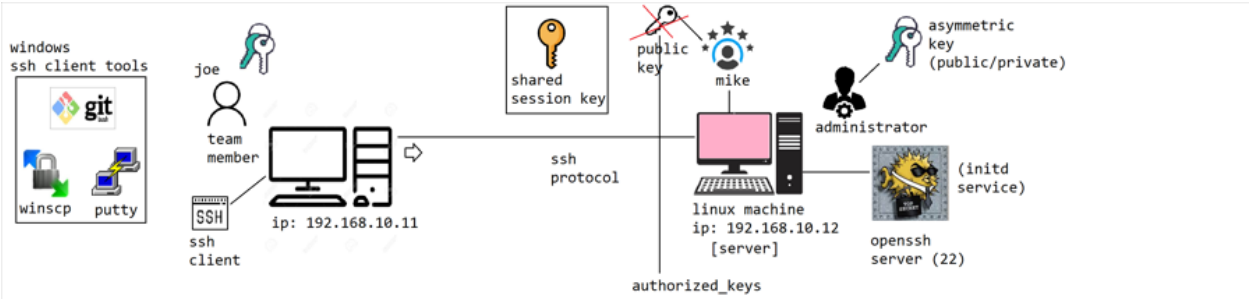
#2. upon generating the shared session key, the server challenges the client to authenticate himself in gaining the access to the server computer
1. The client has to send the username/password onbehalf of the user for whom he wanted to access the remote server by encrypting the data with shared session key.
2. The server vaidates the username/password by decrypting with shared session key and grants the access to the client if the credentials are valid

This process of authenticating the user in granting the access to the remote server is called "passwordAuthentication"

There are problems with passwordAuthentication in gaining the access to the remote server
1. exchanging the username/password over the network is an security breach or even though we encrypt the username/password, there is still a chance someone might decrypt it

2. Granting the access to the remote machine to a client based on username/password becomes permanent access and we cannot revoke the access to the machine easily

To overcome the above problem asymmetric key-based authentication technic has been introduced.



upon establishing a shared session key between both the parties, the server challenges the client for authentication
#1. The client will send the username and the public key of the user onbehalf of whom he wanted to connect to the remote computer
#2. The server upon receiving the username/public key, he checks, whether the public key provided is in authorized_keys of the user (mike), if not server rejects the connection
#3. if the public key provided is in authorized_keys the server will generates an random text, encrypts that with the public key of the user and sends back to the client challenging him to prove whether the client poses the private key or not
#4. client upon receiving the ecrypted text, will decrypt it using private key, then he encrypts the decrypted with shared session key along with MD5 Hash combined together will send back to the server proving its authenticity
#5. The server upon receving the text from the client will decrypt using the shared session key and validates with the original, if matches grants the access to the client.
This is how the authentication takes place using private/public keys.

This technic of authenticate the client for granting the access to the remote server is called "passwordlessAuthentication"

- ✓ **How to setup ssh access for a linux machine?**

#1. ensure we have the ip address of the remote computer we wanted to ssh, and check for reachability of the remote computer
In our case we wanted to connect to the virtual machines running on virtualbox from hosted machine **(windows)**
To perform ssh on to the virtualmachine we need to setup network mode, here we have 3 options
#1. NAT with port forward: 22
#2. Bridge Network
#3. Host-only network adapter (no external network or internet)

So we can setup 2 things
1. NAT + Host-only network adapter

2. Bridge

#2. Install openssh server on the server computer
sudo apt update -y
sudo apt install -y openssh-server

#3. upon installing the openssh server, it creates an ssh server configuration file under /etc/ssh/sshd_config, it holds configuration settings in managing the ssh server
Few of the configuration settings available as part of the sshd_config file are as below.

2.1 the default port on which the ssh server runs is on: 22, which we can change by modifying the sshd_config file
port=22

2.2 The default authentication mechanism granting the client to access the server is "passwordAuthentication"
passwordAuthentication: yes

if we want to disable passwordAuthentication then we need set it to no and restart the ssh server
sudo systemctl restart ssh

From client machine:
Windows:-

- ✓ **How to ssh onto the remote machine?**

1. gitbash
    ssh username@ip

2. putty/winscp
open putty and enter ip address and click on connect
prompt for username/password and will be connected

3. windows powershell
by default has bash ssh client

- ✓ **How to configure passwordless authentication to the remote server?**

#1. never enable ssh access for the root user of the linux machine (best practise)
#2.
Let us add an new user joe
sudo adduser joe

#3. to enable keybased authentication we need to generate private/public key
there are many ways to do this
In windows:
#1
    1. we need to use puttygen utility or tool to generate private/public key
    2. generate the .pub and .ppk (.pub=public key), (.ppk=private key) and place them under windown $HOME/.ssh directory

Both Windows & Linux:

Windows: gitbash

Linux: bash shell utility

we need to run a command for generating the SSH keys

ssh-keygen

The above command prompts for keyfile name and generates .pub/.ppk and stores

#4. now add the public key on to the remote machine of the user for whom we wanted to allow ssh access

For eg.. now we wanted to enable ssh access for the "joe" user, then add the public_key we generated to joe on the server

- ✓ **How to add public key to a user allowing the ssh access?**

for each user under the $HOME directory we need to create an directory called .ssh

for eg.. for joe user under his home directory we need to create /home/joe/.ssh/

inside that we need to create an file called authorized_keys with permission to the file

now add the public key we generated at the client side into the authorized_key file of that user

ssh -i ~/.ssh/joekey joe@192.168.1.113

1. when we initiate the sshconnection with above command, the openssh server receives the request.

2. then it checks under joe user, inside authoried_keys is the public key the client is sending is available or not

- ✓ **How to copy the generated public key on to the remote user authorized_keys file?**

1. manually copy paste into the authorized_keys file

2. we need to have ssh access to the remote server either through password/passwordless with exiting key, then we can use the below approach

1. scp to the remote server

scp stands for secure copy is a shell command to copy files onto a remote computer and works based on ssh protocol

scp sourcefile user@ip:location

2. ssh-copyid

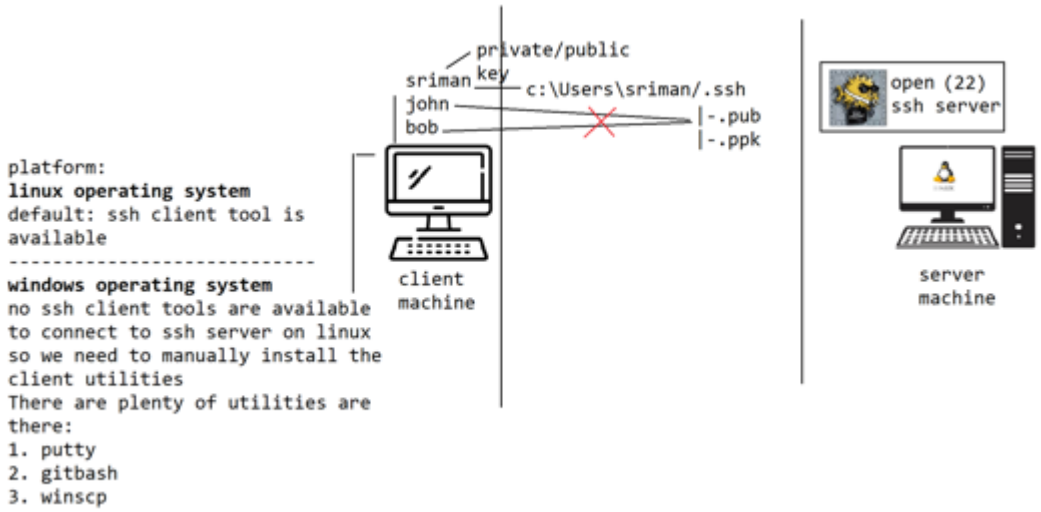
ssh-keygen

.pub and .pem =ssh required .pem as an input

puttygen

.pub/.ppk = putty requires, .ppk as input

==========



There are 2 tools available at client-side for ssh connection to the server.
#Server
seed the public key to the remote machine user onbehalf of whom we want to SSH
For eg.. if it is joe user then copy the public key into joe user authorized_keys
which means:- /home/joe/.ssh/authorized_keys

- ✓ **How to copy the public key onto the remote user?**

There are 3 ways are there
1. manually copy/paste
---
we need to have access to the remote machine over ssh protocol

2. scp the key to the remote machine
scp stands for secure copy which is used for copying the files onto the remote computer over ssh protocol
syntax:-
scp sourcefile user@ip:/path (location where you want to copy the file)
scp ~/.ssh/id_rsa.pub joe@192.168.1.113:/home/joe/.ssh/

you are on host machine:
scp user@ip:/path file
vm local
now the file will be copied to /home/joe/.ssh/ directory with the above command. now place (add) it into authorized_keys

~/.ssh:$
cat id_rsa.pub >> authorized_keys

3. ssh-copy-id
directly copies the key on to the remote user authorized_keys file

ssh-copy-id -i keyfile user@ip
for eg..
ssh-copy-id -i id_rsa.pub joe@192.168.1.113

The remote server always require public key only under authorized keys file of the remoteuser.

Client-Side
Windows platform:
----
putty: software to ssh onto the remote server
gitbash/powershell: ships bash ssh-client utility (similar to linux)

Mac/Linux platform:
----
bash ssh-client utility

To ssh onto the remote machine based on passwordlessAuthentication we need to generate keys. there are 2 technics are there in generating keys
1. ssh-keygen = linux bash utility
2. puttygen = windows tool

#1. ssh-keygen
It is an linux bash utility that most of the people uses for generating the keypair. the ssh-keygen generates a .pub/.pem file
.pub = public key which we can seed to the remote user
openssh private key file without extension

The pem file should be used for ssh into the remote server while using ssh-client utility
ssh -i ~/.ssh/id_rsa joe@192.168.1.113

#2. puttygen
it is an windows utility that is used for generating rsa and other algorithm specific public/private keys
it generates 2 keys
.ppk = private key
.pub = public key

so the above 2 keys cannot be used to ssh onto the remote server using bash ssh-client tool
because ssh-client requires .pem file only

we need putty only. putty tool takes .ppk and exchanges the key-based authentication and grants access to remote server

aws:
.pem = privacy enhanced mail
In cloud environments the cloud providers provides .pem file to us
we need ssh-client (bash) utility to connect to aws compute instance (remote server)

But if we want to connect to remote machine using putty (on windows) then we need .ppk/.pub, we can generate these 2 from .pem file using puttygen


```
client->server
scp abc.txt joe@192.168.10.12:/home/joe/

server-client
scp joe@192.168.10.12:/home/joe/xyz.txt /home/sriman/
```

shellscripting

Shellscript for static web application hosting

DEPLOY_SITE.SH

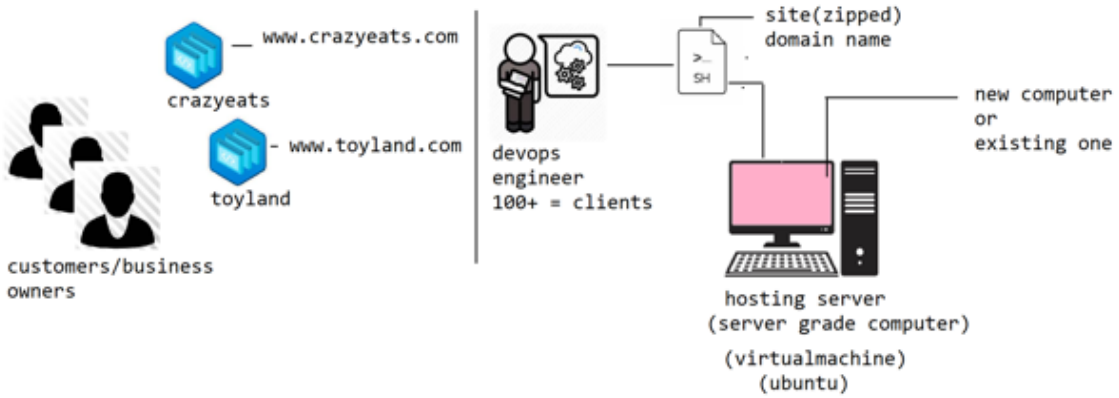
```
BEGIN
        read SITE_DIR
        read SITE_ZIP_LOCATION
        read DOMAIN_NM

        IF apache2 installed == FALSE
        THEN
                install it
        ELSE
                check apache2 status
                IF apache2 == not running
                THEN
                        give error message, saying start apache2 and run the script again
                        exit 1
                FI
        FI
        IF SITE_DIR exists in /var/www
        THEN
                rename with SITE_DIR_DATE
        FI
        copy the new site directory into /var/www
        IF SITE_CONFIG is existing == FALSE
        THEN
                create an site config file with domain entry
```

```
                enable site
                reload apache2 server
        END
        IF DOMAINNAME ENTRY NOT FOUND IN HOSTS_FILE
        THEN
                add an entry for the domain name
        FI
END


copy dir to /var/www
install apache2
create siteconfig
enable site, reload
add the domainname in host file
ufw enabled, the allow 80 port
```

deploysite.sh

```
----------
#!/bin/bash

function checkAndInstallApache2() {
        local APACHE2_INSTALL_STATUS=$(dpkg -s apache2 | grep "Status:" | awk '{print $4}')
        if [ $APACHE2_INSTALL_STATUS == " install ok installed"]
        then
                local APACHE2_RUNNING_STATUS=$(sudo systemctl status apache2 | grep "Active:" |
cut -d ":" -f2| awk '{print $1}')
                if [ $APACH2_RUNNING_STATUS != "active" ]
                then
                        echo "error: apache2 is down, start it and rerun the script for deployment"
                        exit 200
                fi
        else
                sudo apt update -y
                sudo apt install -y apache2
                local APACHE2_INSTALLED_STATUS = $?
                if [ $APACHE2_INSTALLED_STATUS == 1 ]
                then
                        echo "error: error while installing the apache2 server, please check the logs to
resolve"
                        exit 300
                fi
        fi

        return 0
}

#Main Program
```

```
NARGS=$#
if [ $NARGS -ne 3 ]
then
        echo "error: sitedirectory, zip location and domainname are required"
        exit 100
fi
SITE_DIR=$1
ZIP_LOCATION=$2
DOMAIN_NM=$3

checkAndInstallApache2
```



hostsite.sh

```
----------
#!/bin/bash

function checkAndInstallApache2Server() {
        dpkg -s apache2
        local APACHE2_SERVER_FOUND = $?
        if [ APACHE2_SERVER_FOUND -ne 0 ]
        then
                sudo apt update -y
                sudo apt install -y apache2
                local APACHE2_INSTALLED_STATUS=$?
                if [ $APACHE2_INSTALLED_STATUS -ne 0 ]
                then
                        return 200
                fi
        else
                local APACHE2_RUNNING_STATUS = $(sudo systemctl status apache2 | grep "Active" |
awk '{print $2}')
                if [ $APACHE2_RUNNING_STATUS == "inactive" ]
                then
                        return 300
                fi
```

```
	fi
	return 0
}

function setupSiteDirectory() {
	if [ ! -e $ZIP_LOC ]
	then
		return 400
	fi

	if [ -d /var/www/$SITE_DIR ]
	then
		var today=`date +%m%d%y%H%S`
		sudo mv /var/www/$SITE_DIR /var/www/$SITE_DIR_'$today'
	fi

	sudo cp $ZIP_LOC /var/www
	sudo tar -xzvf /var/www/$ZIP_LOC
	sudo chmod -R 755 /var/www/$SITE_DIR

	return 0
}

function checkAndConfigureApache2Site() {
	local var APACHE2_SITE_CONFIG_STATUS=0
	if [ ! -e /etc/apache2/sites-available/$SITE_DIR.conf ]
	then
		sudo cp apache2site.conf.tmpl /etc/apache2/sites-available/$SITE_DIR.conf
		sudo sed -i 's/:SERVER_NAME/$DOMAIN_NM/g' /etc/apache2/sites-
available/$SITE_DIR.conf
		sudo sed -i 's/:DOCUMENT_ROOT/$SITE_DIR/g' /etc/apache2/sites-
available/$SITE_DIR.conf
		sudo a2ensite $SITE_DIR
		sudo systemctl reload apache2
		local APACHE2_SITE_CONFIG_STATUS=$?
		return $APACHE2_RELOAD_STATUS
	fi
	return 0
}

function checkAndAddDNSEntry() {
	sudo cp /etc/hosts /etc/hosts.bak #good practise
	sudo grep "$DOMAIN_NM" /etc/hosts
	local DNS_ENTRY_FOUND=$?
	if [ $DNS_ENTRY_FOUND -ne 0 ]
	then
		sudo chmod 755 /etc/hosts
		sudo echo -e "\n127.0.0.1   $DOMAIN_NM" >> /etc/hosts
```

```
	fi
}

function checkUfw() {
	local UFW_STATUS=$(sudo ufw status | grep "Status:" | awk '{print $2}')
	if [ $UFW_STATUS == "active" ]
	then
		sudo ufw verbose | grep "80/tcp"
		local RULE_STATUS=$?
		if [ $RULE_STATUS -ne 0 ]
		then
			sudo ufw allow 80
		else
			local HTTP_STATUS=$(sudo ufw verbose | grep "80/tcp" | awk '{print $2}')
			if [ $HTTP_STATUS == "deny" ]
			then
				return 500
			fi
		fi
	fi
	return 0
}

#main block

NARGS=$#
if [ $NARGS -ne 3 ]
then
	echo "error: sitedirectory, zipLocation and domainname are required!"
	exit 100
fi
SITE_DIR=$1
ZIP_LOC=$2
DOMAIN_NM=$3

checkAndInstallApache2Server

var APACHE2_SERVER_STATUS=$?
if [ $APACHE2_SERVER_STATUS -eq 200 ]
then
	echo "error: failure during the installation of apache2 server, please check the logs"
	exit 200
elif [ $APACHE2_SERVER_STATUS -eq 300 ]
then
	echo "error: apache2 server is down, start and re-lauch the script to host the application"
	exit 300
fi
```

```
setupSiteDirectory
var SITE_DIRECTORY_STATUS=$?

if [ $SITE_DIRECTORY_STATUS -eq 400 ]
then
        echo "error: zip location is not found"
        exit 400
fi

checkAndConfigureApache2Site
var SITE_CONFIG_STATUS=$?
if [ $SITE_CONFIG_STATUS -ne 0 ]
then
        echo "error: while configuring apache2 site, please check the logs"
        exit $SITE_CONFIG_STATUS
fi
checkAndAddDNSEntry
checkUfw


hostsite crazyeats ~/public_html/crazyeats.tar.gz www.crazyeats.com
      $1            $2              $3

apache2site.conf.tmpl
<VirtualHost *:80>
        ServerName :SERVER_NAME:
        DocumentRoot /var/www/:DOCUMENT_ROOT:
</VirtualHost>

cp site.conf.tmp /etc/apache2/sites-available/crazyeats.conf
sed -i 's/:SERVER_NAME:/$DOMAIN_NM/g' /etc/apache2/sites-available/crazyeats.conf
sed -i 's/:DOCUMENT_ROOT:/$SITE_DIR/g' /etc/apache2/sites-available/crazyeats.conf
```

crazyeats.conf

```
-------------------
<VirtualHost *:80>
        ServerName www.crazyeats.com
        DocumentRoot /var/www/crazyeats
</VirtualHost>
```

toyland.conf

```
-----------------
<VirtualHost *:80>
        ServerName www.toyland.com
        DocumentRoot /var/www/toyland
</VirtualHost>
```