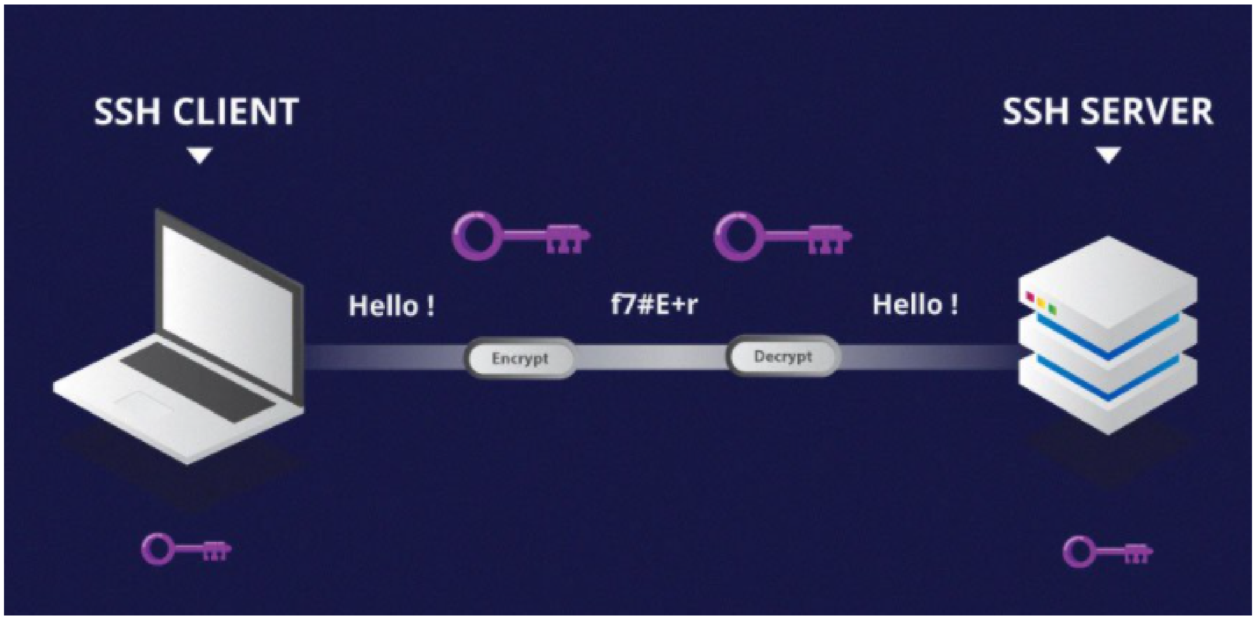
Using ssh for secure and passwordless authentication & Shell Variables

What is SSH?

SSH, or Secure Shell, is a remote administration protocol that allows users to control and modify their remote servers over the Internet. The service was created as a secure replacement for the unencrypted Telnet and uses cryptographic techniques to ensure that all communication to and from the remote server happens in an encrypted manner. It provides a mechanism for authenticating a remote user, transferring inputs from the client to the host, and relaying the output back to the client.



SSH Syntax

ssh username@ip/domainname

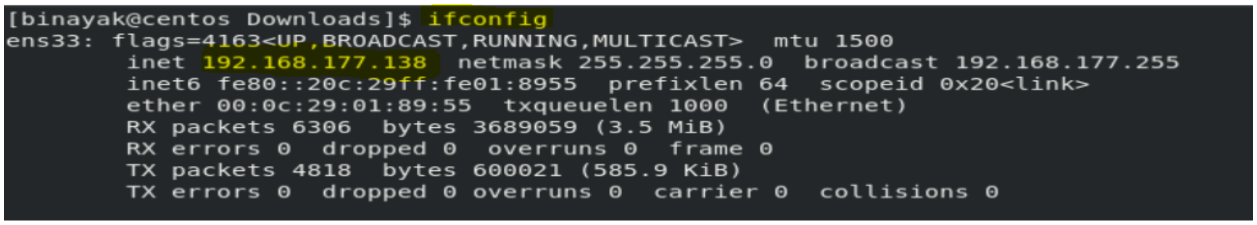
Example:

ssh binayak@192.168.177.138

The SSH key command instructs your system that you want to open an encrypted Secure Shell Connection. {user} represents the account you want to access. For example, you may want to access the root user, which is basically synonymous for system administrators with complete rights to modify anything on the system. {host} refers to the computer you want to access. This can be an IP Address (e.g. 244.235.23.19) or a domain name.

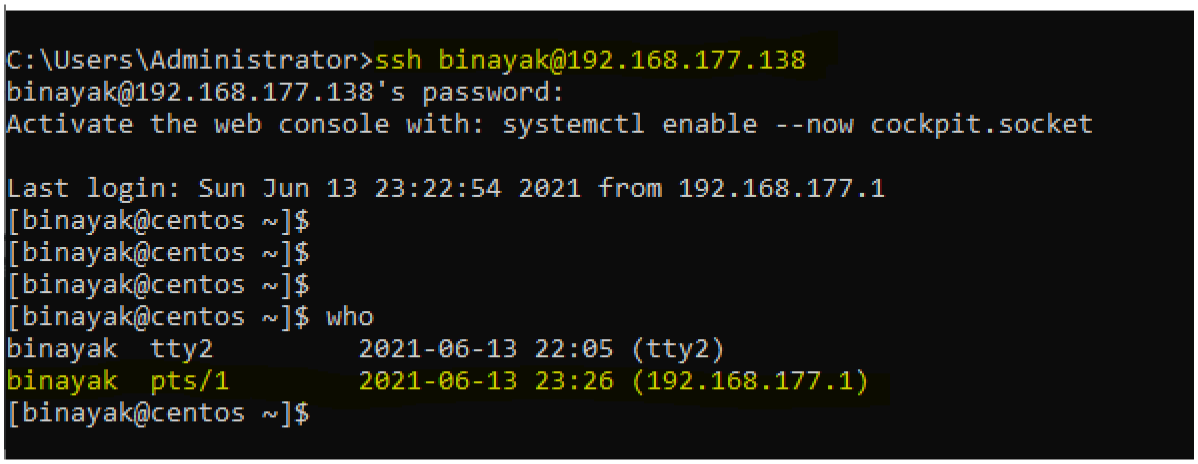
In order to take a ssh of your linux machine:

- Make sure your VM in on NAT/bridged connection
- Find ip of your VM using ifconfig command



```
[binayak@centos Downloads]$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.177.138  netmask 255.255.255.0  broadcast 192.168.177.255
        inet6 fe80::20c:29ff:fe01:8955  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:01:89:55  txqueuelen 1000  (Ethernet)
        RX packets 6306  bytes 3689059 (3.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4818  bytes 600021 (585.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- Go to your base machine
- Assuming your base machine is Microsoft Windows OS.
- Open command prompt.
- Enter ssh username@ip



```
C:\Users\Administrator>ssh binayak@192.168.177.138
binayak@192.168.177.138's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sun Jun 13 23:22:54 2021 from 192.168.177.1
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$ who
binayak  tty2         2021-06-13 22:05 (tty2)
binayak  pts/1        2021-06-13 23:26 (192.168.177.1)
[binayak@centos ~]$
```

What are SSH Keys?

SSH keys come in many sizes, but a popular choice is an RSA 2048-bit encryption, which is comparable to a 617 digit long password. On Windows systems, it is possible to generate your own SSH key pair by downloading and using an SSH client like PuTTY. On Mac and Linux systems, it is possible to generate an SSH key pair using a terminal window. Watch the video below to find out how to generate your own RSA key pair on Mac and Linux.

SSH keys always come in pairs, and every pair is made up of a private key and a public key. Who or what possesses these keys determines the type of SSH key pair. If the private key and the public key remain with the user, this set of SSH keys is referred to as user keys. If the private and public keys are on a remote system, then this key pair is referred to as host keys. Another type of SSH key is a session key. When a large amount of data is being transmitted, session keys are used to encrypt this information.

In a user key set, the private key remains on the system being used to access the remote system (i.e. the user's desktop or laptop) and is used to decrypt information that is exchanged in the SSH protocol. Private keys should never be shared with anyone and should be secured on a system – i.e. the system is secured (full disk encryption, MFA), in the user's possession, and the private key is secured via passphrase.

A public key is used to encrypt information, can be shared, and is used by the user and the remote server. On the server end, the public key is saved in a file that contains a list of authorized public keys. On the user's side, the public SSH key is stored in an SSH key management software or in a file on their computer.

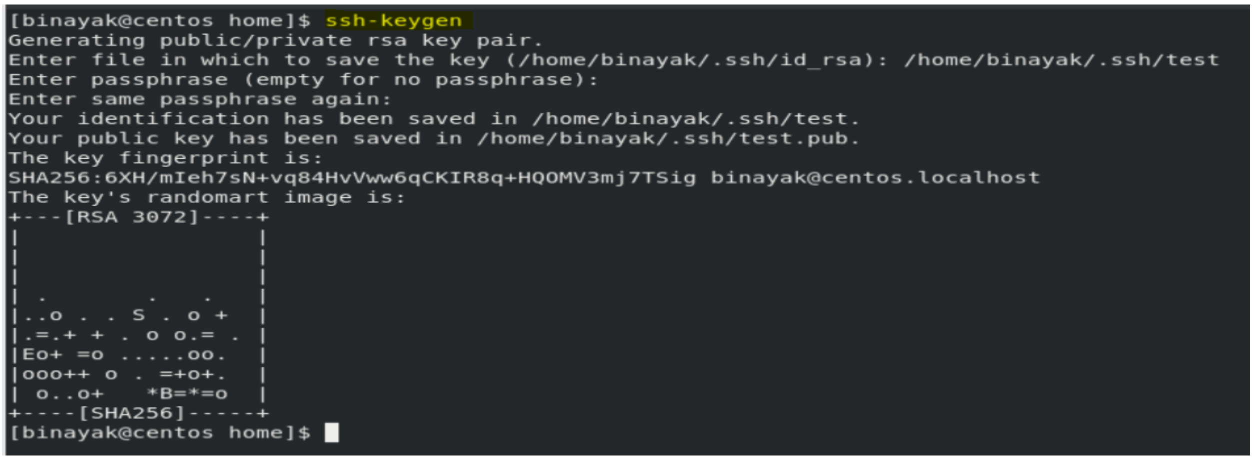
SSH keys provide a straightforward, secure method of logging into your server and are recommended for all users.

This demonstration requires two VM setup.

Make sure you have 2 linux VMs setup.

Step 1 — Creating the RSA Key Pair on VM1

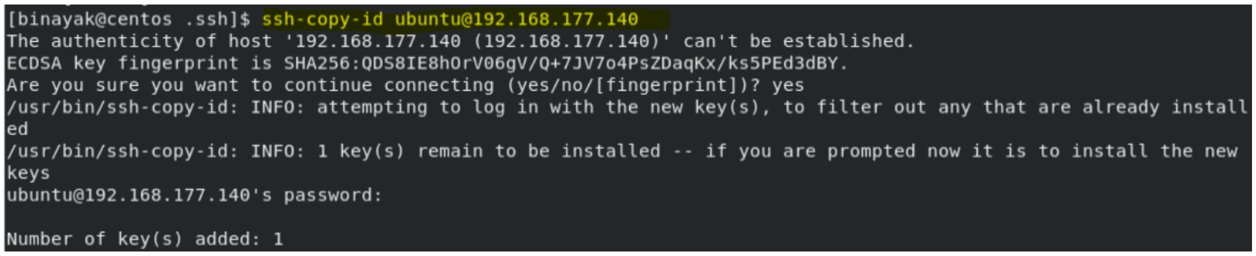
.



```
[binayak@centos home]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/binayak/.ssh/id_rsa): /home/binayak/.ssh/test
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/binayak/.ssh/test.
Your public key has been saved in /home/binayak/.ssh/test.pub.
The key fingerprint is:
SHA256:6XH/mIeh7sN+vq84HvVww6qCKIR8q+HQOMV3mj7TSig binayak@centos.localhost
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|                 |
| .       .   .   |
|..o . . S . o +  |
|.=.+ + . o o.= . |
|Eo+ =o .....oo.  |
|ooo++ o . =+o+.  |
| o..o+    *B=*=o |
+----[SHA256]-----+
[binayak@centos home]$ 
```

Step 2 — Copying the Public Key to Your VM2

The quickest way to copy your public key to the CentOS host is to use a utility called ssh-copy-id. This method is highly recommended if available. If you do not have ssh-copy-id available to you on your client machine, you may use one of the two alternate methods that follow (copying via password-based SSH, or manually copying the key).

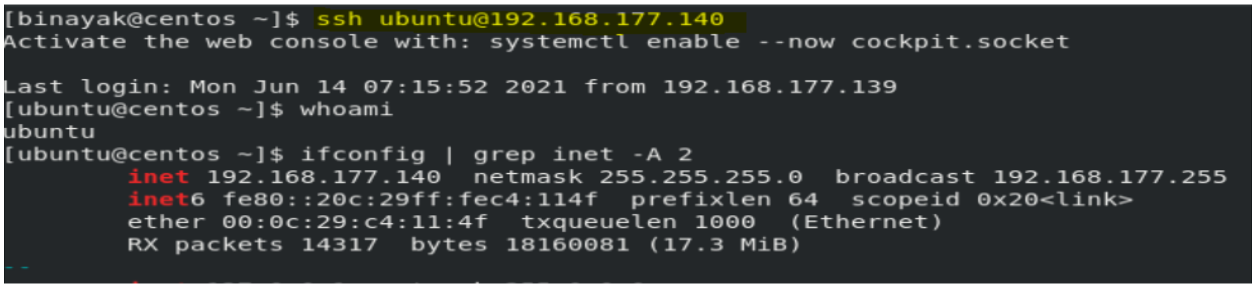


```
[binayak@centos .ssh]$ ssh-copy-id ubuntu@192.168.177.140
The authenticity of host '192.168.177.140 (192.168.177.140)' can't be established.
ECDSA key fingerprint is SHA256:QDS8IE8hOrV06gV/Q+7JV7o4PsZDaqKx/ks5PEd3dBY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already install
ed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new
keys
ubuntu@192.168.177.140's password:

Number of key(s) added: 1
```

Step 3 — Logging In to Your CentOS Server Using SSH Keys

Now just ssh into ubuntu user present on 192.168.177.140.



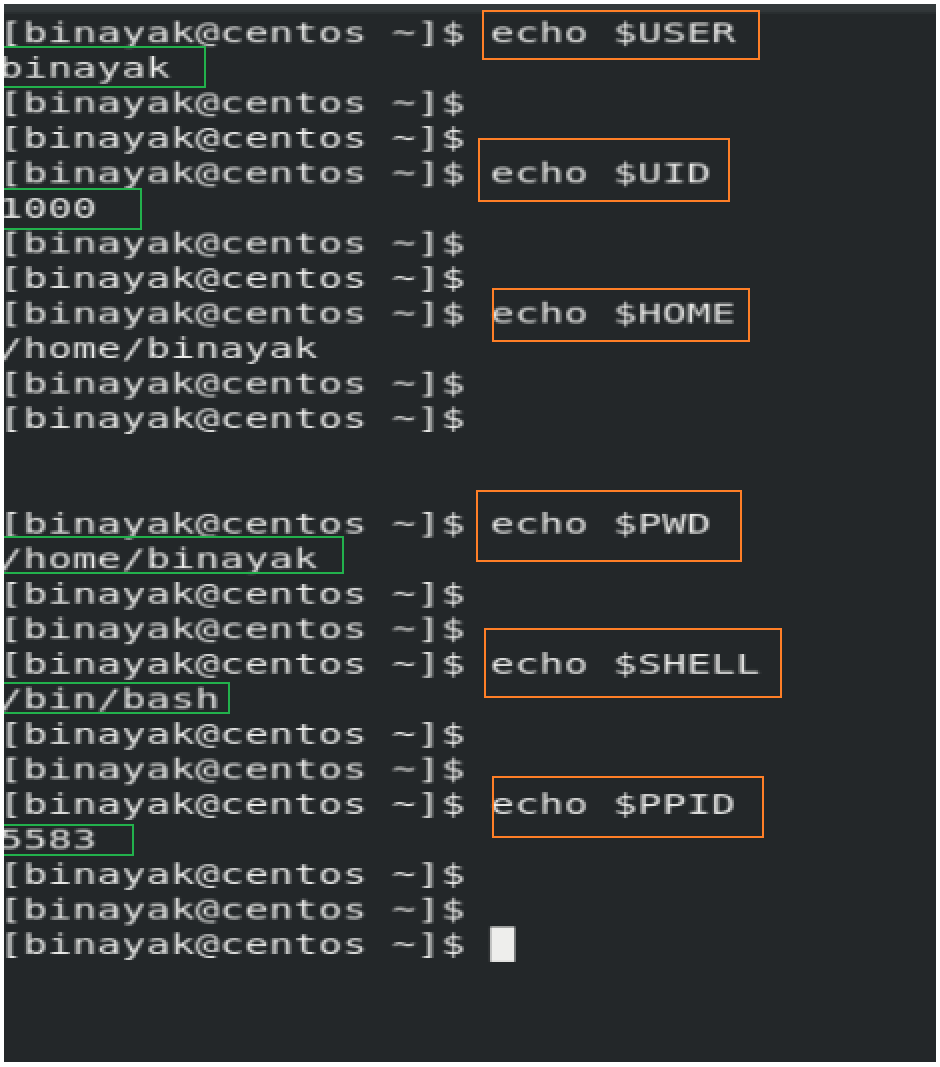
```
[binayak@centos ~]$ ssh ubuntu@192.168.177.140
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Jun 14 07:15:52 2021 from 192.168.177.139
[ubuntu@centos ~]$ whoami
ubuntu
[ubuntu@centos ~]$ ifconfig | grep inet -A 2
        inet 192.168.177.140  netmask 255.255.255.0  broadcast 192.168.177.255
        inet6 fe80::20c:29ff:fec4:114f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:c4:11:4f  txqueuelen 1000  (Ethernet)
        RX packets 14317  bytes 18160081 (17.3 MiB)
--
```

- **Shells and environments**

A shell provides a layer between you and the intricacies of an operating system. With Linux (and UNIX) shells, you can build complex operations by combining basic functions. Using programming constructs, you can then build functions for direct execution in the shell or save functions as shell scripts. A shell script is a sequence of commands that is stored in a file that the shell can run as needed.

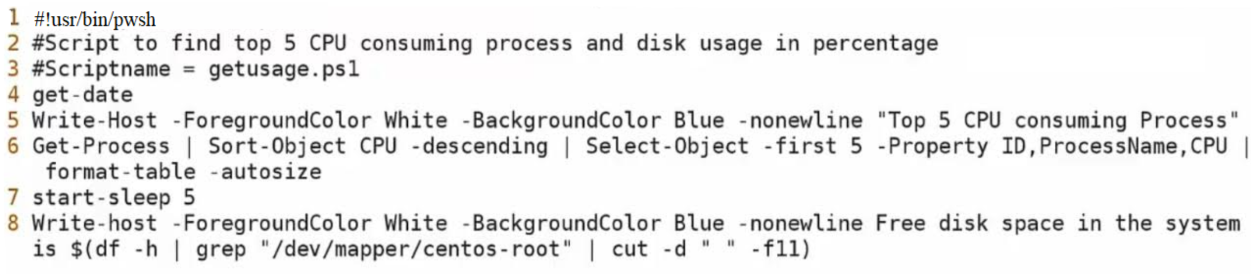
The environment is available to every shell process. When a shell starts, it assigns values from the environment to shell variables. You can use shells, including bash, to create and modify additional shell variables.



```
[binayak@centos ~]$ echo $USER
binayak
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$ echo $UID
1000
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$ echo $HOME
/home/binayak
[binayak@centos ~]$
[binayak@centos ~]$


[binayak@centos ~]$ echo $PWD
/home/binayak
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$ echo $SHELL
/bin/bash
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$ echo $PPID
5583
[binayak@centos ~]$
[binayak@centos ~]$
[binayak@centos ~]$
```

- **Basic shell scripting for system monitoring**



```
#!usr/bin/pwsh
#Script to find top 5 CPU consuming process and disk usage in percentage
#Scriptname = getusage.ps1
get-date
Write-Host -ForegroundColor White -BackgroundColor Blue -nonewline "Top 5 CPU consuming Process"
Get-Process | Sort-Object CPU -descending | Select-Object -first 5 -Property ID,ProcessName,CPU |
  format-table -autosize
start-sleep 5
Write-host -ForegroundColor White -BackgroundColor Blue -nonewline Free disk space in the system
is $(df -h | grep "/dev/mapper/centos-root" | cut -d " " -f11)
```