Name: Revilla, Aldwin Joseph B.	**Date Performed:** 09/05/2023
Course/Section: CPE31S5	**Date Submitted:** 09/06/2023
Instructor: Engr. Roman Richard	**Semester and SY:** 2023-2024

Activity 3: Install SSH server on CentOS or RHEL 8

1. **Objectives:**

1.1 Install Community Enterprise OS or Red Hat Linux OS

1.2 Configure remote SSH connection from remote computer to CentOS/RHEL-8

2. **Discussion:**

CentOS vs. Debian: Overview

CentOS and Debian are Linux distributions that spawn from opposite ends of the candle.

CentOS is a free downstream rebuild of the commercial Red Hat Enterprise Linux distribution where, in contrast, Debian is the free upstream distribution that is the base for other distributions, including the Ubuntu Linux distribution.

As with many Linux distributions, CentOS and Debian are generally more alike than different; it isn't until we dig a little deeper that we find where they branch.

CentOS vs. Debian: Architecture

The available supported architectures can be the determining factor as to whether a distro is a viable option or not. Debian and CentOS are both very popular for x86_64/AMD64, but what other archs are supported by each?

Both Debian and CentOS support AArch64/ARM64, armhf/armhfp , i386 , ppc64el/ppc64le. (Note: armhf/armhfp and i386 are supported in CentOS 7 only.)

CentOS 7 additionally supports POWER9 while Debian and CentOS 8 do not. CentOS 7 focuses on the x86_64/AMD64 architecture with the other archs released through the AltArch SIG (Alternate Architecture Special Interest Group) with CentOS 8 supporting x86_64/AMD64, AArch64 and ppc64le equally.

Debian supports MIPSel, MIPS64el and s390x while CentOS does not. Much like CentOS 8, Debian does not favor one arch over another —all supported architectures are supported equally.

CentOS vs. Debian: Package Management

Most Linux distributions have some form of package manager nowadays, with some more complex and feature-rich than others.

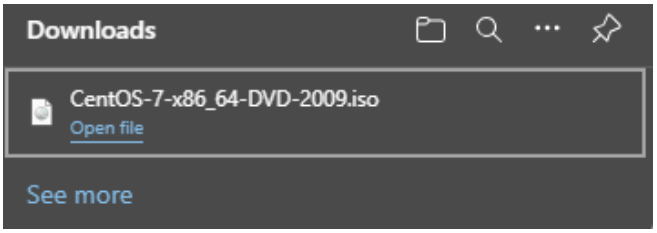
CentOS uses the RPM package format and YUM/DNF as the package manager.

Debian uses the DEB package format and dpkg/APT as the package manager.

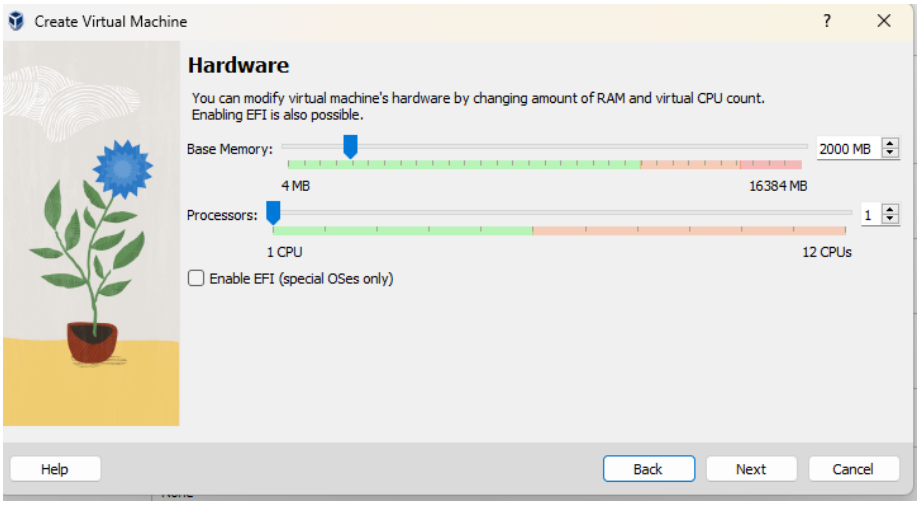
Both offer full-feature package management with network-based repository support, dependency checking and resolution, etc.. If you're familiar with one but not the other, you may have a little trouble switching over, but they're not overwhelmingly different. They both have similar features, just available through a different interface.

Task 1: Download the CentOS or RHEL-8 image (Create screenshots of the following)

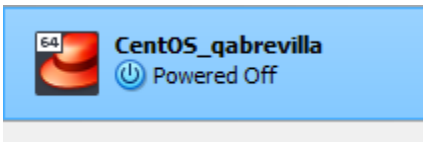
1. Download the image of the CentOS here: http://mirror.rise.ph/centos/7.9.2009/isos/x86_64/



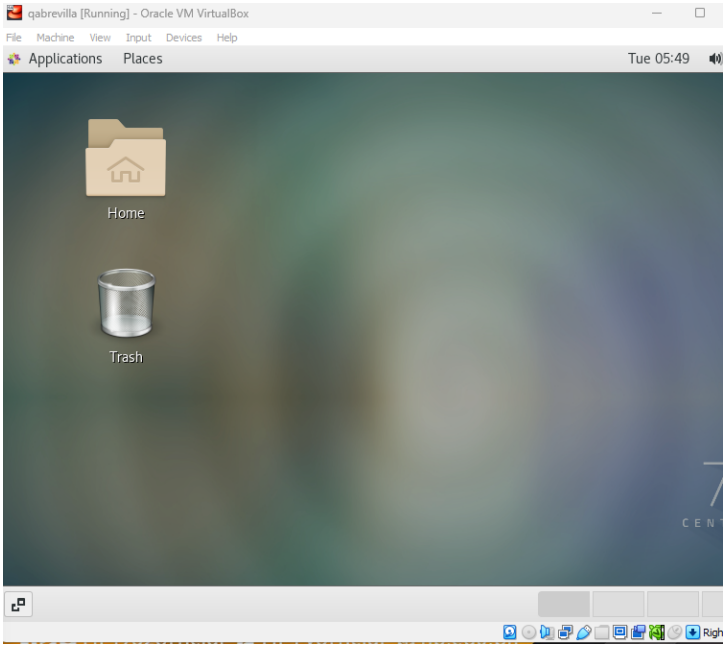
2. Create a VM machine with 2 Gb RAM and 20 Gb HD.



3. Install the downloaded image.



4. Show evidence that the OS was installed already.



Task 2: Install the SSH server package *openssh*

1. Install the ssh server package *openssh* by using the *dnf* command:
   $ dnf install openssh-server

```
[qabrevilla@localhost ~]$ sudo yum install openssh-server

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for qabrevilla:
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.rise.ph
 * extras: mirror.rise.ph
 * updates: mirror.rise.ph
Resolving Dependencies
--> Running transaction check
---> Package openssh-server.x86_64 0:7.4p1-21.el7 will be updated
---> Package openssh-server.x86_64 0:7.4p1-23.el7_9 will be an update
--> Processing Dependency: openssh = 7.4p1-23.el7_9 for package: openssh-server-7.4p1-2
3.el7_9.x86_64
--> Running transaction check
---> Package openssh.x86_64 0:7.4p1-21.el7 will be updated
--> Processing Dependency: openssh = 7.4p1-21.el7 for package: openssh-clients-7.4p1-21
.el7.x86_64
---> Package openssh.x86_64 0:7.4p1-23.el7_9 will be an update
```

```
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : openssh-7.4p1-23.el7_9.x86_64                                1/6
  Updating   : openssh-clients-7.4p1-23.el7_9.x86_64                        2/6
  Updating   : openssh-server-7.4p1-23.el7_9.x86_64                         3/6
  Cleanup    : openssh-server-7.4p1-21.el7.x86_64                           4/6
  Cleanup    : openssh-clients-7.4p1-21.el7.x86_64                          5/6
  Cleanup    : openssh-7.4p1-21.el7.x86_64                                  6/6
  Verifying  : openssh-7.4p1-23.el7_9.x86_64                                1/6
  Verifying  : openssh-clients-7.4p1-23.el7_9.x86_64                        2/6
  Verifying  : openssh-server-7.4p1-23.el7_9.x86_64                         3/6
  Verifying  : openssh-clients-7.4p1-21.el7.x86_64                          4/6
  Verifying  : openssh-7.4p1-21.el7.x86_64                                  5/6
  Verifying  : openssh-server-7.4p1-21.el7.x86_64                           6/6

Updated:
  openssh-server.x86_64 0:7.4p1-23.el7_9

Dependency Updated:
  openssh.x86_64 0:7.4p1-23.el7_9        openssh-clients.x86_64 0:7.4p1-23.el7_9

Complete!
[qabrevilla@localhost ~]$
```

2. Start the *sshd* daemon and set to start after reboot:
   $ systemctl start sshd

```
[qabrevilla@localhost ~]$ systemctl start sshd
[qabrevilla@localhost ~]$
```

   $ systemctl enable sshd

```
Complete!
[qabrevilla@localhost ~]$ systemctl start sshd
[qabrevilla@localhost ~]$ systemctl enable sshd
[qabrevilla@localhost ~]$
```

3. Confirm that the sshd daemon is up and running:
   $ systemctl status sshd

```
qabrevilla@localhost:~
File  Edit  View  Search  Terminal  Help

Dependency Updated:
  openssh.x86_64 0:7.4p1-23.el7_9        openssh-clients.x86_64 0:7.4p1-23.el7_9

Complete!
qabrevilla@localhost ~]$ systemctl start sshd
qabrevilla@localhost ~]$ systemctl enable sshd
qabrevilla@localhost ~]$ systemctl status sshd
sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enable
d)
   Active: active (running) since Tue 2023-09-05 21:48:59 PST; 3min 38s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 3082 (sshd)
   CGroup: /system.slice/sshd.service
           └─3082 /usr/sbin/sshd -D

Sep 05 21:48:59 localhost.localdomain systemd[1]: Stopped OpenSSH server daemon.
Sep 05 21:48:59 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
Sep 05 21:48:59 localhost.localdomain sshd[3082]: Server listening on 0.0.0.0 port 22.
Sep 05 21:48:59 localhost.localdomain sshd[3082]: Server listening on :: port 22.
Sep 05 21:48:59 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
Hint: Some lines were ellipsized, use -l to show in full.
qabrevilla@localhost ~]$
```

4. Open the SSH port 22 to allow incoming traffic:
   $ firewall-cmd --zone=public --permanent --add-service=ssh

```
[qabrevilla@localhost ~]$ firewall-cmd --zone=public --permanent --add-service=
Warning: ALREADY_ENABLED: ssh
success
[qabrevilla@localhost ~]$
```

$ firewall-cmd --reload

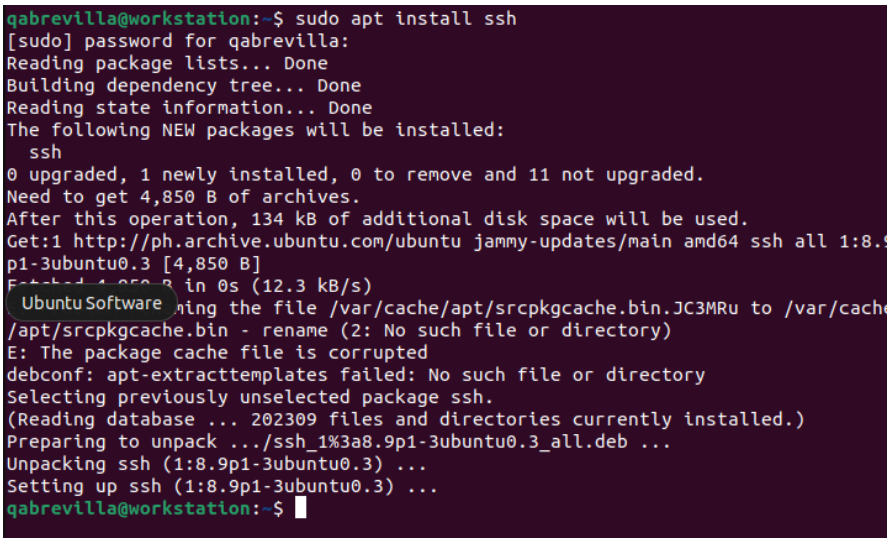
```
[qabrevilla@localhost ~]$ firewall-cmd --reload
success
```

5. Locate the ssh server man config file /*etc/ssh/sshd_config* and perform custom configuration. Every time you make any change to the /*etc/ssh/sshd-config* configuration file reload the *sshd* service to apply changes:
   $ systemctl reload sshd

```
[root@localhost ~]# systemctl reload sshd
[root@localhost ~]#
```

Task 3: Copy the Public Key to CentOS

1. Make sure that *ssh* is installed on the local machine.



2. Using the command *ssh-copy-id*, connect your local machine to CentOS.

```
qabrevilla@workstation:~$ ssh-copy-id qabrevilla@192.168.56.106
The authenticity of host '192.168.56.106 (192.168.56.106)' can't be established.
ED25519 key fingerprint is SHA256:pGKv/+s/xpdgKS3lIsV7fMBz6IfZe3qn73zFwfdjFlc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
qabrevilla@192.168.56.106's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'qabrevilla@192.168.56.106'"
and check to make sure that only the key(s) you wanted were added.
```

3. On CentOS, verify that you have the *authorized_keys*.

```
[qabrevilla@localhost ~]$ ls -la .ssh
total 8
drwx------.  2 qabrevilla qabrevilla   29 Sep  5 22:17 .
drwx------. 16 qabrevilla qabrevilla 4096 Sep  5 22:17 ..
-rw-------.  1 qabrevilla qabrevilla  748 Sep  5 22:17 authorized_keys
[qabrevilla@localhost ~]$ S
```

Task 4: Verify ssh remote connection

1. Using your local machine, connect to CentOS using ssh.

```
qabrevilla@workstation:~$ ssh qabrevilla@192.168.56.106
Last login: Tue Sep  5 21:47:53 2023
[qabrevilla@localhost ~]$
```

2. Show evidence that you are connected.

```
[qabrevilla@localhost ~]$ ping 192.168.56.106
PING 192.168.56.106 (192.168.56.106) 56(84) bytes of data.
64 bytes from 192.168.56.106: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from 192.168.56.106: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 192.168.56.106: icmp_seq=3 ttl=64 time=0.086 ms
64 bytes from 192.168.56.106: icmp_seq=4 ttl=64 time=0.042 ms
64 bytes from 192.168.56.106: icmp_seq=5 ttl=64 time=0.034 ms
64 bytes from 192.168.56.106: icmp_seq=6 ttl=64 time=0.038 ms
^Z
[1]+  Stopped                 ping 192.168.56.106
[qabrevilla@localhost ~]$
```

```
[qabrevilla@localhost ~]$ ping 192.168.56.106
PING 192.168.56.106 (192.168.56.106) 56(84) bytes of data.
64 bytes from 192.168.56.106: icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from 192.168.56.106: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 192.168.56.106: icmp_seq=3 ttl=64 time=0.037 ms
^Z
[1]+  Stopped                 ping 192.168.56.106
[qabrevilla@localhost ~]$
```

Reflections:

Answer the following:

1. What do you think we should look for in choosing the best distribution between Debian and Red Hat Linux distributions?

   Both Debian and Red Hat Linux are well known Linux distributions. Choosing, the first thing we should consider is the usage of the distribution considering if it is for business or project. Red Hat is more for business and organizations that require a high level of security. While Debian is for community-driven projects that are more user friendly and designed for desktop users.

2. What are the main diffence between Debian and Red Hat Linux distributions?

   Red Hat is known for its performance in enterprise grade applications and high grade security and support features. If choosing a more user friendly distribution for managing documents, developing software, administering systems, networking etc, Debian is a better option.

Conclusion

In today's activity, We focused more on configuring ssh on CentOS distribution. We install the ssh server by typing “sudo yum install openssh-server” that enables the ssh package on CentOS. Also after enabling the ssh, we open the ssh port 22 to allow the incoming traffic to allow administration access. Lastly, we configure the CentOS and local machine for setting up the public key on CentOS machines. I also learn the differences of Debian, CentOS, and Red Hat Linux distributions. As a system administrator we should know the differences of the distribution so that we can use and recommend the right distribution for specific usage.