Name: Gabiano, Chris Leonard A.	Date Performed: Sept 13, 2023
Course/Section: CPE31S6	Date Submitted: Sept 13, 2023
Instructor: Dr. Taylar	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.

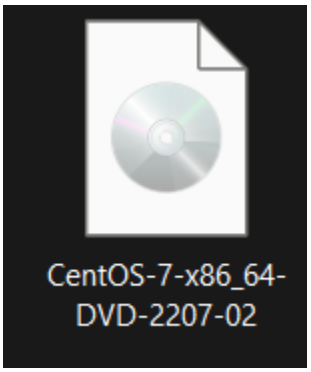
CentOS uses the RPM package format and YUM/DNF as the package manager.

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)

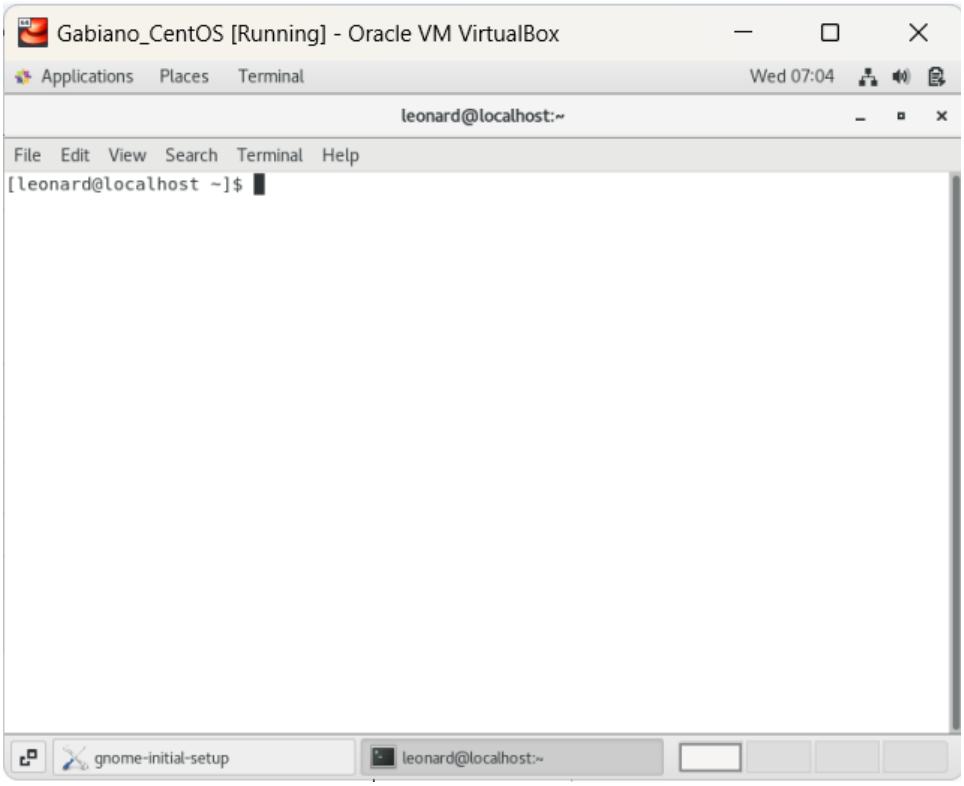
1. Download the image of the CentOS here:



2. http://mirror.rise.ph/centos/7.9.2009/isos/x86_64/
3. Create a VM machine with 2 Gb RAM and 20 Gb HD.

4. Install the downloaded image.

5. 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

```
[leonard@localhost ~]$ sudo yum -y install openssh-server
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.gbnetwork.com
 * extras: mirrors.gbnetwork.com
 * updates: mirrors.gbnetwork.com
Resolving Dependencies
--> Running transaction check
---> Package openssh-server.x86_64 0:7.4p1-22.el7_9 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 0:7.4p1-22.el7_9 will be updated
--> Processing Dependency: openssh = 7.4p1-22.el7_9 for package: openssh-clients-7.4p1-
22.el7_9.x86_64
---> Package openssh.x86_64 0:7.4p1-23.el7_9 will be an update
--> Running transaction check
---> Package openssh-clients.x86_64 0:7.4p1-22.el7_9 will be updated
---> Package openssh-clients.x86_64 0:7.4p1-23.el7_9 will be an update
--> Finished Dependency Resolution
```

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

$ systemctl enable sshd

```
[leonard@localhost ~]$ systemctl start sshd
[leonard@localhost ~]$ systemctl status enable sshd
Unit enable.service could not be found.
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enable
d)
   Active: active (running) since Wed 2023-09-13 07:14:18 EDT; 4min 49s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 10733 (sshd)
    Tasks: 1
   CGroup: /system.slice/sshd.service
           └─10733 /usr/sbin/sshd -D
```

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

```
[leonard@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 Wed 2023-09-13 07:14:18 EDT; 5min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 10733 (sshd)
    Tasks: 1
   CGroup: /system.slice/sshd.service
           └─10733 /usr/sbin/sshd -D

Sep 13 07:14:18 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
Sep 13 07:14:18 localhost.localdomain sshd[10733]: Server listening on 0.0.0.0 port 22.
Sep 13 07:14:18 localhost.localdomain sshd[10733]: Server listening on :: port 22.
Sep 13 07:14:18 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
Hint: Some lines were ellipsized, use -l to show in full.
[leonard@localhost ~]$
```

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

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

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

```
[leonard@localhost ~]$ systemctl reload sshd
[leonard@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.

```
leonard@WORKSTATION:~$ ssh-copy-id leonard@192.168.53.191
The authenticity of host '192.168.53.191 (192.168.53.191)' can't be established
.
ED25519 key fingerprint is SHA256:XycS+EggsiLTQ1dYnwd8BjLYTSWpDnIXOFUMjWuzL1c.
This key is not known by any other names
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 installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
leonard@192.168.53.191's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'leonard@192.168.53.191'"
and check to make sure that only the key(s) you wanted were added.
```

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

```
[leonard@localhost .ssh]$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCOL/DocKi6XefglizMNN6n3p1EG2xme00jk7Hs0d0ndpE1bgz
MAVGh/4W/caFYaM1loaGrM5Sv3fIGCpZgPCNoWPp1C6L9RK1JD6l+p4fWPgsno2en7aaHfLbHSzpnsZiYwHNP9g
VjO+tUUk242DRnZwGz//veBc4EZH4AbWUoeFmpKg2B1tTkLQJ7hwEuqiZIFkxQ2YR9caaXNlvnGsboHTELIQqeA
L5NyjbGolWcyULhbdcJpSfheNuvqi9+MDdf9kfmQFDmdpDYXZ/1ulB8HXCHqSmR2JSN+T5zl93SosQZswJTsYrm
ocxgeiQXC8r9pmubv/E/yY6LUlmbiPkWhj7Gb1olIlu3wDYDcUUuZtLlhl0wI263k0Gh6cUG6LMXp7kyBtL/r3W
PAQLxP8m1mg0Sx1qZLOPsssLIJG+G+vUbWoF/MZsif/ReAa21G9gP3kClDx60vInjB0I3dpQxacPuGAYeJ/YWyP
yqasGtckHqFwIEvImK90+ZWv5gQd88Jz+EsmIe6+rwovv6HLcJDQhpqQ4pGDRtoyOfT/tht357ng9tzepE3te9i
QyEUfHFw6zWaZwq+VAH03WIkY5pyJkhh5FutmHZe8iMnuOUQp3HNTq2IhE/386Peo6PWPS/liXhpfai9W04sUJL
af/fjH36ApyK1E+7vBT46IzWhw== leonard@WORKSTATION
[leonard@localhost .ssh]$
```

Task 4: Verify ssh remote connection

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

```
leonard@WORKSTATION:~$ ping 192.168.53.191
PING 192.168.53.191 (192.168.53.191) 56(84) bytes of data.
64 bytes from 192.168.53.191: icmp_seq=1 ttl=63 time=2.99 ms
64 bytes from 192.168.53.191: icmp_seq=2 ttl=63 time=2.09 ms
64 bytes from 192.168.53.191: icmp_seq=3 ttl=63 time=1.62 ms
64 bytes from 192.168.53.191: icmp_seq=4 ttl=63 time=2.53 ms

64 bytes from 192.168.53.191: icmp_seq=5 ttl=63 time=2.56 ms
^C
--- 192.168.53.191 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4012ms
rtt min/avg/max/mdev = 1.618/2.357/2.989/0.466 ms
```

2. Show evidence that you are connected.

```
leonard@WORKSTATION:~$ ping 192.168.53.191
PING 192.168.53.191 (192.168.53.191) 56(84) bytes of data.
64 bytes from 192.168.53.191: icmp_seq=1 ttl=63 time=2.99 ms
64 bytes from 192.168.53.191: icmp_seq=2 ttl=63 time=2.09 ms
64 bytes from 192.168.53.191: icmp_seq=3 ttl=63 time=1.62 ms
64 bytes from 192.168.53.191: icmp_seq=4 ttl=63 time=2.53 ms

64 bytes from 192.168.53.191: icmp_seq=5 ttl=63 time=2.56 ms
^C
--- 192.168.53.191 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4012ms
rtt min/avg/max/mdev = 1.618/2.357/2.989/0.466 ms
```

```
[leonard@localhost ~]$ ping 192.168.53.191
PING 192.168.53.191 (192.168.53.191) 56(84) bytes of data.
64 bytes from 192.168.53.191: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from 192.168.53.191: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 192.168.53.191: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 192.168.53.191: icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from 192.168.53.191: icmp_seq=5 ttl=64 time=0.052 ms
64 bytes from 192.168.53.191: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 192.168.53.191: icmp_seq=7 ttl=64 time=0.049 ms
64 bytes from 192.168.53.191: icmp_seq=8 ttl=64 time=0.050 ms
64 bytes from 192.168.53.191: icmp_seq=9 ttl=64 time=0.058 ms
64 bytes from 192.168.53.191: icmp_seq=10 ttl=64 time=0.070 ms

64 bytes from 192.168.53.191: icmp_seq=11 ttl=64 time=0.056 ms
64 bytes from 192.168.53.191: icmp_seq=12 ttl=64 time=0.087 ms
^C
--- 192.168.53.191 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11027ms
rtt min/avg/max/mdev = 0.043/0.057/0.087/0.011 ms
```

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?**

   When making a choice between Debian and Red Hat Linux distributions as an ICT student, it's akin to selecting tools for a specific task. Debian can be seen as a Swiss Army knife of sorts, offering a broad range of open-source software and community-driven support, making it suitable for exploration and personal projects. Red Hat, on the other hand, is like a specialized, professional-grade toolset, emphasizing stability, security, and enterprise-level support.

   If you're still learning the ropes or want to experiment with Linux, Debian's flexibility and community help may be more fitting. However, if your ICT studies focus on large-scale systems or corporate IT environments, Red Hat's robustness and official support can provide valuable exposure.

   In essence, Debian is the versatile all-rounder, while Red Hat is the reliable, heavy-duty choice. The decision should hinge on your learning objectives and

whether you prefer a diverse, open-source ecosystem or the dependability and support that a more enterprise-focused distribution can offer.

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

In the realm of Linux distributions, Debian and Red Hat stand apart with distinct characteristics. Debian is like an open-source playground, where a diverse community collaborates to provide a wide array of software options for enthusiasts and learners. On the other hand, Red Hat takes on a more corporate role, offering Red Hat Enterprise Linux (RHEL) as a dependable choice for businesses, complete with professional support, certifications, and management tools. As an ICT student, think of Debian as your creative sandbox for experimentation and learning, while Red Hat is the seasoned, reliable professional you turn to when stability and support are paramount.