Please complete all quizzes, guided exercises, and labs for each chapter. Insert YES for each completed quiz and your screenshots in appropriate locations for each guided exercise/lab

Guided Exercise: Access the Remote Command Line

In this exercise, you log in to a remote system as different users and execute commands.

Outcomes

- Log in to a remote system.
- Execute commands with the OpenSSH secure shell.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start ssh-access
```

Procedure 10.1. Instructions

1. From `workstation`, open an SSH session to the `servera` machine as the `student` user.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$
```

2. Open an SSH session to the `serverb` machine as the `student` user. Accept the host key. Use `student` as the password when prompted for the password of the `student` user on the `serverb` machine.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be
established.
ED25519 key fingerprint is
SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KWOTK80.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
Warning: Permanently added 'serverb' (ED25519) to the list of known
hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

The `ssh` command records the host key on the `/home/student/.ssh/known_hosts` file in the `servera` machine to identify the `serverb` machine. The `student` user initiated the

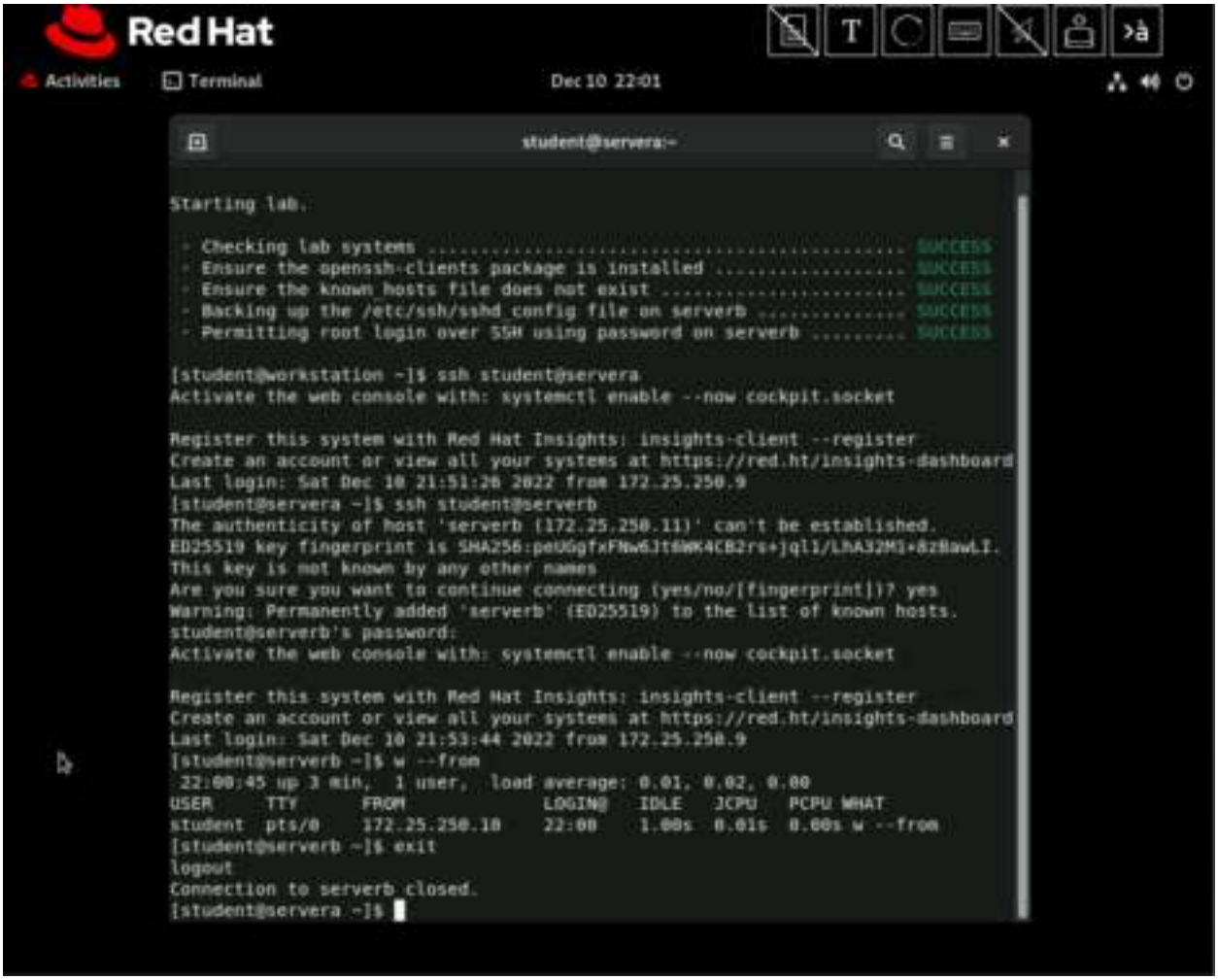
SSH connection from the `servera` machine. If the `/home/student/.ssh/known_hosts` file does not exist, then it is created along with the new entry in it. The `ssh` command fails to execute properly if the remote host appears to have a different key from the recorded key.

3. Display the users that are currently logged in to the `serverb` machine. The `student` user is logged in to the system from the host with an IP address of `172.25.250.10`, which is the `servera` machine in the classroom network.

```
[student@serverb ~]$ w --from
03:39:04 up 16 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
student  pts/0    172.25.250.10    20:40    1.00s  0.01s  0.00s w --
from
```

4. Exit the `student` user's shell on the `serverb` machine.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$
```



5. Open an SSH session to the `serverb` machine as the `root` user. Use `redhat` as the password of the `root` user. The command did not ask you to accept the host key, because it was found among the known hosts. If the identity of the `serverb` machine changes, then OpenSSH prompts you to challenge the new host key.

```
[student@servera ~]$ ssh root@serverb
root@serverb's password: redhat
...output omitted...
[root@serverb ~]#
```

6. Run the `w` command to display the users that are currently logged in to the `serverb` machine. The output indicates that the `root` user is logged in to the system from the host with an IP address of `172.25.250.10`, which is the `servera` machine in the classroom network.

```
[root@serverb ~]# w --from
03:46:05 up 23 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0    172.25.250.10    20:44    1.00s  0.02s  0.00s w --
from
```

7. Exit the `root` user's shell on the `serverb` machine.

```
[root@serverb ~]# exit
logout
Connection to serverb closed.
[student@servera ~]$
```

8. Remove the `/home/student/.ssh/known_hosts` file from the `servera` machine. This operation causes `ssh` to lose the recorded identities of the remote systems.

```
[student@servera ~]$ rm /home/student/.ssh/known_hosts
```

Host keys can change for legitimate reasons: perhaps the remote machine was replaced because of a hardware failure, or the remote machine was reinstalled. Usually, it is advisable to remove the key entry only for the particular host in the `known_hosts` file. Because this particular `known_hosts` file has only one entry, you can remove the entire file.

9. Open an SSH session to the `serverb` machine as the `student` user. If asked, accept the host key. Use `student` when prompted for the password of the `student` user on the `serverb` machine.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be
established.
ED25519 key fingerprint is
SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KWOTK80.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
Warning: Permanently added 'serverb' (ED25519) to the list of known
hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

The `ssh` command asked for your confirmation to accept or reject the host key because it could not find one for the remote host.

10. Exit the `student` user's shell on the `serverb` machine and confirm that a new instance of `known_hosts` exists on the `servera` machine.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$ ls -l /home/student/.ssh/known_hosts
-rw-------. 1 student student 819 Mar 24 03:47
/home/student/.ssh/known_hosts
```

11. Confirm that the new instance of the `known_hosts` file has the host key of the `serverb` machine. The following command output is an example; the actual output on your `workstation` might be different.

```
[student@servera ~]$ cat /home/student/.ssh/known_hosts
...output omitted...
serverb ecdsa-sha2-nistp256 AAAAB3NzaC1yc2EAAAADAQ...
...output omitted...
```

12. Run the `hostname` command remotely on the `serverb` machine without accessing the interactive shell.

```
[student@servera ~]$ ssh student@serverb hostname
student@serverb's password: student
serverb.lab.example.com
```

13. Return to the `workstation` system as the `student` user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ssh-access
```

This concludes the section.


Red Hat
Activities
Terminal
Dec 10 22:05
student@workstation:~
Last login: Sat Dec 10 22:00:26 2022 from 172.25.250.10
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$ ls -l /home/student/.ssh/known_hosts
-rw-------. 1 student student 819 Dec 10 22:03 /home/student/.ssh/known_hosts
[student@servera ~]$ cat /home/student/.ssh/known_hosts
serverb ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOmiLKMExRns51g70TxMsOmgHuUSGOBUxHh
uUGcv19uT
serverb ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCBWDOooY+rh6NPa9yhLsNQXBqcQknTL/W
5d3zPvHLLd7KaC4IiEUxnwbfLBitBtRcirbQFxO2OAm+6WJ2kpo34ZnlQYJz3AulZjuvat7x9OxzO2d
uSwSUbLtqteMtl3vS7RPUZlSnGQKuPrkcLrWYZDQveC11efyBhyskjJYPXu7Csd3iA2EmtcNa18AUoK
0f2XOC7R9wSb5D1md1lvaavQ107YFlzDHppRncWI51PwbyICaCVq7Kv4LdrcmQfsAyobPizWgp7u3Ui
/Narxk7ev50tmBmI3nxnC6Fiec60zqhbWrnFI062kV9p+YWHOExjLEY1WRR3GJ26dAyfief2dpbuza+
LHG88Nfo0j3m3PlxW+AvM9eeboJEu+h7KfqWlXr2zr5s7pdF8pM3f7+dEG9uaWaeznUc7N9XFXh+keV
xBJZK4PUy5u2Ytqv9Fm4WNvpnvXSmav9lfevtZFe2XFj9RcVqTrUFJ7Zz/87V2JCm5ILGNbzR9z9mhc
HRMtxc=
serverb ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAAB
BBC58A5ZVCNeibYgG2JlhzCLqJRVHdXmhNCBEGiQ+5ywTj/127A55guBZGLzyMp6K1J4wuTWHS4RfZz
vycmC8vto=
[student@servera ~]$ ssh student@serverb hostname
student@serverb's password:
serverb.lab.example.com
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ lab finish ssh-access
Finishing lab.
- Checking lab systems ..... SUCCESS
- Restoring original /etc/ssh/sshd_config on serverb ..... SUCCESS
- Remove the known_hosts file on servera ..... SUCCESS
[student@workstation ~]$

Guided Exercise: Configure SSH Key-based Authentication

In this exercise, you configure a user to use key-based authentication for SSH.

Outcomes

- Generate an SSH key pair without passphrase protection.
- Generate an SSH key pair with passphrase protection.
- Authenticate with both passphrase-less and passphrase-protected SSH keys.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start ssh-configure
```

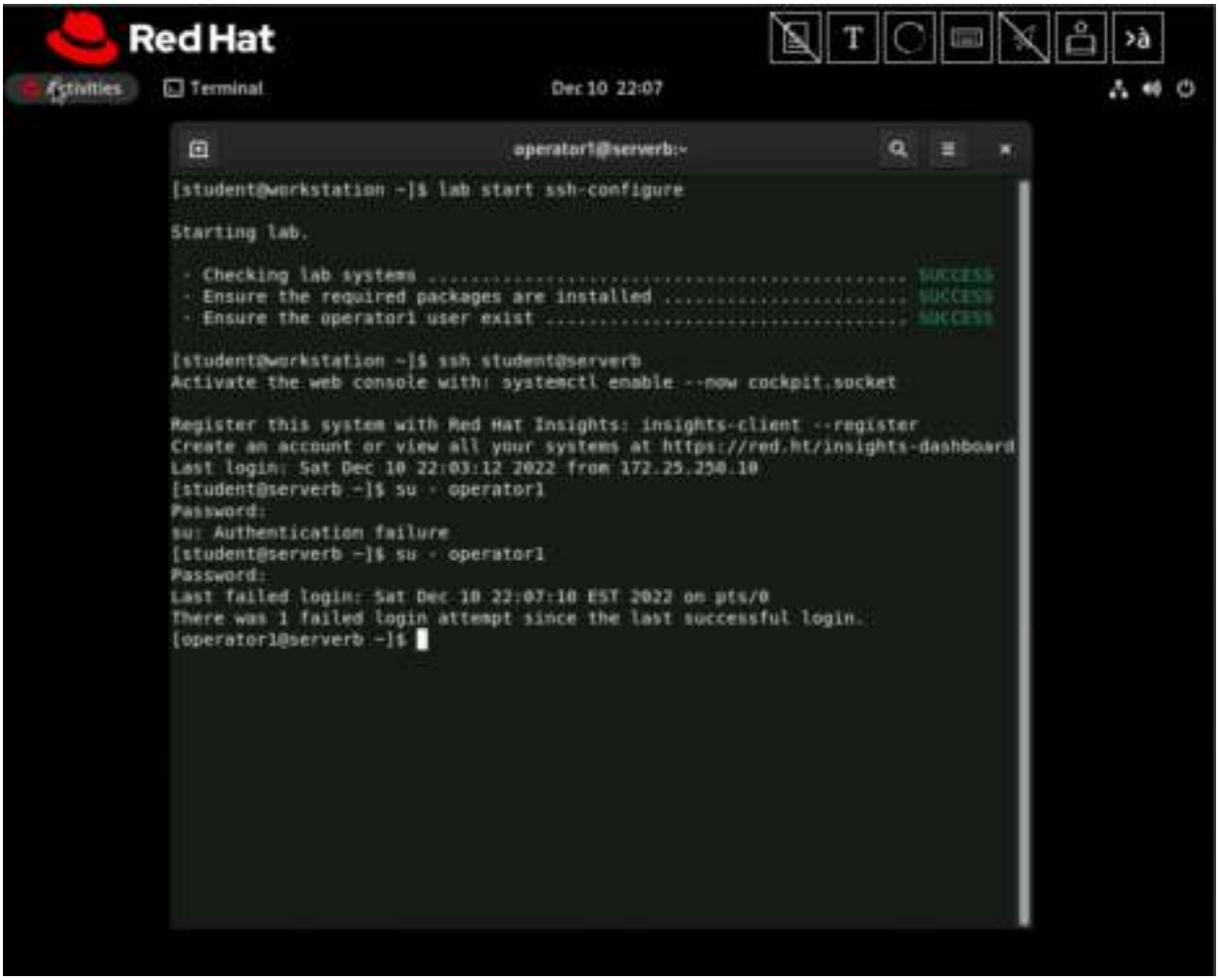
Procedure 10.2. Instructions

1. Log in to the `serverb` machine as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. Switch to the `operator1` user on the `serverb` machine. Use `redhat` as the password.

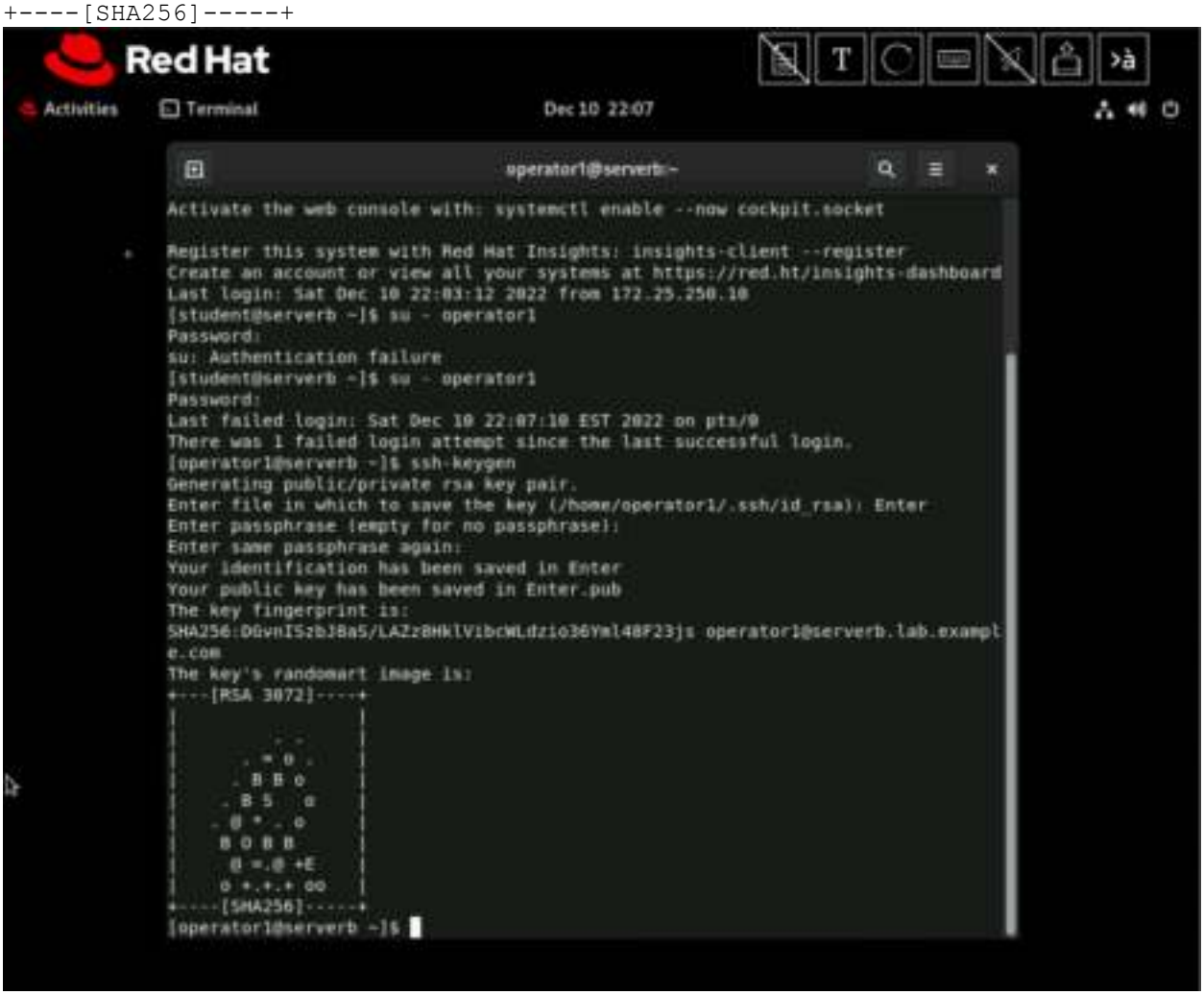
```
[student@serverb ~]$ su - operator1
Password: redhat
[operator1@serverb ~]$
```



3. Generate a set of SSH keys. Do not enter a passphrase.

```
[operator1@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator1/.ssh/id_rsa):
Enter
Created directory '/home/operator1/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator1/.ssh/id_rsa.
Your public key has been saved in /home/operator1/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:JainiQdnRosC+xXh operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|E+*+ooo .        |
|.= o.o o .       |
|o.. = . . o      |
|+. + * . o .     |
|+ = X . S +      |
| + @ +   = .     |
|. + =   o        |
```

```
|.o . . . .       |
|o     o..        |
+----[SHA256]-----+
```



4. Send the public key of the SSH key pair to the `operator1` user on the `servera` machine, with `redhat` as the password.

```
[operator1@serverb ~]$ ssh-copy-id operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/operator1/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be
established.
ED25519 key fingerprint is
SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KWOTK80.
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 prompted now it is to install the new keys
operator1@servera's password: redhat
```

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

Now try logging into the machine, with:   "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
```

5. Execute the `hostname` command on the `servera` machine remotely by using the `ssh` command without accessing the remote interactive shell.

```
[operator1@serverb ~]$ ssh operator1@servera hostname
servera.lab.example.com
```

The preceding `ssh` command does not prompt you for a password because it uses the passphrase-less private key against the exported public key to authenticate as the `operator1` user on the `servera` machine. This approach is not secure because anyone who has access to the private key file can log in to the `servera` machine as the `operator1` user. The secure alternative is to protect the private key with a passphrase, which is a following step.

6. Generate another set of SSH keys with the default name and without a passphrase, overwriting the previously generated SSH key files. Try to connect to the `servera` machine by using the new SSH keys. The `ssh` command asks for a password, because it cannot authenticate with the SSH key. Run again the `ssh` command with the `-v` (verbose) option to verify it.

   Send the new public key of the SSH key pair to the `operator1` user on the `servera` machine, to replace the previous public key. Use `redhat` as the password for the `operator1` user on the `servera` machine. Execute the `hostname` command on the `servera` machine remotely by using the `ssh` command without accessing the remote interactive shell to verify that it works again.

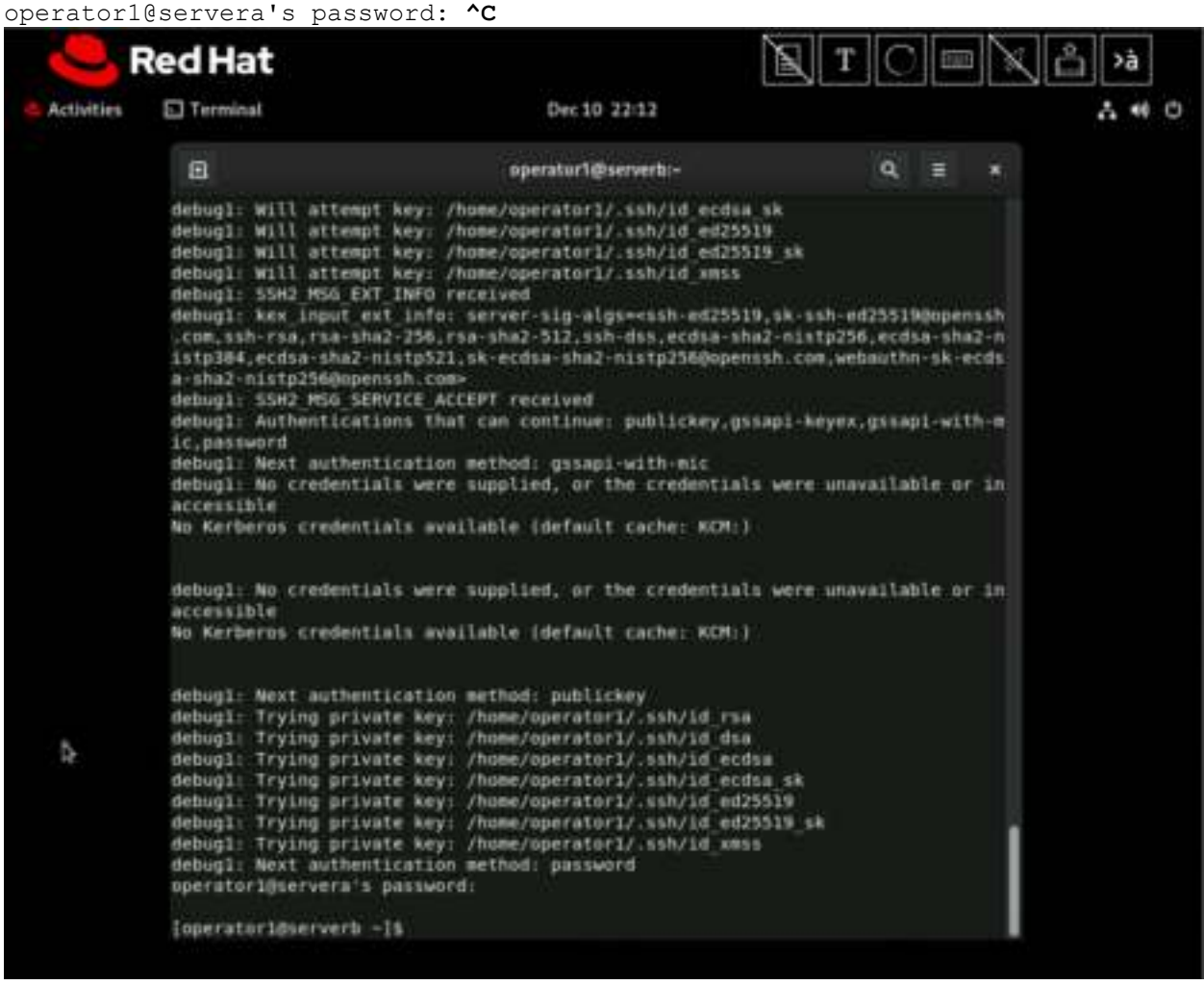
   1. Again generate another set of SSH keys with the default name and without a passphrase, overwriting the previously generated SSH key files.

      ```
      [operator1@serverb ~]$ ssh-keygen
      Generating public/private rsa key pair.
      Enter file in which to save the key
      (/home/operator1/.ssh/id_rsa): Enter
      /home/operator1/.ssh/id_rsa already exists.
      Overwrite (y/n)? y
      Enter passphrase (empty for no passphrase): Enter
      Enter same passphrase again: Enter
      Your identification has been saved in /home/operator1/.ssh/id_rsa
      Your public key has been saved in /home/operator1/.ssh/id_rsa.pub
      ...output omitted...
      ```

   2. Try to connect to the `servera` machine by using the new SSH keys. The `ssh` command asks for a password, because it cannot authenticate with the SSH key. Press **Ctrl+c** to exit from the `ssh` command when it prompts for a password. Run again the `ssh` command with the `-v` (verbose) option to verify it. Press again **Ctrl+c** to exit from the `ssh` command when it prompts for a password.

      ```
      [operator1@serverb ~]$ ssh operator1@servera hostname
      operator1@servera's password: ^C
      [operator1@serverb ~]$ ssh -v operator1@servera hostname
      OpenSSH_8.7p1, OpenSSL 3.0.1 14 Dec 2021
      debug1: Reading configuration data /etc/ssh/ssh_config
      debug1: Reading configuration data /etc/ssh/ssh_config.d/01-
      training.conf
      ...output omitted...
      debug1: Next authentication method: publickey
      debug1: Offering public key: /home/operator1/.ssh/id_rsa RSA
      SHA256:ad597Zf64xckV26xht8bjQbzqSPuOXQPXksGEWVsP80
      debug1: Authentications that can continue: publickey,gssapi-
      keyex,gssapi-with-mic,password
      debug1: Trying private key: /home/operator1/.ssh/id_dsa
      debug1: Trying private key: /home/operator1/.ssh/id_ecdsa
      debug1: Trying private key: /home/operator1/.ssh/id_ecdsa_sk
      debug1: Trying private key: /home/operator1/.ssh/id_ed25519
      debug1: Trying private key: /home/operator1/.ssh/id_ed25519_sk
      ```

```
debug1: Trying private key: /home/operator1/.ssh/id_xmss
debug1: Next authentication method: password
operator1@servera's password: ^C
```



3. Send the new public key of the SSH key pair to the `operator1` user on the `servera` machine, to replace the previous public key. Use `redhat` as the password for the `operator1` user on the `servera` machine. Execute the `hostname` command on the `servera` machine remotely by using the `ssh` command without accessing the remote interactive shell to verify that it works again.

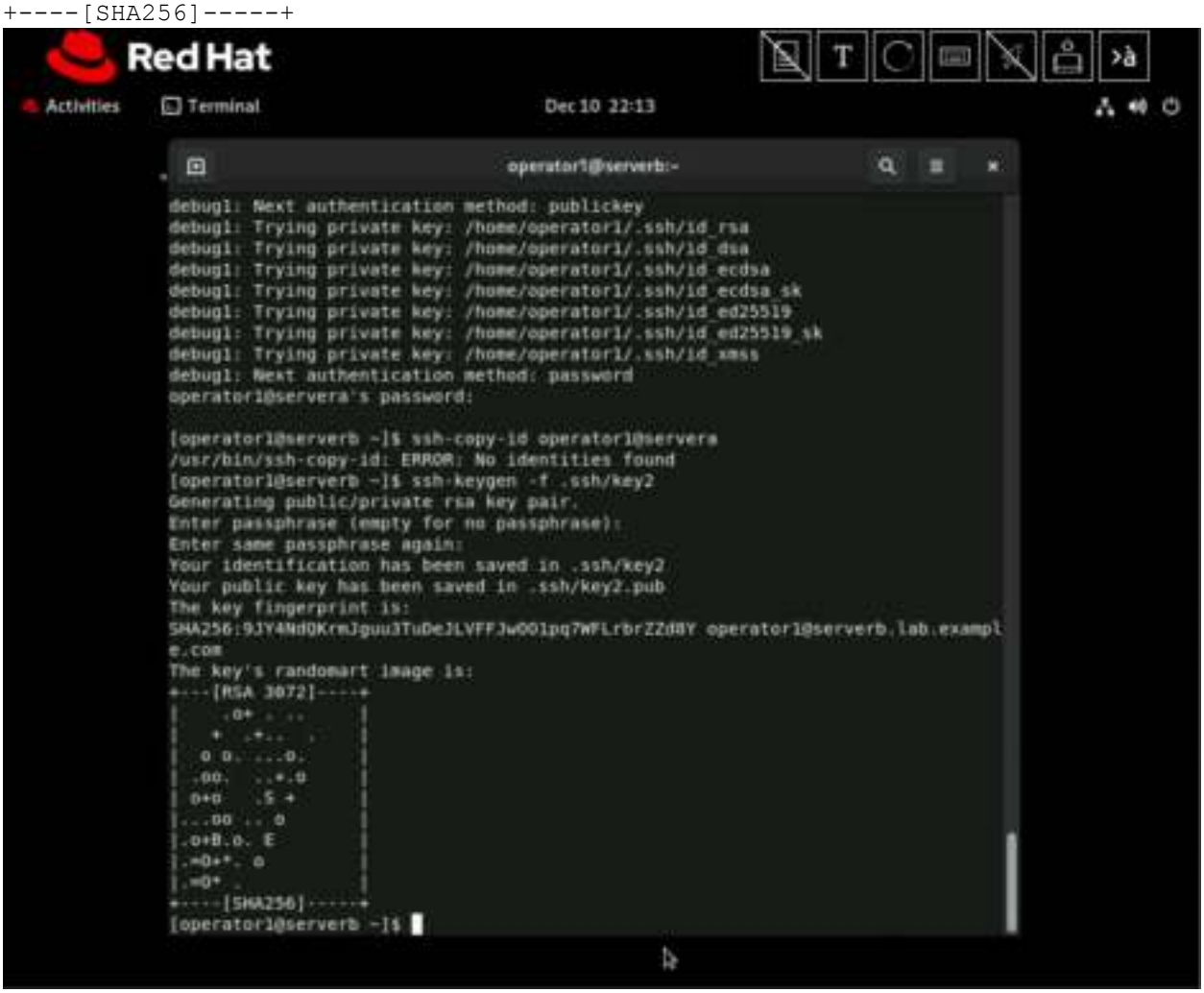
```
[operator1@serverb ~]$ ssh-copy-id operator1@servera
...output omitted...
operator1@servera's password: redhat

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh
'operator1@servera'"
and check to make sure that only the key(s) you wanted were
added.
[operator1@serverb ~]$ ssh operator1@servera hostname
servera.lab.example.com
```

7. Generate another set of SSH keys with passphrase-protection. Save the key as `/home/operator1/.ssh/key2`. Use `redhatpass` as the passphrase of the private key.

```
[operator1@serverb ~]$ ssh-keygen -f .ssh/key2
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): redhatpass
Enter same passphrase again: redhatpass
Your identification has been saved in .ssh/key2.
Your public key has been saved in .ssh/key2.pub.
The key fingerprint is:
SHA256:OCtCjfPm5QrbPBgqb operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|O=X*             |
|OB=.             |
|E*o.             |
|Booo   .         |
|..= . o S        |
| +.o   o         |
|+.oo+ o          |
|+o.O.+           |
|+ . =o.          |
+----[SHA256]-----+
```

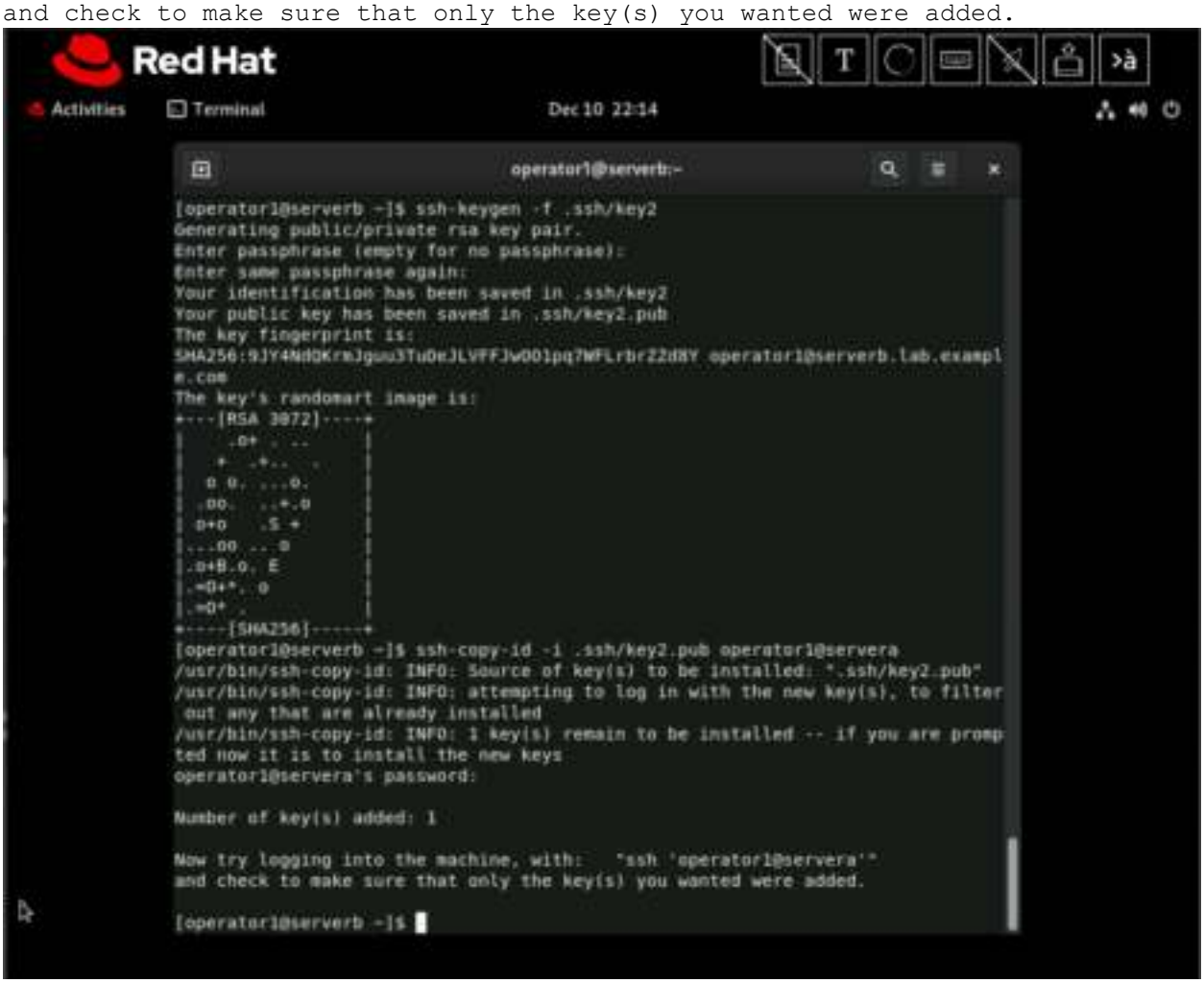


8. Send the public key of the passphrase-protected key pair to the `operator1` user on the `servera` machine. The command does not prompt you for a password because it uses the public key of the passphrase-less private key that you exported to the `servera` machine in the preceding step.

```
[operator1@serverb ~]$ ssh-copy-id -i .ssh/key2.pub operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
".ssh/key2.pub"
/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 prompted now it is to install the new keys

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
```



9. Execute the `hostname` command on the `servera` machine remotely by using the `ssh` command. Use the `/home/operator1/.ssh/key2` key as the identity file. Specify `redhatpass` as the passphrase, which you set for the private key in the preceding step.

The command prompts you for the passphrase that you used to protect the private key of the SSH key pair. If an attacker gains access to the private key, then the attacker cannot use it to access other systems because a passphrase protects the private key itself. The `ssh` command uses a different passphrase from the `operator1` user on the `servera` machine, and so users must know both.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname
Enter passphrase for key '.ssh/key2': redhatpass
servera.lab.example.com
```

Use the `ssh-agent` program, as in the following step, to avoid interactively typing the passphrase while logging in with SSH. Using the `ssh-agent` program is both more convenient and more secure when the administrators log in to remote systems regularly.

10. Run the `ssh-agent` program in your `Bash` shell and add the passphrase-protected private key (`/home/operator1/.ssh/key2`) of the SSH key pair to the shell session.

The command starts the `ssh-agent` program and configures the shell session to use it. Then, you use the `ssh-add` command to provide the unlocked private key to the `ssh-agent` program.

```
[operator1@serverb ~]$ eval $(ssh-agent)
Agent pid 1729
[operator1@serverb ~]$ ssh-add .ssh/key2
Enter passphrase for .ssh/key2: redhatpass
Identity added: .ssh/key2 (operator1@serverb.lab.example.com)
```

11. Execute the `hostname` command on the `servera` machine remotely without accessing a remote interactive shell. Use the `/home/operator1/.ssh/key2` key as the identity file.

    The command does not prompt you to enter the passphrase interactively.

    ```
    [operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname
    servera.lab.example.com
    ```

12. Open another terminal on the `workstation` machine and log in to the `serverb` machine as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

13. On the `serverb` machine, switch to the `operator1` user and remotely log in to the `servera` machine. Use the `/home/operator1/.ssh/key2` key as the identity file to authenticate using the SSH keys.
    1. Use the `su` command to switch to the `operator1` user. Use `redhat` as the password for the `operator1` user.

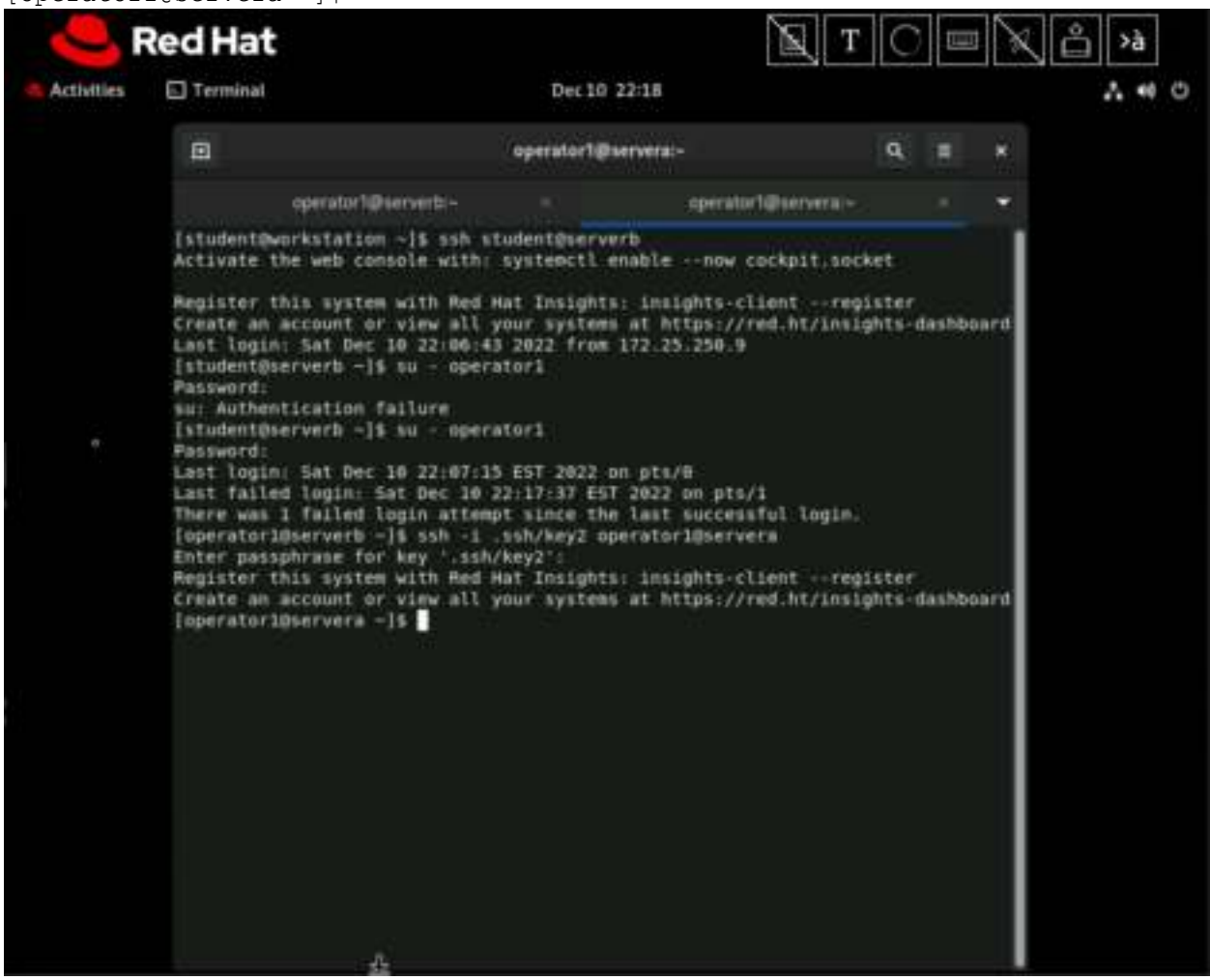
       ```
       [student@serverb ~]$ su - operator1
       Password: redhat
       [operator1@serverb ~]$
       ```

    2. Log in to the `servera` machine as the `operator1` user.

       The command prompts you to enter the passphrase interactively because you do not invoke the SSH connection from the same shell where you started the `ssh-agent` program.

       ```
       [operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera
       Enter passphrase for key '.ssh/key2': redhatpass
       ...output omitted...
       ```

```
[operator1@servera ~]$
```



14. Exit and close all extra terminals and return to the `workstation` machine.
    1. Exit and close extra terminal windows. The `exit` commands leave the `operator1` user's shell; terminate the shell session where `ssh-agent` is active; and return to the `student` user's shell on the `serverb` machine.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[operator1@serverb ~]$
```

    2. Return to the `workstation` system as the `student` user.

```
[operator1@serverb ~]$ exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ssh-configure
```

This concludes the section.

Guided Exercise: Customize OpenSSH Service Configuration

In this exercise, you disable direct logins as `root` and disable password-based authentication for the OpenSSH service on one of your servers.

Outcomes

- Disable direct logins as `root` over `ssh`.
- Disable password-based authentication for remote users to connect over SSH.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start ssh-customize
```

Procedure 10.3. Instructions

1. From `workstation`, open an SSH session to the `serverb` machine as the `student` user.

```
[student@workstation ~]$ ssh student@serverb
[student@serverb ~]$
```

2. Use the `su` command to switch to the `operator2` user on the `serverb` machine. Use `redhat` as the password for the `operator2` user.

```
[student@serverb ~]$ su - operator2
Password: redhat
[operator2@serverb ~]$
```

3. Use the `ssh-keygen` command to generate SSH keys. Do not enter any passphrase for the keys.

```
[operator2@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator2/.ssh/id_rsa):
Enter
Created directory '/home/operator2/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator2/.ssh/id_rsa.
Your public key has been saved in /home/operator2/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:LN5x1irX0OWxgyd/qhATNgZWOtLUj16EZkM1JHkCR+I
operator2@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|         *=+     |
|        = =O.o.    |
|       . Eo=B  o   |
|        o +.=o+ o  |
|       . S..= =  |
```

```
|      . o +. + . |
|       . o + . . .|
|          o .   o |
|             ...  |
+----[SHA256]-----+
```

4. Use the `ssh-copy-id` command to send the public key of the SSH key pair to the `operator2` user on the `servera` machine. Use `redhat` as the password for the `operator2` user on `servera`.

```
[operator2@serverb ~]$ ssh-copy-id operator2@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/operator2/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be
established.
ED25519 key fingerprint is
SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KWOTK80.
Are you sure you want to continue connecting (yes/no)? 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 prompted now it is to install the new keys
```

```
operator2@servera's password: redhat
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'operator2@servera'"
and check to make sure that only the key(s) you wanted were added.
```

5. Confirm that you can successfully log in to the `servera` machine as the `operator2` user with the SSH keys.
    1. Open an SSH session to the `servera` machine as the `operator2` user.

        ```
        [operator2@serverb ~]$ ssh operator2@servera
        ...output omitted...
        [operator2@servera ~]$
        ```

        The preceding `ssh` command used SSH keys for authentication.

    2. Log out of `servera`.

        ```
        [operator2@servera ~]$ exit
        logout
        Connection to servera closed.
        ```

6. Confirm that you can successfully log in to the `servera` machine as the `root` user with `redhat` as the password.
    1. Open an SSH session to the `servera` machine as the `root` user with `redhat` as the password.

```
[operator2@serverb ~]$ ssh root@servera
root@servera's password: redhat
...output omitted...
[root@servera ~]#
```

The preceding `ssh` command used the password of the superuser for authentication because SSH keys do not exist for the superuser.

    2. Log out of the `servera` machine.

```
[root@servera ~]# exit
logout
Connection to servera closed.
[operator2@serverb ~]$
```

7. Confirm that you can successfully log in to the `servera` machine as the `operator3` user with `redhat` as the password.
    1. Open an SSH session to the `servera` machine as the `operator3` user with `redhat` as the password.

```
[operator2@serverb ~]$ ssh operator3@servera
operator3@servera's password: redhat
...output omitted...
[operator3@servera ~]$
```

The preceding `ssh` command used the password of the `operator3` user for authentication because SSH keys do not exist for the `operator3` user.

    2. Log out of the `servera` machine.

```
[operator3@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$
```

8. Configure the `sshd` service on the `servera` machine to prevent users from logging in as the `root` user. Use `redhat` as the password of the superuser when required.
    1. Open an SSH session to the `servera` machine as the `operator2` user with the SSH keys.

        ```
        [operator2@serverb ~]$ ssh operator2@servera
        ...output omitted...
        [operator2@servera ~]$
        ```

    2. On the `servera` machine, switch to the `root` user. Use `redhat` as the password for the `root` user.

        ```
        [operator2@servera ~]$ su -
        Password: redhat
        [root@servera ~]#
        ```

    3. Set `PermitRootLogin` to `no` in the `/etc/ssh/sshd_config` file and reload the `sshd` service. You can use the `vim /etc/ssh/sshd_config` command to edit the configuration file of the `sshd` service.

        ```
        ...output omitted...
        PermitRootLogin no
        ...output omitted...
        [root@servera ~]# systemctl reload sshd
        ```

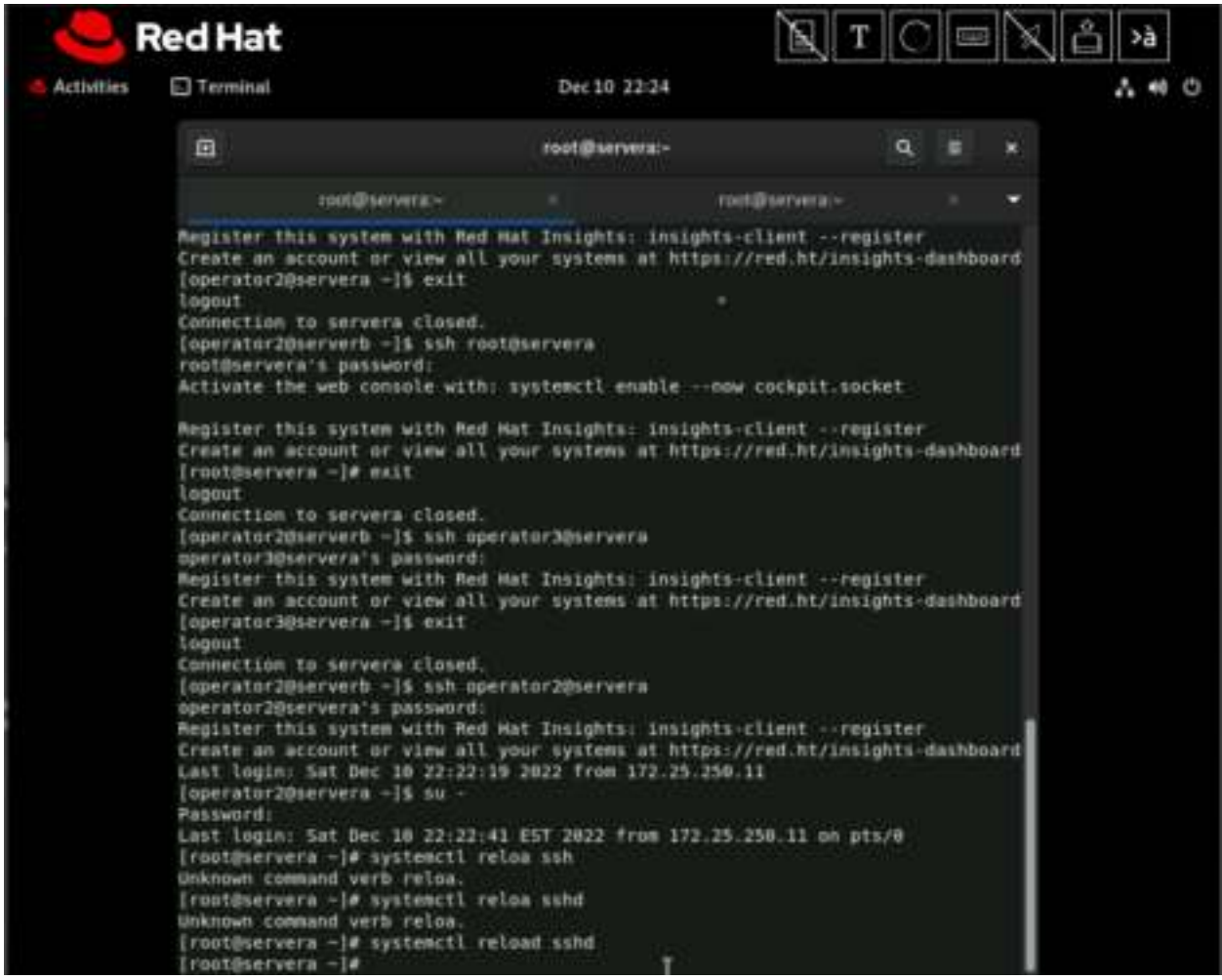
    4. Open another terminal on `workstation` and open an SSH session to the `serverb` machine as the `operator2` user. From the `serverb` machine, try to log in to the `servera` machine as the `root` user. This command should fail because you disabled the `root` user login over SSH in the preceding step.

        Note

        For your convenience, password-less login is already configured between `workstation` and `serverb` in the classroom environment.

        ```
        [student@workstation ~]$ ssh operator2@serverb
        ...output omitted...
        [operator2@serverb ~]$ ssh root@servera
        root@servera's password: redhat
        Permission denied, please try again.
        root@servera's password: redhat
        Permission denied, please try again.
        root@servera's password: redhat
        root@servera: Permission denied (publickey,gssapi-keyex,gssapi-
        with-mic,password).
        ```

        By default, the `ssh` command attempts to authenticate with key-based authentication first, and if that method fails, then with password-based authentication.



9. Configure the `sshd` service on the `servera` machine to allow users to authenticate with SSH keys only, rather than with their passwords.
    1. Return to the first terminal with the `root` user's active shell on the `servera` machine. Set the `PasswordAuthentication` parameter to `no` in the `/etc/ssh/sshd_config` file and reload the `sshd` service. You can use the `vim /etc/ssh/sshd_config` command to edit the configuration file of the `sshd` service.

        ```
        ...output omitted...
        PasswordAuthentication no
        ...output omitted...
        [root@servera ~]# systemctl reload sshd
        ```

    2. Go to the second terminal with the `operator2` user's active shell on the `serverb` machine, and try to log in to the `servera` machine as the `operator3` user. This command should fail because SSH keys are not configured for the `operator3` user, and the `sshd` service on the `servera` machine does not allow the use of passwords for authentication.

```
[operator2@serverb ~]$ ssh operator3@servera
operator3@servera: Permission denied (publickey,gssapi-
keyex,gssapi-with-mic).
```

Note

For more granularity, you can use the explicit `-o PubkeyAuthentication=no` and `-o PasswordAuthentication=yes` options with the `ssh` command. You can then override the `ssh` command's defaults and confidently determine that the preceding command fails based on the settings that you adjusted in the `/etc/ssh/sshd_config` file in the preceding step.

3. Return to the first terminal with the `root` user's active shell on the `servera` machine. Verify that `PubkeyAuthentication` is enabled in the `/etc/ssh/sshd_config` file. You can use the `vim /etc/ssh/sshd_config` command to view the configuration file of the `sshd` service.

```
...output omitted...
#PubkeyAuthentication yes
...output omitted...
```

The `PubkeyAuthentication` line is commented. Any commented line in this file uses the default value. Commented lines indicate the default values of a parameter. The public key authentication of SSH is active by default, as the commented line indicates.

4. Return to the second terminal with the `operator2` user's active shell on the `serverb` machine and try to log in to the `servera` machine as the `operator2` user. This command should succeed because the SSH keys are configured for the `operator2` user to log in to the `servera` machine from the `serverb` machine.

```
[operator2@serverb ~]$ ssh operator2@servera
...output omitted...
[operator2@servera ~]$
```

5. From the second terminal, exit the `operator2` user's shell on both the `servera` and `serverb` machines.

```
[operator2@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

6. Close the second terminal on the `workstation` machine.

```
[student@workstation ~]$ exit
```

7. From the first terminal, exit the `root` user's shell on the `servera` machine.

```
[root@servera ~]# exit
logout
```

8. From the first terminal, exit the `operator2` user's shell on both the `servera` and `serverb` machines.

```
[operator2@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$ exit
logout
[student@serverb ~]$
```

9. Log out of `serverb` and return to the `student` user's shell on `workstation`.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ssh-customize
```

This concludes the section.

Lab: Configure and Secure SSH

In this lab, you set up key-based authentication for users, and disable direct login as `root` and password authentication for all users for the OpenSSH service on one of your servers.

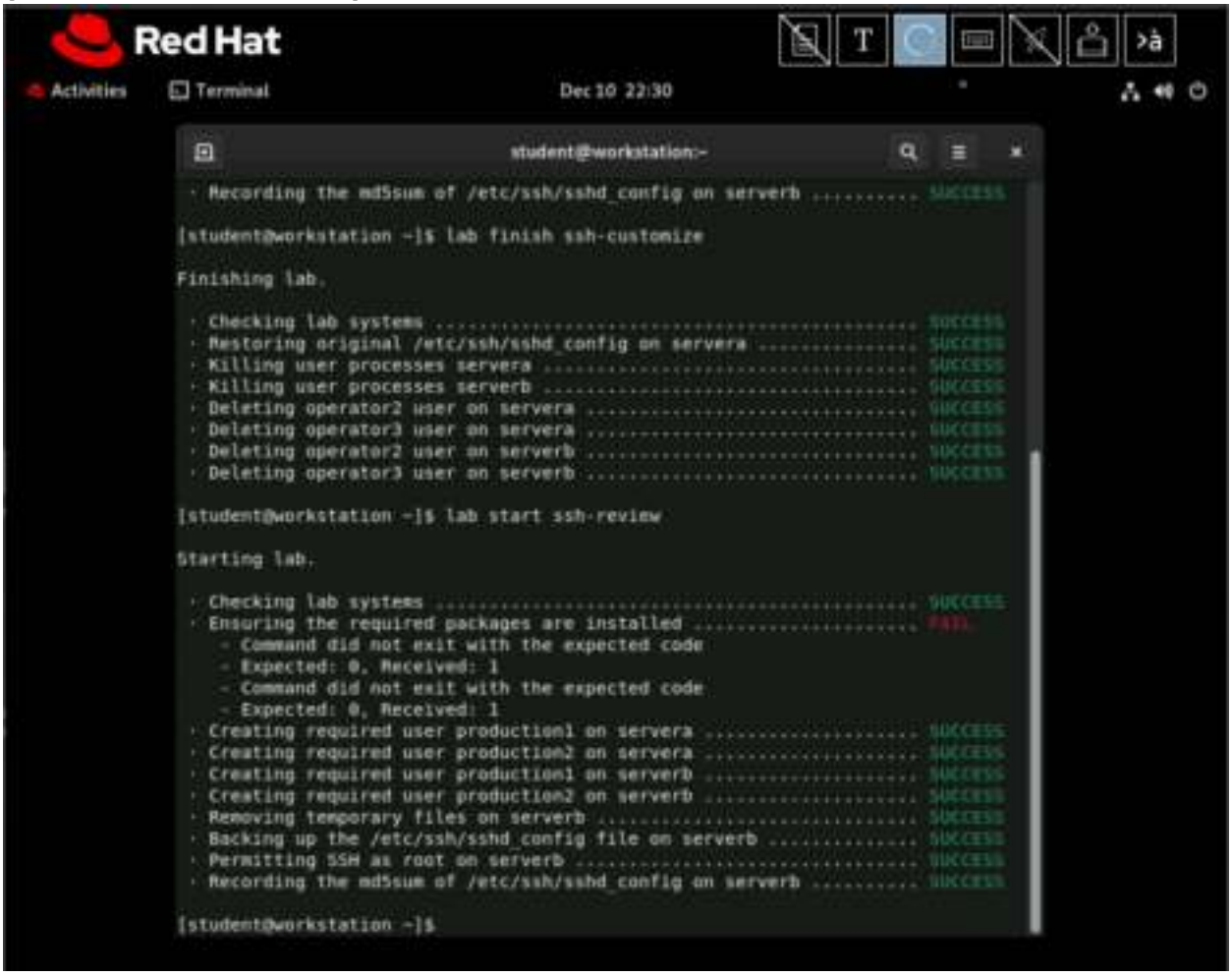
Outcomes

- Authenticate with SSH keys.
- Prevent users from directly logging in as the `root` user over the `ssh` service.
- Prevent users from logging in to the system with SSH password-based authentication.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start ssh-review
```



Procedure 10.4. Instructions

1. From the `workstation` machine, log in to the `servera` machine as the `student` user.

2. Switch to the `production1` user on the `servera` machine. Enter `redhat` as the password.

3. Generate passphrase-less SSH keys for the `production1` user on the `servera` machine.

4. Send the public key of the SSH key pair to the `production1` user on the `serverb` machine.

5. Verify that the `production1` user can successfully log in to the `serverb` machine with the SSH keys.

6. Configure the `sshd` service on `serverb` to prevent users from logging in as the `root` user. Use `redhat` as the `root` password.

7. Configure the `sshd` service on the `serverb` machine to allow users to authenticate with SSH keys only, rather than with their passwords.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade ssh-review
```

Finish

On the `workstation` machine, change to the `student` user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish ssh-review
```

This concludes the section.