# C4 M2 L5 Qwiklab: Service Management in Linux

1 hour 1 Credit

Rate Lab

## Introduction

As a system administrator, you will need to know how to check the status of a running service, and how to stop, start and restart running services. You'll also need to know how to configure services and fix problems you encounter while running them.

In this lab, you'll look at the list of services running on a Linux machine. You will practice stopping and starting some and query their status.

You'll also fix a problem in a service that is failing to start, and edit the configuration of another service.

**Head's up**: Make sure to click the **"Start Lab"** button at the top of the screen. It may take a while for the lab to load. Please wait until the lab is running. To mark this lab as completed, make sure to click **"End Lab"** when you're done!

**You'll have 60 minutes to complete this lab.**

### Start the lab

You'll need to start the lab before you can access the materials in the virtual machine OS. To do this, click the green “Start Lab” button at the top of the screen.

**Note:** For this lab you are going to access the **Linux VM** through your **local SSH Client**, and not use the **Google Console** (**Open GCP Console** button is not available for this lab).

After you click the “Start Lab” button, you will see all the SSH connection details on the left-hand side of your screen. You should have a screen that looks like this:


# Accessing the virtual machine

Please find one of the three relevant options below based on your device's operating system.

**Note:** Working with Qwiklabs may be similar to the work you'd perform as an **IT Support Specialist**; you'll be interfacing with a cutting-edge technology that requires multiple steps to access, and perhaps healthy doses of patience and persistence(!). You'll also be using **SSH** to enter the labs -- a critical skill in IT Support that you’ll be able to practice through the labs.

## Option 1: Windows Users: Connecting to your VM

In this section, you will use the PuTTY Secure Shell (SSH) client and your VM’s External IP address to connect.

**Download your PPK key file**

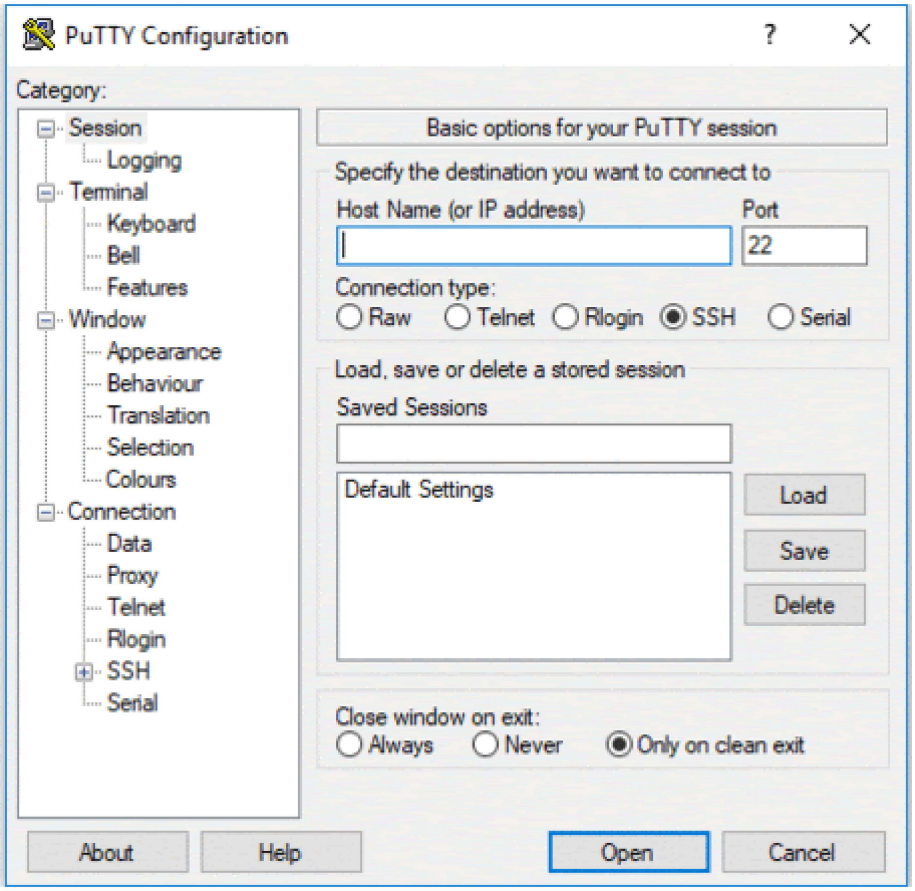
You can download the VM’s private key file in the PuTTY-compatible **PPK** format from the Qwiklabs Start Lab page. Click on **Download PPK**.


**Connect to your VM using SSH and PuTTY**

1. You can download Putty from here

2. In the **Host Name (or IP address)** box, enter username@external_ip_address.

**Note:** Replace **username** and **external_ip_address** with values provided in the lab.



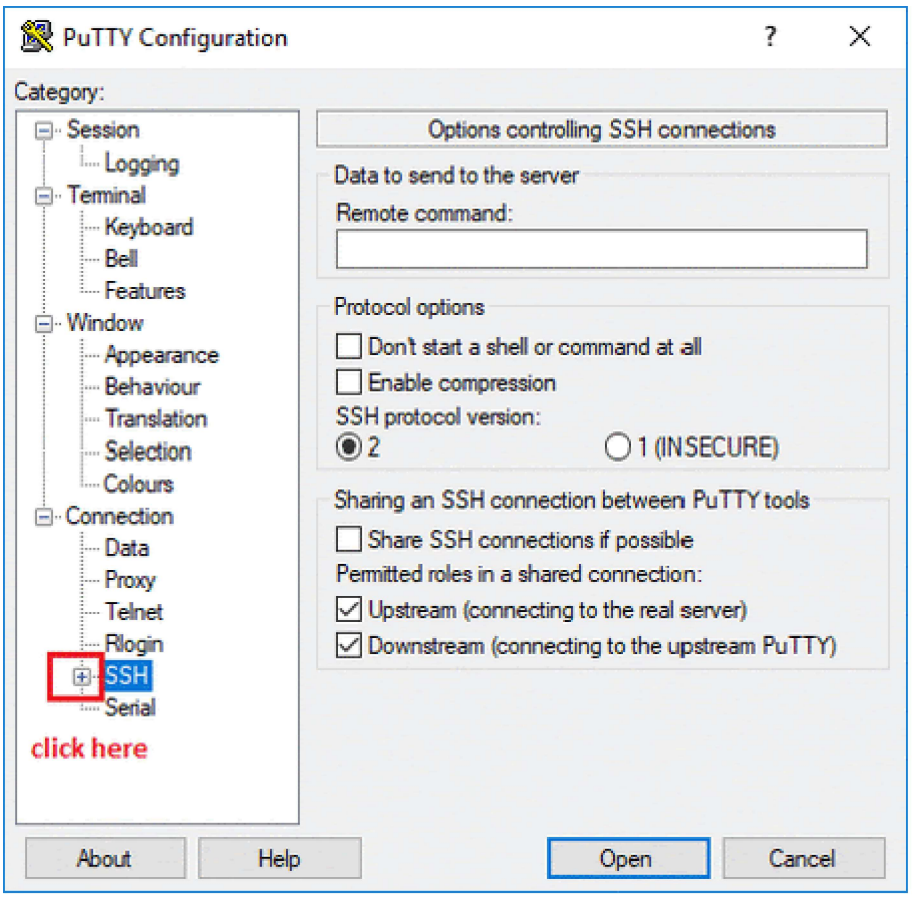

3. In the **Category** list, expand **SSH**.

4. Click **Auth** (don't expand it).

5. In the **Private key file for authentication** box, browse to the PPK file that you downloaded and double-click it.

6. Click on the **Open** button.

**Note:** PPK file is to be imported into PuTTY tool using the Browse option available in it. It should not be opened directly but only to be used in PuTTY.




7. Click **Yes** when prompted to allow a first connection to this remote SSH server. Because you are using a key pair for authentication, you will not be prompted for a password.

**Common issues**

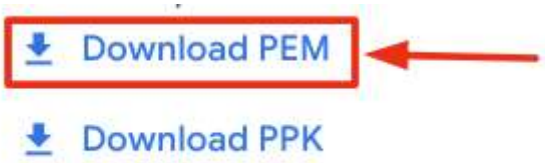
If PuTTY fails to connect to your Linux VM, verify that:

- You entered **<username>@<external ip address>** in PuTTY.
- You downloaded the fresh new PPK file for this lab from Qwiklabs.
- You are using the downloaded PPK file in PuTTY.

## Option 2: OSX and Linux users: Connecting to your VM via SSH

**Download your VM's private key file.**

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.




**Connect to the VM using the local Terminal application**

A **terminal** is a program which provides a **text-based interface for typing commands**. Here you will use your terminal as an SSH client to connect with lab provided Linux VM.

1. Open the Terminal application.
    - To open the terminal in Linux use the shortcut key **Ctrl+Alt+t**.
    - To open terminal in **Mac** (OSX) enter **cmd + space** and search for **terminal**.
2. Enter the following commands.

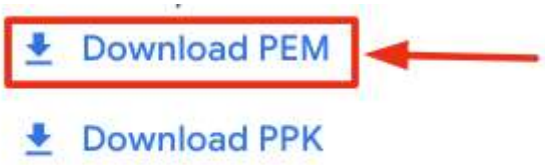
**Note:** Substitute the **path/filename for the PEM** file you downloaded, **username** and **External IP Address**.
You will most likely find the PEM file in **Downloads**. If you have not changed the download settings of your system, then the path of the PEM key will be **~/Downloads/qwikLABS-XXXXX.pem**

```
chmod 600 ~/Downloads/qwikLABS-XXXXX.pem
ssh -i ~/Downloads/qwikLABS-XXXXX.pem username@External Ip Address
```


## Option 3: Chrome OS users: Connecting to your VM via SSH

**Note:** Make sure you are not in **Incognito/Private mode** while launching the application.
**Download your VM's private key file.**

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.




**Connect to your VM**

1. Add Secure Shell from here to your Chrome browser.
2. Open the Secure Shell app and click on **[New Connection]**.


3. In the **username** section, enter the username given in the Connection Details Panel of the lab. And for the **hostname** section, enter the external IP of your VM instance that is mentioned in the Connection Details Panel of the lab.


4. In the **Identity** section, import the downloaded PEM key by clicking on the **Import…** button beside the field. Choose your PEM key and click on the **OPEN** button.

**Note:** If the key is still not available after importing it, refresh the application, and select it from the **Identity** drop-down menu.

5. Once your key is uploaded, click on the **[ENTER] Connect** button below.


6. For any prompts, type **yes** to continue.

7. You have now successfully connected to your Linux VM.

You're now ready to continue with the lab!

## Linux commands reminder

In this lab, we'll use a number of Linux commands that were already explained during Course 3. Here is a reminder of what these commands do:

- `sudo <command>`: executes a command with administrator rights
- `ls <directory>`: lists the files in a directory
- `mv <old_name> <new_name>`: moves or renames a file from the old name to the new name
- `tail <file>`: shows the last lines of a file
- `cat <file>`: prints the whole contents of a file
- `grep <pattern> <file>`: filters the text of a file according to the pattern
- `less <file>`: lets you browse a file

Additionally, you can combine these commands using the `|` sign. For example:

`sudo cat /var/log/syslog | grep error | tail`
will first print the output of `/var/log/syslog`, then keep only the lines that say "error" and then the last 10 lines of that output.

We will also present a number of new commands such as `systemctl`, `service`, `logger` and `date`. We will briefly explain what these commands do when they are shown. Remember that you can always read the manual page using `man <command_name>` to learn more about a command.

While you can copy and paste the commands that are presented in this lab, we recommend typing them out manually, to help with understanding and remembering them.

## Listing system services

Let's look at the services that are installed in the machine. In order to do this, we will use the `systemctl` command (ctl stands for "control").

If you run `systemctl`, with no parameters, it will list all services that are known to their system. The columns shown are: the name of the file that defines the service, whether they are loaded or not, whether they are active or inactive, what their state is (running, exited, failed) and finally, the long name for the service. You can scroll this list using the PageUp and PageDown keys on your keyboard, and use ‘q' to quit.

If you are interested in seeing only the services that are running, you can use the following command:

```
sudo systemctl --state=running
```

This will show a shorter list, although it will still include quite a number of services. Again, you can scroll with PageUp and PageDown and use ‘q' to quit.

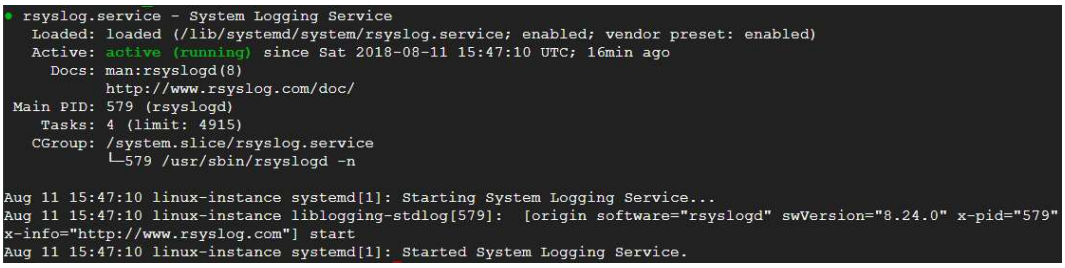
# Stopping and starting services

Alright, now that we've listed the services let's practice stopping and starting some of them. The first service that we are going to stop is the `rsyslog` service. This service is in charge of writing content to the log files, as in `/var/log/syslog`, `/var/log/kern.log`, `/var/log/auth.log` and others. Processes that generate output will send that output to the rsyslog service and the service will write it to the corresponding log files depending on how the system was configured.

Let's first start by checking the status of the service. We do this by using the `service` command with the `status` action:

```
sudo service rsyslog status
```

**Output**:




This is showing us a lot of information about the service: it's loaded (which means that the OS has the information about the service in memory), it's enabled (which means that it will start automatically on boot), it's active and running. It also tells

us where to find some documentation about the service and more. Finally, it shows us the last log lines that this service generated.

We can see this service in action by using the `logger` command:

```
logger This is a test log entry
```

The logger command will send the text to the rsyslog service and the service will then write it into `/var/log/syslog`. We can check that this is the case by looking at the last lines in `/var/log/syslog`.

```
sudo tail -1 /var/log/syslog
```

**Output**:

```
gcpstaging21031_student@linux-instance:~$ tail -1 /var/log/syslog
Aug 11 16:06:05 linux-instance gcpstaging21031_student: This is a test log entry
gcpstaging21031_student@linux-instance:~$
```

Let's now go ahead and stop the rsyslog service:

```
sudo service rsyslog stop
```

We need to execute the command with sudo, because while all users can query the status of services, only users with administrator rights can stop or start services.

We get no output from the command. This is a common behavior for many Linux commands. When you ask the system to do something, many commands will just perform the action without generating any output unless there's an error.

To see the current state, we can query the status of the service again:

```
sudo service rsyslog status
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ sudo service rsyslog status
● rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Fri 2018-08-17 16:08:06 UTC; 5s ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 1171 (code=exited, status=0/SUCCESS)

Aug 17 16:04:11 linux-instance systemd[1]: Starting System Logging Service...
Aug 17 16:04:12 linux-instance systemd[1]: Started System Logging Service.
Aug 17 16:08:06 linux-instance systemd[1]: Stopping System Logging Service...
Aug 17 16:08:06 linux-instance systemd[1]: Stopped System Logging Service.
Aug 17 16:08:06 linux-instance systemd[1]: Stopped System Logging Service.
gcpstaging21306_student@linux-instance:~$
```

We see that the service is now stopped. We can also see what the command logged to `/var/log/syslog` when finishing:

```
sudo tail -5 /var/log/syslog
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ tail -5 /var/log/syslog
Aug 17 16:05:57 linux-instance systemd[7020]: Reached target Default.
Aug 17 16:05:57 linux-instance systemd[7020]: Startup finished in 170ms.
Aug 17 16:05:57 linux-instance systemd[1]: Started User Manager for UID 1001.
Aug 17 16:07:54 linux-instance gcpstaging21306_student: This is a test log entry
Aug 17 16:08:06 linux-instance rsyslogd: [origin software="rsyslogd" swVersion="8.16.0" x-pid="1171" x-info="ht
tp://www.rsyslog.com"] exiting on signal 15.
gcpstaging21306_student@linux-instance:~$
```

In the last line, we see that the rsyslog service has exited and is no longer running.

We can try sending text with our logger command again:

```
logger This is another test log entry
```

And then check that the contents of `/var/log/syslog`:

```
sudo tail /var/log/syslog
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ tail /var/log/syslog
Aug 17 16:05:57 linux-instance systemd[1]: Started Session 1 of user gcpstaging21306_student.
Aug 17 16:05:57 linux-instance systemd[7020]: Reached target Timers.
Aug 17 16:05:57 linux-instance systemd[7020]: Reached target Paths.
Aug 17 16:05:57 linux-instance systemd[7020]: Reached target Sockets.
Aug 17 16:05:57 linux-instance systemd[7020]: Reached target Basic System.
Aug 17 16:05:57 linux-instance systemd[7020]: Reached target Default.
Aug 17 16:05:57 linux-instance systemd[7020]: Startup finished in 170ms.
Aug 17 16:05:57 linux-instance systemd[1]: Started User Manager for UID 1001.
Aug 17 16:07:54 linux-instance gcpstaging21306_student: This is a test log entry
Aug 17 16:08:06 linux-instance rsyslogd: [origin software="rsyslogd" swVersion="8.16.0" x-pid="1171" x-info="ht
tp://www.rsyslog.com"] exiting on signal 15.
gcpstaging21306_student@linux-instance:~$
```

We can see that nothing was logged, because rsyslog wasn't running. Let's start it back up:

```
sudo service rsyslog start
```

Again, no output, this is expected. We can check the status:

```
sudo service rsyslog status
```

**Output**:

```
gcpstaging21041_student@linux-instance:~$ sudo service rsyslog status
● rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2018-08-11 23:24:10 UTC; 16s ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 6723 (rsyslogd)
    Tasks: 4 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─6723 /usr/sbin/rsyslogd -n

Aug 11 23:24:09 linux-instance systemd[1]: Starting System Logging Service...
Aug 11 23:24:09 linux-instance liblogging-stdlog[6723]:  [origin software="rsyslogd" swVersion="8.24.0"
x-pid="6723" x-info="http://www.rsyslog.com"] start
Aug 11 23:24:10 linux-instance systemd[1]: Started System Logging Service.
gcpstaging21041_student@linux-instance:~$
```

And see that it's running again. Let's try our logger command one more time:

```
logger This is another test log entry
```

And then check that the contents of `/var/log/syslog`:

```
sudo tail /var/log/syslog
```

**Output**:

```
gcpstaging21041_student@linux-instance:~$ tail -5 /var/log/syslog
Aug 11 23:24:10 linux-instance systemd[1]: Listening on Syslog Socket.
Aug 11 23:24:10 linux-instance systemd[1]: Starting System Logging Service...
Aug 11 23:24:09 linux-instance liblogging-stdlog:  [origin software="rsyslogd" swVersion="8.24.0" x-pid=
"6723" x-info="http://www.rsyslog.com"] start
Aug 11 23:24:10 linux-instance systemd[1]: Started System Logging Service.
Aug 11 23:25:08 linux-instance gcpstaging21041_student: This is another test log entry
gcpstaging21041_student@linux-instance:~$
```

The message is now there!

Click *Check my progress* to verify the objective. Stop and Start rsyslog Service

# Fixing a failing service

We can list services that are in a failed state with the systemctl command:

```
sudo systemctl --state=failed
```

**Output**:

```
gcpstaging21041_student@linux-instance:~$ systemctl --state=failed
  UNIT          LOAD   ACTIVE SUB    DESCRIPTION
● cups.service loaded failed failed CUPS Scheduler

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

1 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
gcpstaging21041_student@linux-instance:~$
```

We see here that the cups service is in a failed state. This is the service used to manage printers on Linux systems. We can get more information about this failure by checking the status:

```
sudo service cups status
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ sudo service cups status
● cups.service - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Fri 2018-08-17 16:05:05 UTC; 9min ago
     Docs: man:cupsd(8)
  Process: 6961 ExecStart=/usr/sbin/cupsd -l (code=exited, status=1/FAILURE)
 Main PID: 6961 (code=exited, status=1/FAILURE)

Aug 17 16:05:05 linux-instance systemd[1]: Started CUPS Scheduler.
Aug 17 16:05:05 linux-instance cupsd[6961]: Unable to open "/etc/cups/cupsd.conf" - No such file or directory
Aug 17 16:05:05 linux-instance systemd[1]: cups.service: Main process exited, code=exited, status=1/FAILURE
Aug 17 16:05:05 linux-instance systemd[1]: cups.service: Unit entered failed state.
Aug 17 16:05:05 linux-instance systemd[1]: cups.service: Failed with result 'exit-code'.
gcpstaging21306_student@linux-instance:~$
```

In the log lines that we get, we see that the process failed to start. It's telling us that it's unable to find `/etc/cups/cupsd.conf`, which is the location where the configuration of this service is located.

So, let's look at the contents of that directory:

```
sudo ls -l /etc/cups
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ ls -l /etc/cups
total 56
-rw-r--r-- 1 root root 15303 May  7 18:07 cups-browsed.conf
-rw-r--r-- 1 root root  4630 Aug 17 16:04 cupsd.conf.old
-rw-r--r-- 1 root root  2931 Jun 22 18:30 cups-files.conf
drwxr-xr-x 2 root root  4096 Jun 22 18:30 interfaces
drwxr-xr-x 2 root lp    4096 Jun 22 18:30 ppd
-rw-r--r-- 1 root root   240 Aug 17 16:05 raw.convs
-rw-r--r-- 1 root root   211 Aug 17 16:05 raw.types
-rw-r--r-- 1 root root   142 Jun 22 18:30 snmp.conf
drwx------ 2 root lp    4096 Aug 17 16:04 ssl
-rw-r----- 1 root lp      90 Aug 17 16:05 subscriptions.conf
gcpstaging21306_student@linux-instance:~$
```

There's no `cupsd.conf`, but there is `cupsd.conf.old`. Apparently the configuration file was deleted. Good thing we kept a copy! Let's move that file so that cups can find it and start successfully:

```
sudo mv /etc/cups/cupsd.conf.old /etc/cups/cupsd.conf
```

As with the other commands, we get no output after executing this. We can run `ls` again to see that the file was renamed correctly:

```
sudo ls -l /etc/cups
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ ls -l /etc/cups
total 56
-rw-r--r-- 1 root root 15303 May  7 18:07 cups-browsed.conf
-rw-r--r-- 1 root root  4630 Aug 17 16:04 cupsd.conf
-rw-r--r-- 1 root root  2931 Jun 22 18:30 cups-files.conf
drwxr-xr-x 2 root root  4096 Jun 22 18:30 interfaces
drwxr-xr-x 2 root lp    4096 Jun 22 18:30 ppd
-rw-r--r-- 1 root root   240 Aug 17 16:05 raw.convs
-rw-r--r-- 1 root root   211 Aug 17 16:05 raw.types
-rw-r--r-- 1 root root   142 Jun 22 18:30 snmp.conf
drwx------ 2 root lp    4096 Aug 17 16:04 ssl
-rw-r----- 1 root lp      90 Aug 17 16:05 subscriptions.conf
gcpstaging21306_student@linux-instance:~$
```

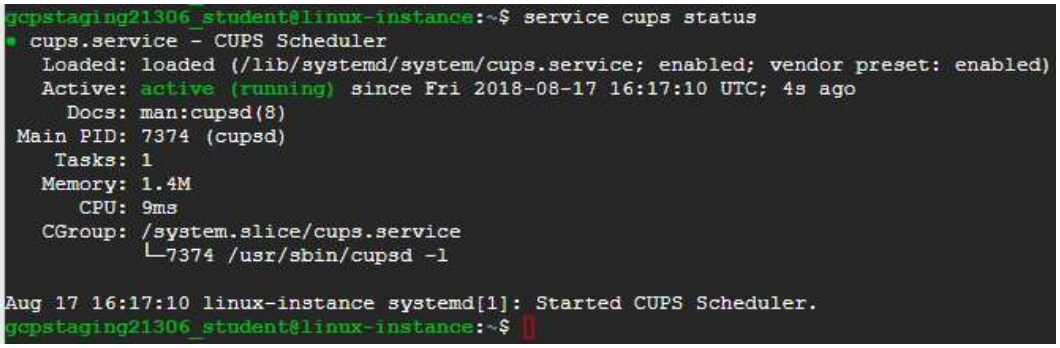
Now that the file was renamed successfully, we can start cups:

```
sudo service cups start
```

And then check the status:

```
sudo service cups status
```

**Output**:




We've fixed it!

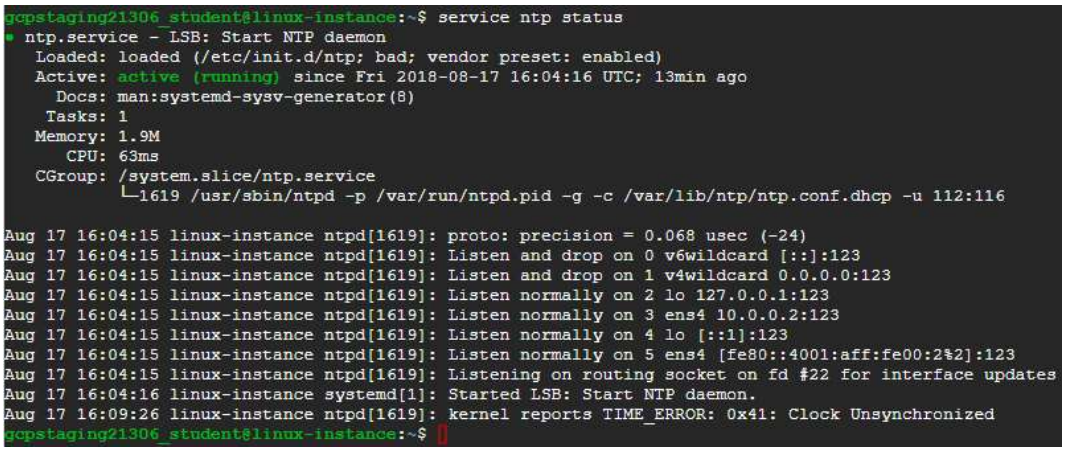
Click *Check my progress* to verify the objective. Fix Cups Service

# Restarting services

Let's now have a look at the service that is keeping the date and time of the machine accurate: NTP.

```
sudo service ntp status
```

**Output**:



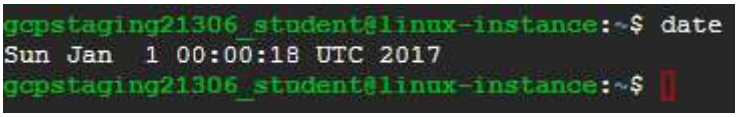

It's running and keeping the time on our machine synchronized with time servers around the world. Now, what if we manually change the date? ntp realizes that this is not normal clock drift. It will detect the change but not interfere. Let's change the date using the date command:

```
sudo date -s '2017-01-01 00:00:00'
```

If we wait a few seconds, we can check that the date is still the same using the date command without parameters.

```
date
```

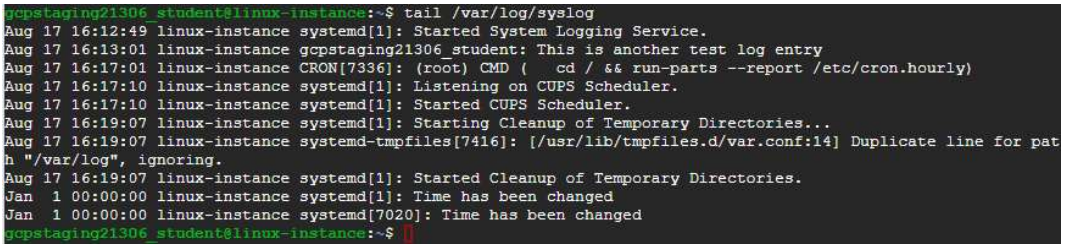
**Output**:



Let's look at the last lines in syslog:

```
sudo tail /var/log/syslog
```

**Output**:



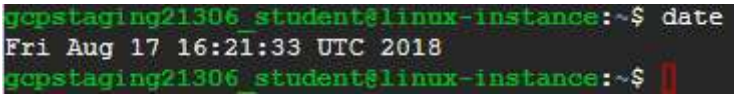

So, we see that the machine detected the time change, but our ntp service did not change it. This is because the ntp service avoids doing any drastic changes while it's running. It will, however, perform drastic changes when it starts. If we now restart ntp, the service will notice the change in time and fix it to the current time:

```
sudo service ntp restart
```

Once we have restarted it, we can check the date again.

```
date
```

**Output**:




Restaring the service command is a handy way of stopping a service and then starting it immediately back up.

**Note**: Sometimes the ntp service might take time to update the current time. If you are not getting the current time then execute the date command after few seconds.
Click *Check my progress* to verify the objective. Change the Date and Restart NTP

# Reloading Services

Finally, let's look at the reload action. Take this action when you want a service to re-read its configuration without actually doing a full stop and start.

Let's go back to the cups service that we just fixed. The logs generated by cups are written into the `/var/log/cups` directory. We can see the contents of the directory using the ls command:

```
sudo ls -l /var/log/cups
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ ls -l /var/log/cups
total 4
-rw-r----- 1 root adm 848 Aug 17 16:17 access_log
-rw-r----- 1 root adm   0 Aug 17 16:04 error_log
-rw-r----- 1 root adm   0 Aug 17 16:04 page_log
gcpstaging21306_student@linux-instance:~$
```

The error log is empty. That's expected because by default cups will only write warning or error messages into that file. If you want cups to log debug messages into that file, you'll need to change the LogLevel parameter in the configuration file. Let's do that.

Let's edit `/etc/cups/cupsd.conf` using the nano editor.

```
sudo nano /etc/cups/cupsd.conf
```

In one of the first lines of the file you'll see there's a line that says `LogLevel warn`. We want to replace **warn** with **debug**:

```
#
# Configuration file for the CUPS scheduler.  See "man cupsd.conf" for a
# complete description of this file.
#

# Log general information in error_log - change "warn" to "debug"
# for troubleshooting...
LogLevel debug
PageLogFormat

# Deactivate CUPS' internal logrotating, as we provide a better one, especially
# LogLevel debug2 gets usable now
MaxLogSize 0
```

Once you've done this, press **"Ctrl-X"** to exit the editor. It will ask you if you want to save your changes, press **"Y"** for yes and then **enter** at the filename prompt.

Once we've done this, we can reload cups:

```
sudo service cups reload
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ sudo service cups reload
 * Reloading Common Unix Printing System cupsd
gcpstaging21306_student@linux-instance:~$
```

And once we've done that, we can see that there's now a lot of content in `/var/log/cups/error_log`.

```
sudo cat /var/log/cups/error_log
```

**Output**:

```
gcpstaging21306_student@linux-instance:~$ cat /var/log/cups/error_log
I [17/Aug/2018:16:24:44 +0000] Listening to [v1.::1]:631 (IPv6)
I [17/Aug/2018:16:24:44 +0000] Listening to 127.0.0.1:631 (IPv4)
I [17/Aug/2018:16:24:44 +0000] Remote access is disabled.
D [17/Aug/2018:16:24:44 +0000] Added auto ServerAlias linux-instance
I [17/Aug/2018:16:24:44 +0000] Loaded configuration file "/etc/cups/cupsd.conf"
D [17/Aug/2018:16:24:44 +0000] Using keychain "/etc/cups/ssl" for server name "linux-instance".
I [17/Aug/2018:16:24:44 +0000] Configured for up to 100 clients.
I [17/Aug/2018:16:24:44 +0000] Allowing up to 100 client connections per host.
I [17/Aug/2018:16:24:44 +0000] Using policy "default" as the default.
I [17/Aug/2018:16:24:44 +0000] Full reload is required.
I [17/Aug/2018:16:24:44 +0000] Saving job.cache...
I [17/Aug/2018:16:24:44 +0000] Loaded MIME database from "/usr/share/cups/mime" and "/etc/cups": 49 types,
ilters...
I [17/Aug/2018:16:24:44 +0000] Loading job cache file "/var/cache/cups/job.cache"...
D [17/Aug/2018:16:24:44 +0000] cupsdAddSubscription(mask=0, dest=(nil)(), job=(nil)(0), uri="(null)")
D [17/Aug/2018:16:24:44 +0000] cupsdAddSubscription(mask=0, dest=(nil)(), job=(nil)(0), uri="(null)")
I [17/Aug/2018:16:24:44 +0000] Full reload complete.
I [17/Aug/2018:16:24:44 +0000] Listening to /var/run/cups/cups.sock on fd 3...
I [17/Aug/2018:16:24:44 +0000] Listening to [v1.::1]:631 on fd 10...
I [17/Aug/2018:16:24:44 +0000] Listening to 127.0.0.1:631 on fd 11...
I [17/Aug/2018:16:24:44 +0000] Resuming new connection processing...
D [17/Aug/2018:16:24:44 +0000] cupsdSetBusyState: newbusy="Not busy", busy="Not busy"
D [17/Aug/2018:16:24:44 +0000] Notifier dbus started - PID = 7555
D [17/Aug/2018:16:24:44 +0000] cupsdMarkDirty(----S)
D [17/Aug/2018:16:24:44 +0000] cupsdSetBusyState: newbusy="Dirty files", busy="Not busy"
D [17/Aug/2018:16:24:44 +0000] [Notifier] state=3
I [17/Aug/2018:16:24:44 +0000] Expiring subscriptions...
D [17/Aug/2018:16:24:44 +0000] Report: clients=0
D [17/Aug/2018:16:24:44 +0000] Report: jobs=0
D [17/Aug/2018:16:24:44 +0000] Report: jobs-active=0
D [17/Aug/2018:16:24:44 +0000] Report: printers=0
D [17/Aug/2018:16:24:44 +0000] Report: stringpool-string-count=365
D [17/Aug/2018:16:24:44 +0000] Report: stringpool-alloc-bytes=3688
D [17/Aug/2018:16:24:44 +0000] Report: stringpool-total-bytes=4752
D [17/Aug/2018:16:24:44 +0000] [Notifier] Connected to D-BUS
D [17/Aug/2018:16:24:44 +0000] [Notifier] ServerRestarted
I [17/Aug/2018:16:24:45 +0000] Expiring subscriptions...
I [17/Aug/2018:16:25:15 +0000] Saving subscriptions.conf...
D [17/Aug/2018:16:25:15 +0000] cupsdSetBusyState: newbusy="Not busy", busy="Dirty files"
I [17/Aug/2018:16:25:15 +0000] Expiring subscriptions...
gcpstaging21306_student@linux-instance:~$
```

If you check the status of the service, you'll see that it was not restarted (it's been running since we fixed it).

`sudo service cups status`

**Output**:

```
gcpstaging21306_student@linux-instance:~$ sudo service cups status
● cups.service - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2018-08-17 16:17:10 UTC; 9min ago
     Docs: man:cupsd(8)
 Main PID: 7374 (cupsd)
    Tasks: 2
   Memory: 2.5M
      CPU: 18ms
   CGroup: /system.slice/cups.service
           ├─7374 /usr/sbin/cupsd -l
           └─7555 /usr/lib/cups/notifier/dbus dbus://

Aug 17 16:17:10 linux-instance systemd[1]: Started CUPS Scheduler.
gcpstaging21306_student@linux-instance:~$
```

By using the reload action, we caused the service to re-read its configuration without being stopped at any point.

Click *Check my progress* to verify the objective. Change the Log Level for Cups

# Conclusion

Congrats! You've successfully listed all the services that are running on the machine, practiced stopping and starting some of these services, and queried their status. You also fixed a problem in the service that was failing to start and edited the configuration of another service.

These are important commands and problem-solving skills that you'll use on a daily basis as a system administrator. Keep it up!

# End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.