Linux Shell Scripting - Check Local EC2 Services

In this tutorial I will be demonstrating how to check services running locally on an EC2 instance.

Prerequisites

- an AWS Free Tier account
- AWS Ubuntu 20 EC2 instance with SSH, Nginx & Postfix services installed
- AWS RHEL 8 EC2 instance with SSHD, Nginx & Postfix services installed
- internet access

If you do not have an AWS account, you can access my **AWS Create Free Tier Account** tutorial here.

After creating an AWS account, I have a number of tutorials that can be completed in order to be ready to begin this tutorial.

Ubuntu 20 EC2	RHEL 8 EC2
Create AWS Ubuntu 20 EC2 Instance	Create AWS RHEL 8 EC2 Instance
AWS Ubuntu 20 EC2 Nginx Install	AWS RHEL 8 EC2 Nginx Install
AWS Ubuntu 20 EC2 Postfix Install	AWS RHEL 8 EC2 Postfix Install

During script creation, I will first test the script at the command line to ensure it works correctly. After verifying that it works, I will automate the process using a cron job.

Steps to complete tutorial:

Create Script

First, in my home directory, I created a new **scripts** directory and, in it, I placed a newly created empty script file named **check_service.sh**

```
# on Ubuntu 20 EC2
/home/ubuntu/scripts/check_service.sh
```

```
# on RHEL 8 EC2
/home/ec2-user/scripts/check_service.sh
```

I will begin by adding the full path of the command interpreter (in this case **/bin/bash**), as well as, a brief description of what the script does:

```
#!/bin/bash
#
# This script is used to check if a service running locally on an EC2 instance
```

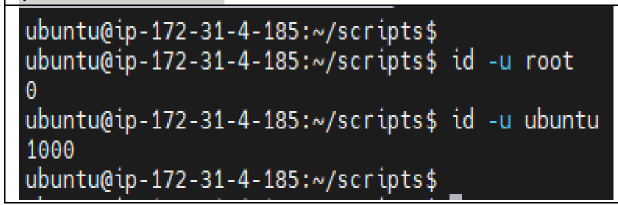
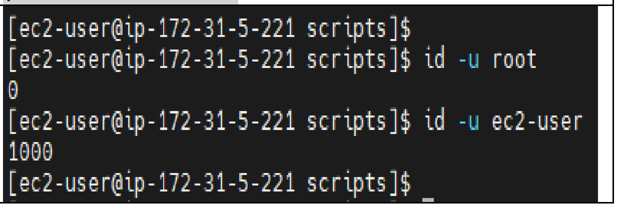
Ensure Superuser Privileges

Then, I want to ensure that the script is being executed by a user with superuser privileges (either **root** or using **sudo**). I will check the environment variable **UID** which is a unique identifier assigned to each user. If the **$UID == 0**, then either the **root** user, or a user with **sudo** privileges, has executed the script.

I will verify the **UID** values assigned to the root user & non-root users on my EC2 instances:

Ubuntu 20 EC2	**RHEL 8 EC2**

`$ id -u root`
`$ id -u ubuntu`
	`$ id -u root`
	`$ id -u ec2-user`

 `ubuntu@ip-172-31-4-185:~/scripts$`
`ubuntu@ip-172-31-4-185:~/scripts$ id -u root`
`0`
`ubuntu@ip-172-31-4-185:~/scripts$ id -u ubuntu`
`1000`
`ubuntu@ip-172-31-4-185:~/scripts$ _`
	 `[ec2-user@ip-172-31-5-221 scripts]$`
	`[ec2-user@ip-172-31-5-221 scripts]$ id -u root`
	`0`
	`[ec2-user@ip-172-31-5-221 scripts]$ id -u ec2-user`
	`1000`
	`[ec2-user@ip-172-31-5-221 scripts]$ _`

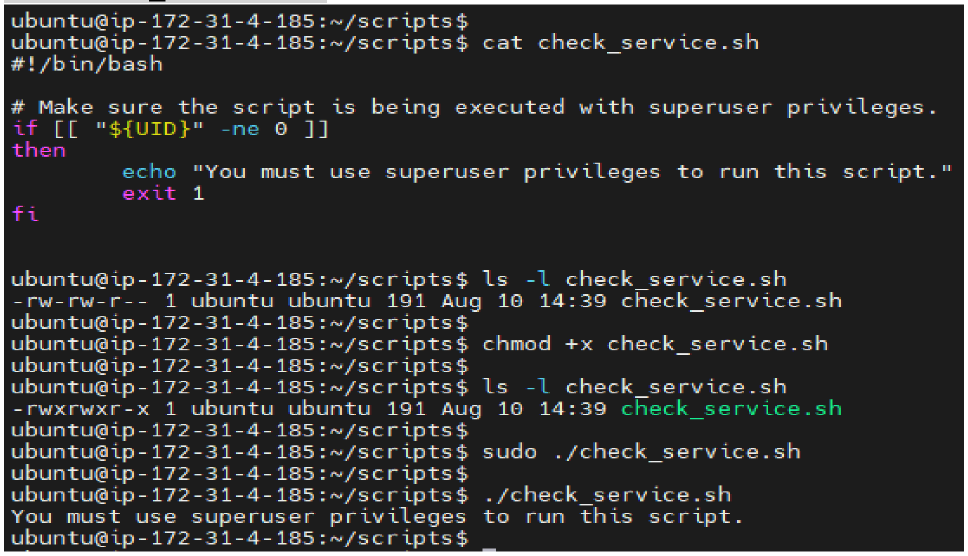
We can see from the above command outputs that the root user has a UID of 0 and the non-root users each have a UID of 1000.

Using this code snippet, I am checking if the **root** user, or a non-root user using **sudo** privileges, is executing the script. If not, provide a helpful usage message and exit the script.

```
# Make sure the script is being executed with superuser privileges.
if [[ "${UID}" -ne 0 ]]
then
        echo "You must use superuser privileges to run this script."
        exit 1
fi
```

I will now test this out, for success & failure, by setting the execute permission on the script and executing it:

```
$ chmod +x check_service.sh
$ sudo ./check_service.sh
$ ./check_service.sh
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ cat check_service.sh
#!/bin/bash

# Make sure the script is being executed with superuser privileges.
if [[ "${UID}" -ne 0 ]]
then
        echo "You must use superuser privileges to run this script."
        exit 1
fi


ubuntu@ip-172-31-4-185:~/scripts$ ls -l check_service.sh
-rw-rw-r-- 1 ubuntu ubuntu 191 Aug 10 14:39 check_service.sh
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ chmod +x check_service.sh
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ ls -l check_service.sh
-rwxrwxr-x 1 ubuntu ubuntu 191 Aug 10 14:39 check_service.sh
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ ./check_service.sh
You must use superuser privileges to run this script.
ubuntu@ip-172-31-4-185:~/scripts$ _
```

You will notice that executing the script without superuser privileges generated the helpful usage message and exited the script.

Service as Script Argument

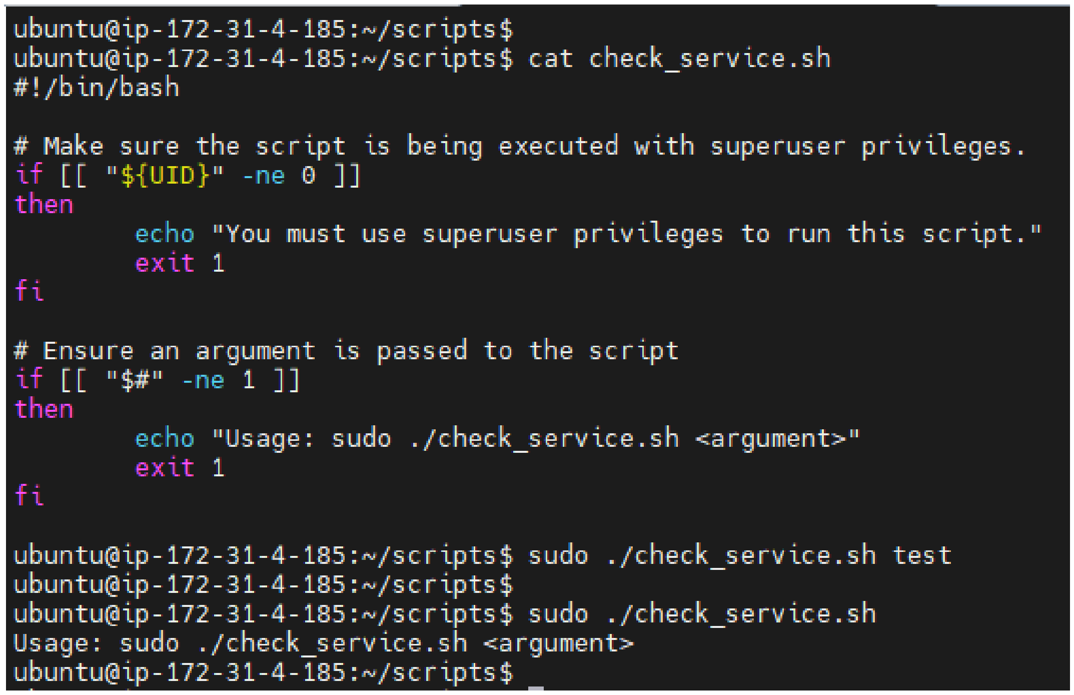
Next, I want to ensure that an argument is passed to the script.

Using the following code snippet, I check that a single argument is passed to the script. If not, provide a helpful usage message and exit the script.

```
# Ensure an argument is passed to the script
if [[ "$#" -ne 1 ]]
then
        echo "Usage: sudo ./check_service.sh <argument>"
        exit 1
fi
```

I will now test this out, for success & failure, by executing the script again:

```
$ sudo ./check_service.sh test
$ sudo ./check_service.sh
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ cat check_service.sh
#!/bin/bash

# Make sure the script is being executed with superuser privileges.
if [[ "${UID}" -ne 0 ]]
then
        echo "You must use superuser privileges to run this script."
        exit 1
fi

# Ensure an argument is passed to the script
if [[ "$#" -ne 1 ]]
then
        echo "Usage: sudo ./check_service.sh <argument>"
        exit 1
fi

ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh test
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh
Usage: sudo ./check_service.sh <argument>
ubuntu@ip-172-31-4-185:~/scripts$ _
```

Again, you will notice that executing the script without an argument generated the helpful usage message and exited the script.

Before proceeding, I want to verify that three services (**ssh, nginx, postfix**) are available locally on both my Ubuntu 20 & RHEL 8 EC2 instances.

Below, I am using the **`--all`** switch to check for both running, and non-running, services. This will ensure that even services in the stopped state are included in the output.

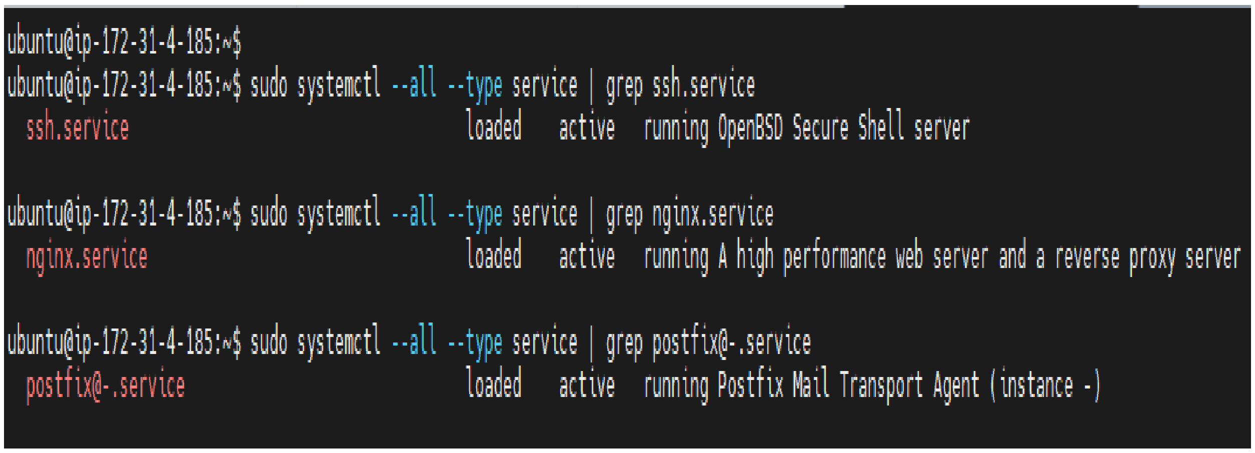
Please note that on Ubuntu 20, there are 2 systemd Postfix services.

We are concerned with the Postfix instance: ***`postfix@-.service`***

`$ sudo systemctl --all --type service | grep ssh.service`

`$ sudo systemctl --all --type service | grep nginx.service`

`$ sudo systemctl --all --type service | grep postfix@-.service`

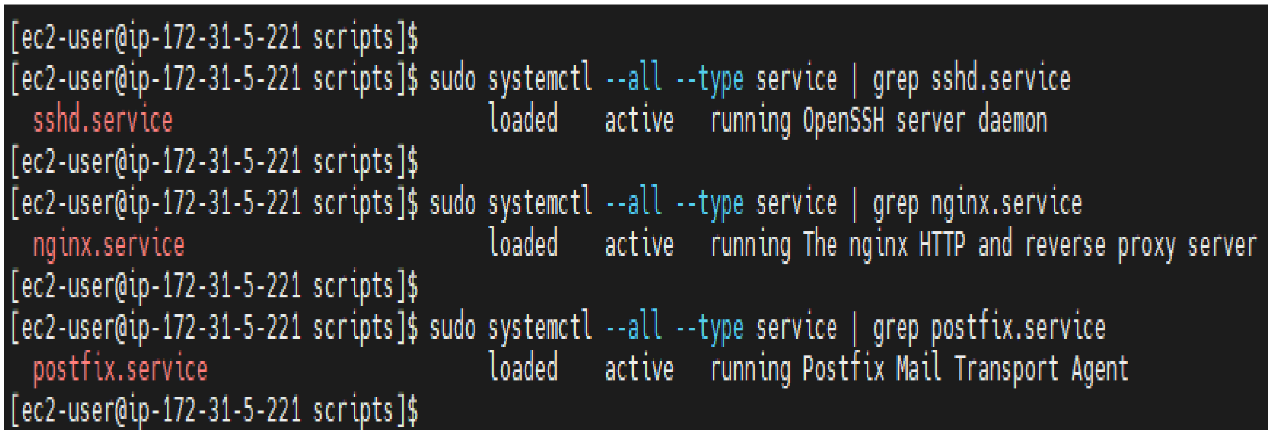


```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo systemctl --all --type service | grep ssh.service
  ssh.service                            loaded    active   running OpenBSD Secure Shell server

ubuntu@ip-172-31-4-185:~$ sudo systemctl --all --type service | grep nginx.service
  nginx.service                          loaded    active   running A high performance web server and a reverse proxy server

ubuntu@ip-172-31-4-185:~$ sudo systemctl --all --type service | grep postfix@-.service
  postfix@-.service                      loaded    active   running Postfix Mail Transport Agent (instance -)
```

`$ sudo systemctl --all --type service | grep sshd.service`

`$ sudo systemctl --all --type service | grep nginx.service`

`$ sudo systemctl --all --type service | grep postfix.service`



```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl --all --type service | grep sshd.service
  sshd.service                           loaded    active   running OpenSSH server daemon
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl --all --type service | grep nginx.service
  nginx.service                          loaded    active   running The nginx HTTP and reverse proxy server
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl --all --type service | grep postfix.service
  postfix.service                        loaded    active   running Postfix Mail Transport Agent
[ec2-user@ip-172-31-5-221 scripts]$
```

The three services (**ssh/d, nginx & postfix**) are available on both of my EC2 instances.

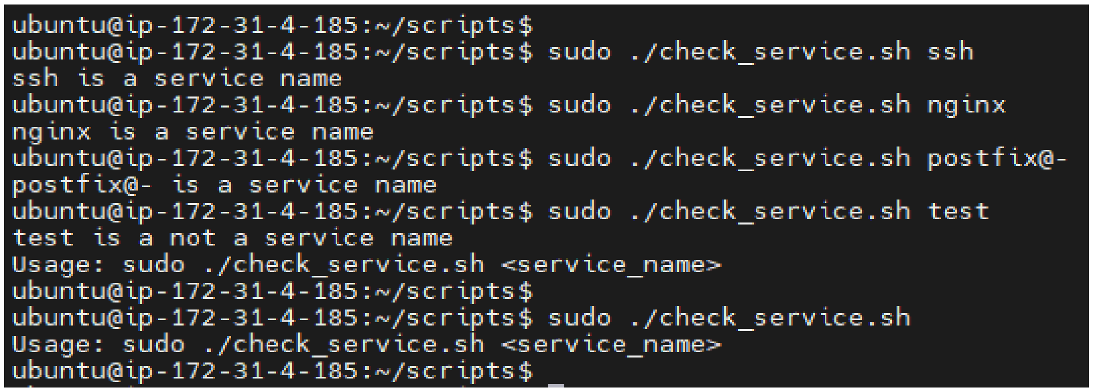
Next, I want to ensure that a single service name is provided as the argument to the script.

Changing the previous code snippet, I now check that the single argument passed to the script is a valid service name. If not, provide a helpful usage message and exit the script.

```
# Ensure a valid service name argument is passed to the script
if [[ "$#" -eq 1 ]]
then
      RESULT=$(systemctl --all --type service | grep ${1} | wc -l)

      if [[ ${RESULT} -gt 0 ]]
      then
            echo "${1} is a service name"
      else
            echo "${1} is a not a service name"
            echo "Usage: sudo ./check_service.sh <service_name>"
            exit 1
      fi
else
      echo "Usage: sudo ./check_service.sh <service_name>"
      exit 1
fi
```

I will now test this out, for success & failure, on my Ubuntu 20 EC2, by executing the following:

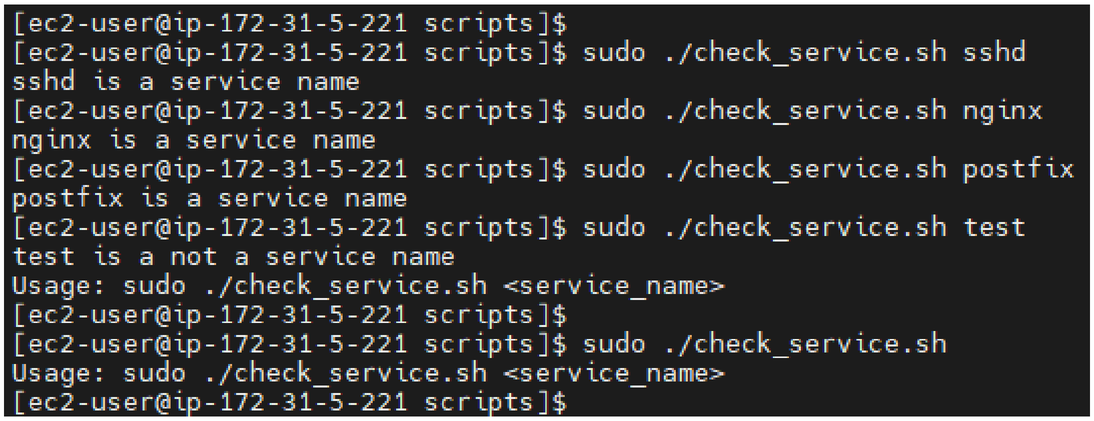
```
$ sudo ./check_service.sh ssh
$ sudo ./check_service.sh nginx
$ sudo ./check_service.sh postfix@-
$ sudo ./check_service.sh test
$ sudo ./check_service.sh
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh ssh
ssh is a service name
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh nginx
nginx is a service name
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh postfix@-
postfix@- is a service name
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh test
test is a not a service name
Usage: sudo ./check_service.sh <service_name>
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh
Usage: sudo ./check_service.sh <service_name>
ubuntu@ip-172-31-4-185:~/scripts$
```

Now, I will test this out, for success & failure, on my RHEL 8 EC2, by executing the following:

```
$ sudo ./check_service.sh sshd
$ sudo ./check_service.sh nginx
$ sudo ./check_service.sh postfix
$ sudo ./check_service.sh test
$ sudo ./check_service.sh
```



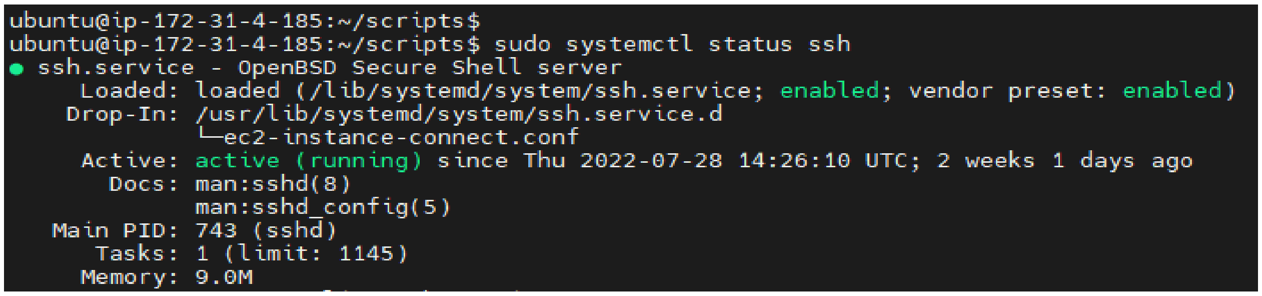
```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh sshd
sshd is a service name
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh nginx
nginx is a service name
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh postfix
postfix is a service name
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh test
test is a not a service name
Usage: sudo ./check_service.sh <service_name>
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh
Usage: sudo ./check_service.sh <service_name>
[ec2-user@ip-172-31-5-221 scripts]$
```

We've confirmed that a service name must be provided as an argument to the script.

Next, we need to determine the state of each service (whether or not it's running).

Service Status

We could just execute **`$ sudo systemctl status <service_name>`**, but then we would have to parse the output to extract what we need and assign it to a variable.



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
    Drop-In: /usr/lib/systemd/system/ssh.service.d
             └─ec2-instance-connect.conf
     Active: active (running) since Thu 2022-07-28 14:26:10 UTC; 2 weeks 1 days ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 743 (sshd)
      Tasks: 1 (limit: 1145)
     Memory: 9.0M
```

Instead, we will check the value of a specific service unit setting, or property.

The basic object of the **systemd** init system is known as a unit. The service unit type is what we are interested in. To see a service unit's low-level settings on the system, we can use the following:

`$ sudo systemctl show <service_name>`

If we want to limit the output of the command, we can use the **`-p`** option for a specific unit property.

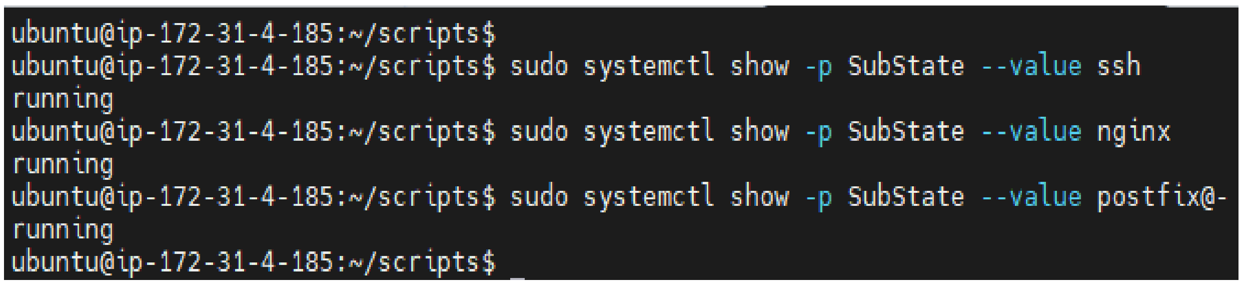
`$ sudo systemctl show -p <property> --value <service_name>`

In our case, we need to determine a service unit's **SubState** property value.

`$ sudo systemctl show -p SubState --value ssh`

`$ sudo systemctl show -p SubState --value nginx`

`$ sudo systemctl show -p SubState --value postfix@-`

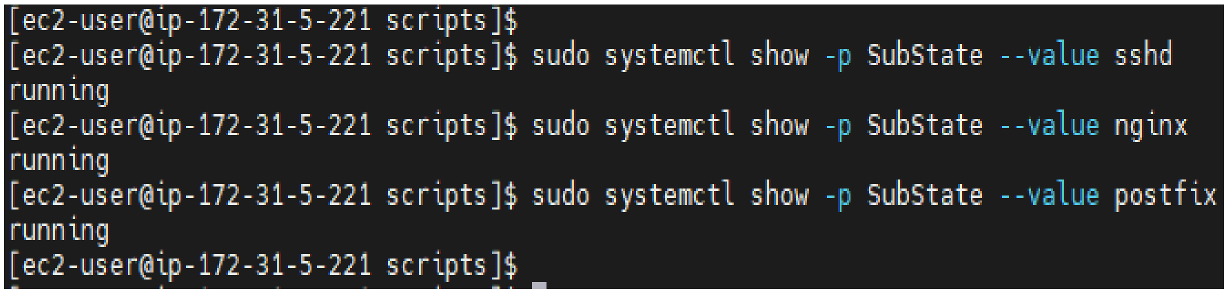


```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo systemctl show -p SubState --value ssh
running
ubuntu@ip-172-31-4-185:~/scripts$ sudo systemctl show -p SubState --value nginx
running
ubuntu@ip-172-31-4-185:~/scripts$ sudo systemctl show -p SubState --value postfix@-
running
ubuntu@ip-172-31-4-185:~/scripts$ _
```

```
$ sudo systemctl show -p SubState --value sshd
```

```
$ sudo systemctl show -p SubState --value nginx
```

```
$ sudo systemctl show -p SubState --value postfix
```



```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl show -p SubState --value sshd
running
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl show -p SubState --value nginx
running
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl show -p SubState --value postfix
running
[ec2-user@ip-172-31-5-221 scripts]$ _
```

I can now assign the output of the above command to a variable that will be used for a conditional test. Before that, I will assign the passed script argument to a variable to clean up the script. After that, I will generate a helpful message that I will use during testing.

```
SERVICE="${1}"
STATUS="$(systemctl show -p SubState --value ${SERVICE})"
MESSAGE="$(echo "${SERVICE} is ${STATUS}")"
```

Then, create variables for my log file. Note, you need to create a **logs** directory under the **scripts** directory. Also be sure to use the correct variable value based on which EC2 instance you are on.

```
# on my Ubuntu 20 EC2
LOGDIR="/home/ubuntu/scripts/logs"
# on my RHEL 8 EC2
LOGDIR="/home/ec2-user/scripts/logs"

LOGFILE="${LOGDIR}/${SERVICE}.log"
TIMESTAMP="$(date +"%Y-%m-%d-%T")"
```

Before running the script again, I add the following to the script to ensure that the log file exists:

```
touch "${LOGFILE}"
```

I am now ready to test whether the specified service is running, or not. If it isn't, I will restart it. Along the way, each step is logged.

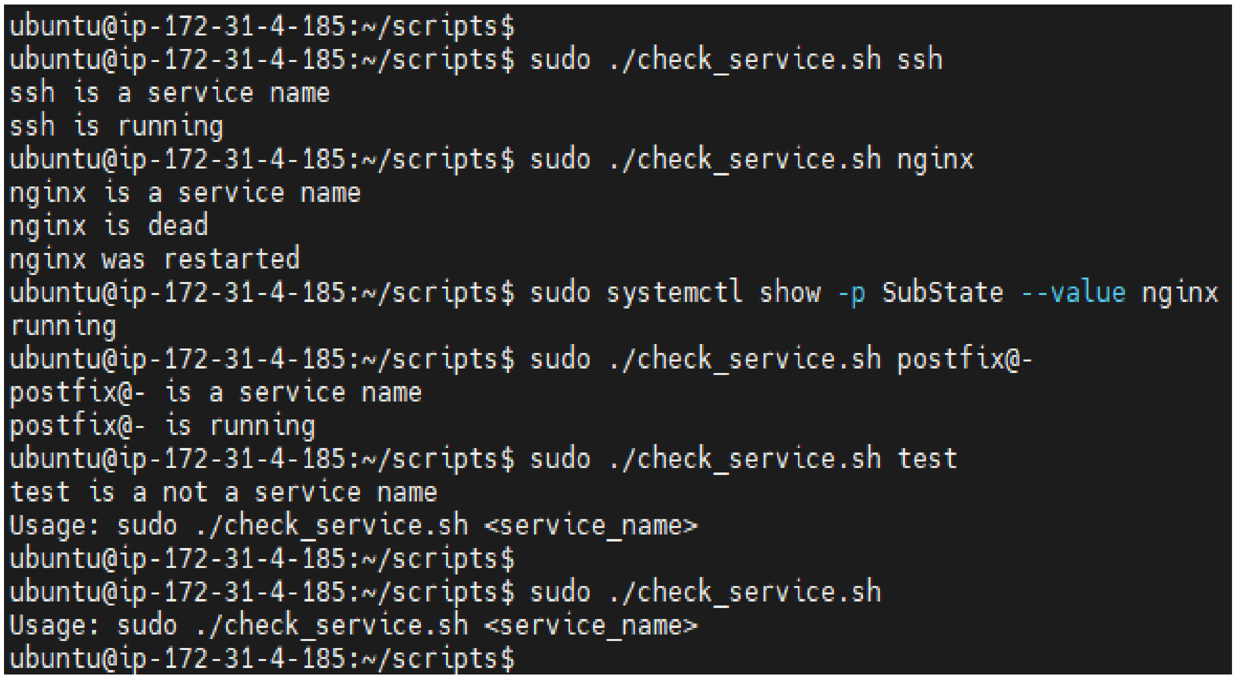
```
if [[ "${STATUS}" != "running" ]]
then
        echo "${MESSAGE}"
        # redirect both stdout & stderr to the log file
        echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
        echo "${TIMESTAMP} -- Restarting ${SERVICE}" &>> ${LOGFILE}

        # on Ubuntu 20 EC2 must ensure postfix service is restarted,
        # not only instance postfix@-
         if [[ "${SERVICE}" == "postfix@-"  ]]
        then
                # delete the shortest substring starting from the end
                systemctl restart "${SERVICE%@.}" &>> ${LOGFILE}
         else
                systemctl restart ${SERVICE} &>> ${LOGFILE}
        fi

        # if the previously executed command was successful, $? == 0
        if [[ "${?}" -ne 0 ]]
        then
                MESSAGE="$(echo "${SERVICE} could NOT be restarted.")"
                echo "${MESSAGE}"
                echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
                exit 1
        else
                MESSAGE="$(echo "${SERVICE} was restarted")"
                echo "${MESSAGE}"
                echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
        fi
else
        echo "${MESSAGE}"
        echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
fi

exit 0
```

Before testing this out, I will manually stop the **nginx** service on my Ubuntu 20 EC2.

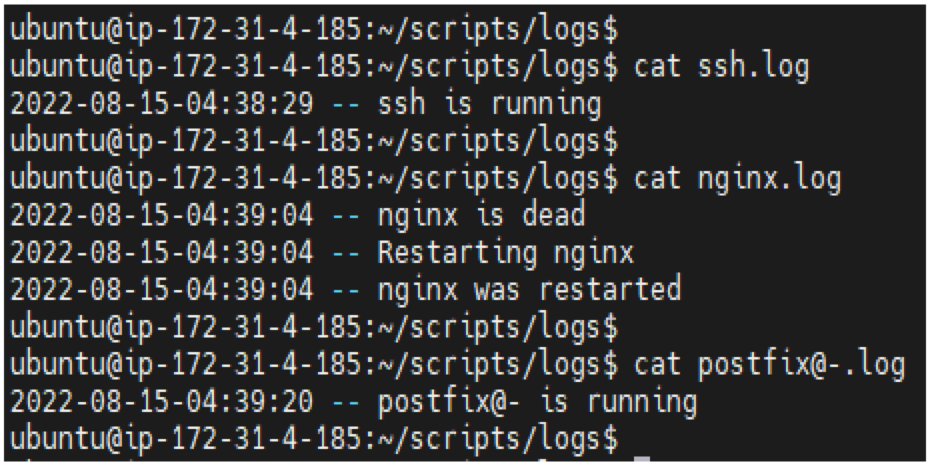
```
$ sudo systemctl stop nginx
```

I will now test, for success & failure, on my Ubuntu 20 EC2, by executing the following:

```
$ sudo ./check_service.sh ssh
$ sudo ./check_service.sh nginx
$ sudo systemctl show -p SubState --value nginx
$ sudo ./check_service.sh postfix@-
$ sudo ./check_service.sh test
$ sudo ./check_service.sh
```


```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh ssh
ssh is a service name
ssh is running
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh nginx
nginx is a service name
nginx is dead
nginx was restarted
ubuntu@ip-172-31-4-185:~/scripts$ sudo systemctl show -p SubState --value nginx
running
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh postfix@-
postfix@- is a service name
postfix@- is running
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh test
test is a not a service name
Usage: sudo ./check_service.sh <service_name>
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_service.sh
Usage: sudo ./check_service.sh <service_name>
ubuntu@ip-172-31-4-185:~/scripts$
```

I can now check the log files generated from my testing on my Ubuntu 20 EC2:

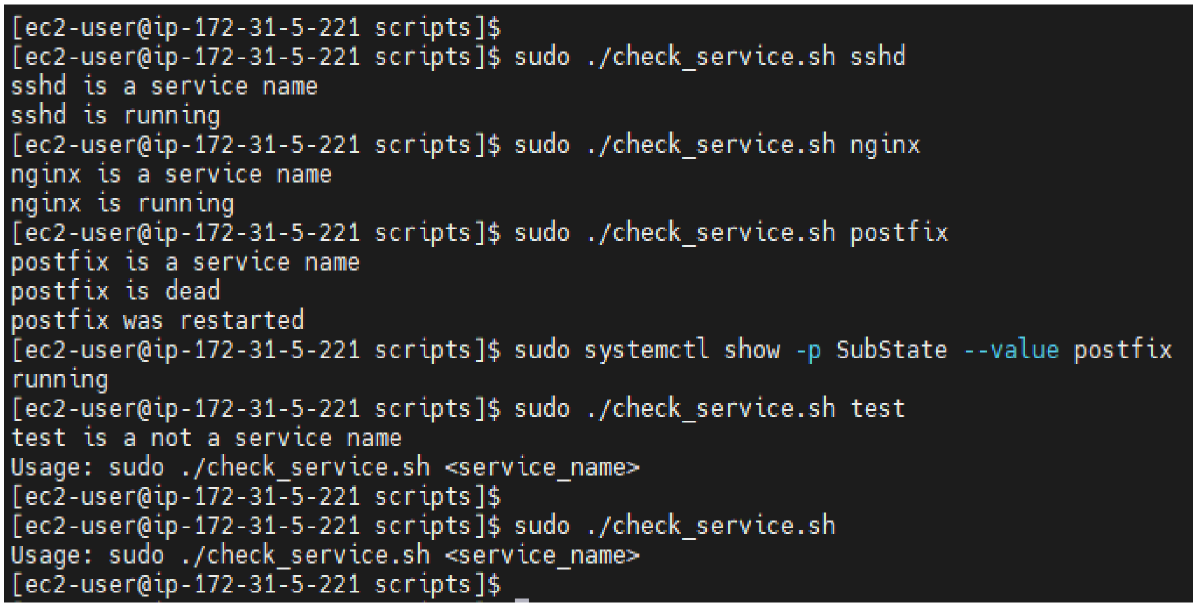

```
ubuntu@ip-172-31-4-185:~/scripts/logs$
ubuntu@ip-172-31-4-185:~/scripts/logs$ cat ssh.log
2022-08-15-04:38:29 -- ssh is running
ubuntu@ip-172-31-4-185:~/scripts/logs$
ubuntu@ip-172-31-4-185:~/scripts/logs$ cat nginx.log
2022-08-15-04:39:04 -- nginx is dead
2022-08-15-04:39:04 -- Restarting nginx
2022-08-15-04:39:04 -- nginx was restarted
ubuntu@ip-172-31-4-185:~/scripts/logs$
ubuntu@ip-172-31-4-185:~/scripts/logs$ cat postfix@-.log
2022-08-15-04:39:20 -- postfix@- is running
ubuntu@ip-172-31-4-185:~/scripts/logs$
```

I will now manually stop the **postfix** service on my RHEL 8 EC2.

```
$ sudo systemctl stop postfix
```

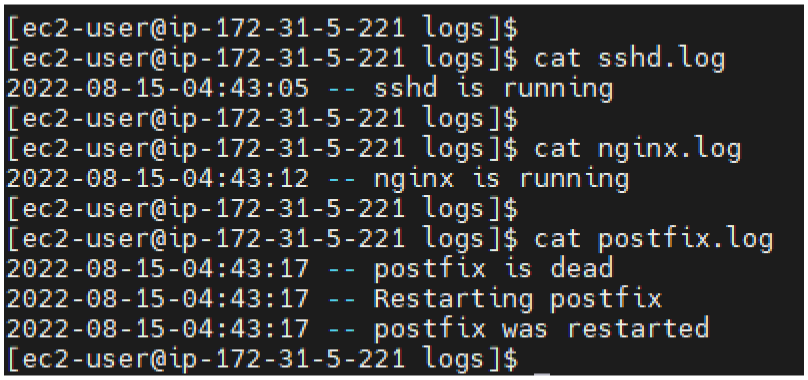
Now, I will test this out, for success & failure, on my RHEL 8 EC2, by executing the following:

```
$ sudo ./check_service.sh sshd
$ sudo ./check_service.sh nginx
$ sudo ./check_service.sh postfix
$ sudo systemctl show -p SubState --value postfix
$ sudo ./check_service.sh test
$ sudo ./check_service.sh
```



```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh sshd
sshd is a service name
sshd is running
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh nginx
nginx is a service name
nginx is running
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh postfix
postfix is a service name
postfix is dead
postfix was restarted
[ec2-user@ip-172-31-5-221 scripts]$ sudo systemctl show -p SubState --value postfix
running
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh test
test is a not a service name
Usage: sudo ./check_service.sh <service_name>
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_service.sh
Usage: sudo ./check_service.sh <service_name>
[ec2-user@ip-172-31-5-221 scripts]$ _
```

I can now check the log files generated from my testing on my RHEL 8 EC2:



```
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat sshd.log
2022-08-15-04:43:05 -- sshd is running
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat nginx.log
2022-08-15-04:43:12 -- nginx is running
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat postfix.log
2022-08-15-04:43:17 -- postfix is dead
2022-08-15-04:43:17 -- Restarting postfix
2022-08-15-04:43:17 -- postfix was restarted
[ec2-user@ip-172-31-5-221 logs]$ _
```

The script worked on both of my EC2 instances.

Review Script

```
#!/bin/bash

# This script is used to check if a service is running locally on an EC2 instance

# Make sure the script is being executed with superuser privileges.
if [[ "${UID}" -ne 0 ]]
then
	echo "You must use superuser privileges to run this script."
	exit 1
fi

# Ensure a valid service name argument is passed to the script
if [[ "$#" -eq 1 ]]
then
	RESULT=$(systemctl --all --type service | grep ${1} | wc -l)
	if [[ ${RESULT} -gt 0 ]]
	then
		echo "${1} is a service name"
	else
		echo "${1} is a not a service name"
		echo "Usage: sudo ./check_service.sh <service_name>"
		exit 1
	fi
else
	echo "Usage: sudo ./check_service.sh <service_name>"
	exit 1
fi

SERVICE="${1}"
STATUS="$(systemctl show -p SubState --value ${SERVICE})"
MESSAGE="$(echo "${SERVICE} is ${STATUS}")"

# LOGDIR depends on EC2 instance type (Ubuntu 20 OR RHEL 8)
#LOGDIR="/home/ubuntu/scripts/logs"
#LOGDIR="/home/ec2-user/scripts/logs"
TIMESTAMP="$(date +"%Y-%m-%d-%T")"
LOGFILE="${LOGDIR}/${SERVICE}.log"

touch "${LOGFILE}"

if [[ "${STATUS}" != "running" ]]
then
	# redirect both stdout & stderr to the log file
	echo "${MESSAGE}"
	echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
	echo "${TIMESTAMP} -- Restarting ${SERVICE}" &>> ${LOGFILE}
	
	# on Ubuntu 20 EC2 must ensure postfix service is restarted,
	# not only instance postfix@-
	if [[ "${SERVICE}" == "postfix@-"  ]]
	then
		# delete the shortest substring starting from the end
		systemctl restart "${SERVICE%@.}" &>> ${LOGFILE}
	else
		systemctl restart ${SERVICE} &>> ${LOGFILE}
	fi
	
	# if the previously executed command was successful, $? == 0
	if [[ "${?}" -ne 0 ]]
	then
		MESSAGE="$(echo "${SERVICE} could NOT be restarted. Check the logs!")"
		echo "${MESSAGE}"
		echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
		exit 1
	else
		MESSAGE="$(echo "${SERVICE} was restarted")"
		echo "${MESSAGE}"
		echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
	fi
else
	echo "${MESSAGE}"
	echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
```

```
fi

exit 0
```

Automate with CROND

Please note, the SSH/SSHD service must be running on an EC2 instance for it to be accessible. Should the SSH/SSHD service be down on an EC2 instance, the AWS console and EC2 dashboard would need to be accessed manually to restart the instance so that the service would become available.

To prevent this, I can create a cron job and use this shell script to check whether the SSH/SSHD service is running, or not. If it isn't, it will automatically be restarted.

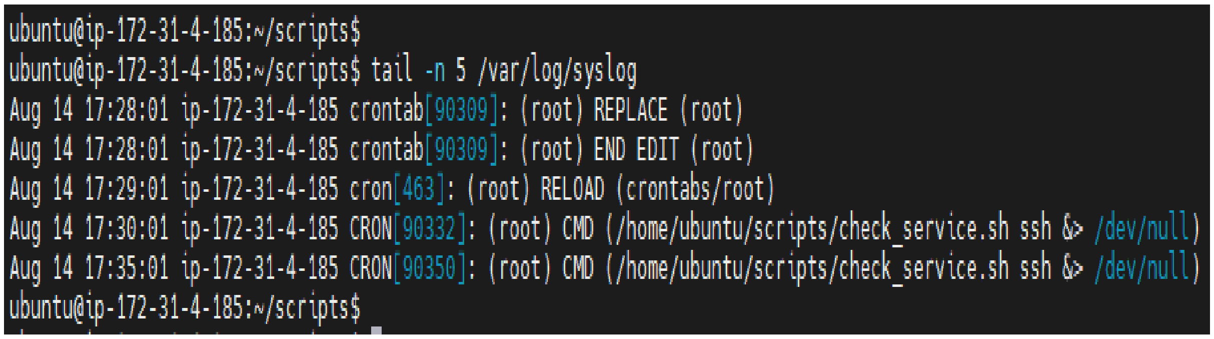
I will create a cron job, on each of my EC2 instances, using: **`check_service.sh`**. I want the cron job to run every 5 minutes and have the output generated discarded by redirecting it to **`/dev/null`**. This will prevent cron job email notifications from being sent out each time the job completes.

```
$ sudo crontab -e
```

```
# Ubuntu 20 EC2
# m h dom mon dow (m==minute, h==hour, dom==day of month, mon==month, dow==day of week)
*/5 * * * * /home/ubuntu/scripts/check_service.sh ssh &> /dev/null

# RHEL 8 EC2
# m h dom mon dow (m==minute, h==hour, dom==day of month, mon==month, dow==day of week)
*/5 * * * * /home/ec2-user/scripts/check_service.sh sshd &> /dev/null
```

I will check the system logs to ensure that the cron job is running on my Ubuntu 20 EC2.

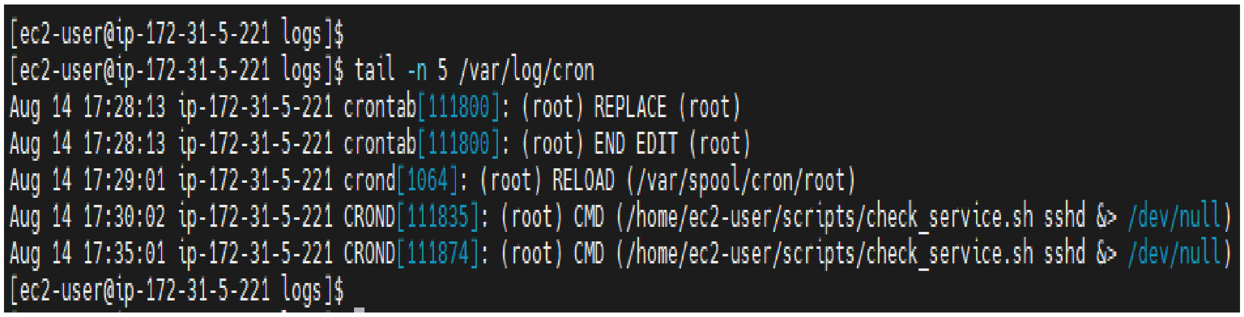
```
$ tail -n 5 /var/log/syslog
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ tail -n 5 /var/log/syslog
Aug 14 17:28:01 ip-172-31-4-185 crontab[90309]: (root) REPLACE (root)
Aug 14 17:28:01 ip-172-31-4-185 crontab[90309]: (root) END EDIT (root)
Aug 14 17:29:01 ip-172-31-4-185 cron[463]: (root) RELOAD (crontabs/root)
Aug 14 17:30:01 ip-172-31-4-185 CRON[90332]: (root) CMD (/home/ubuntu/scripts/check_service.sh ssh &> /dev/null)
Aug 14 17:35:01 ip-172-31-4-185 CRON[90350]: (root) CMD (/home/ubuntu/scripts/check_service.sh ssh &> /dev/null)
ubuntu@ip-172-31-4-185:~/scripts$
```

I will check the cron logs to ensure that the cron job is running on my RHEL 8 EC2.

```
$ tail -n 5 /var/log/cron
```

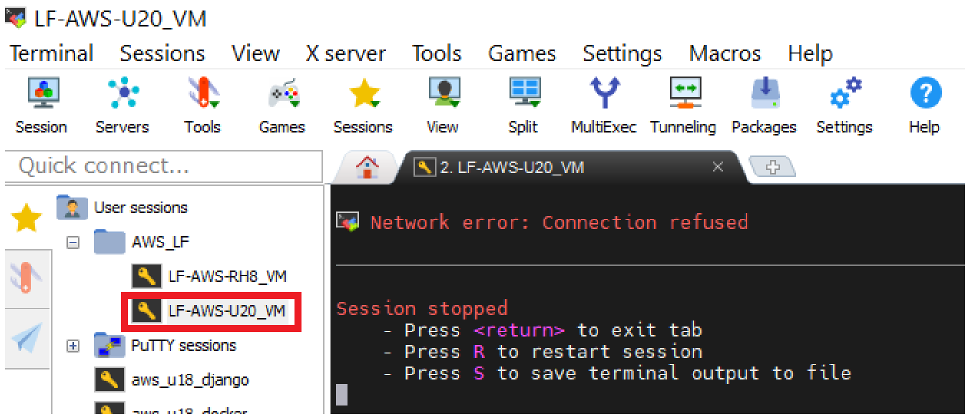


```
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ tail -n 5 /var/log/cron
Aug 14 17:28:13 ip-172-31-5-221 crontab[111800]: (root) REPLACE (root)
Aug 14 17:28:13 ip-172-31-5-221 crontab[111800]: (root) END EDIT (root)
Aug 14 17:29:01 ip-172-31-5-221 crond[1064]: (root) RELOAD (/var/spool/cron/root)
Aug 14 17:30:02 ip-172-31-5-221 CROND[111835]: (root) CMD (/home/ec2-user/scripts/check_service.sh sshd &> /dev/null)
Aug 14 17:35:01 ip-172-31-5-221 CROND[111874]: (root) CMD (/home/ec2-user/scripts/check_service.sh sshd &> /dev/null)
[ec2-user@ip-172-31-5-221 logs]$
```

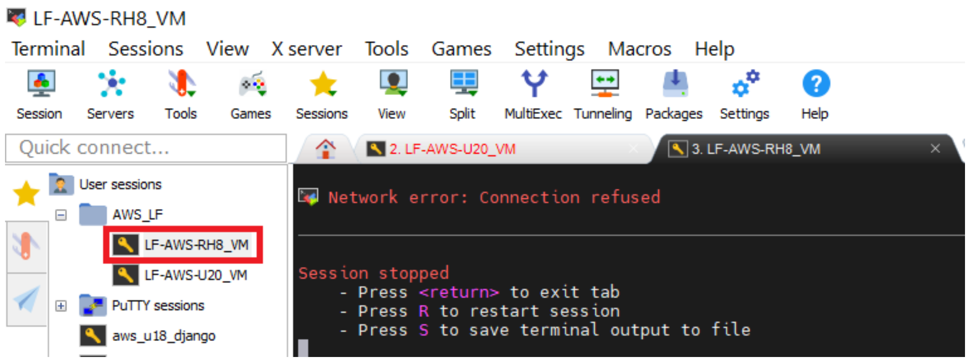
Now that the cron jobs are running, I will manually stop the service on each EC2 instance.

```
# Ubuntu 20 EC2
$ sudo systemctl stop ssh

# RHEL 8 EC2
$ sudo systemctl stop sshd
```

I will now close my open SSH sessions and attempt to reopen them.


LF-AWS-U20_VM
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. LF-AWS-U20_VM
User sessions
AWS_LF
LF-AWS-RH8_VM
LF-AWS-U20_VM
PuTTY sessions
aws_u18_django

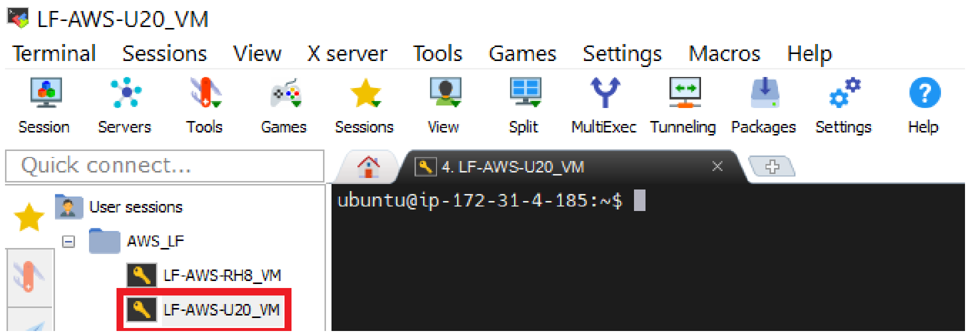
```
Network error: Connection refused

Session stopped
    - Press <return> to exit tab
    - Press R to restart session
    - Press S to save terminal output to file
```


LF-AWS-RH8_VM
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. LF-AWS-U20_VM
3. LF-AWS-RH8_VM
User sessions
AWS_LF
LF-AWS-RH8_VM
LF-AWS-U20_VM
PuTTY sessions
aws_u18_django

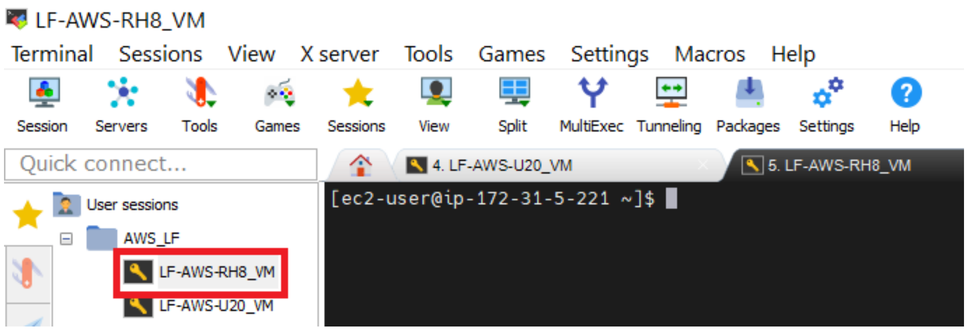
```
Network error: Connection refused

Session stopped
    - Press <return> to exit tab
    - Press R to restart session
    - Press S to save terminal output to file
```

You will notice that I cannot connect to either EC2 instance.

I will wait the 5 minutes and then try connecting again.


LF-AWS-U20_VM
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
4. LF-AWS-U20_VM
User sessions
AWS_LF
LF-AWS-RH8_VM
LF-AWS-U20_VM

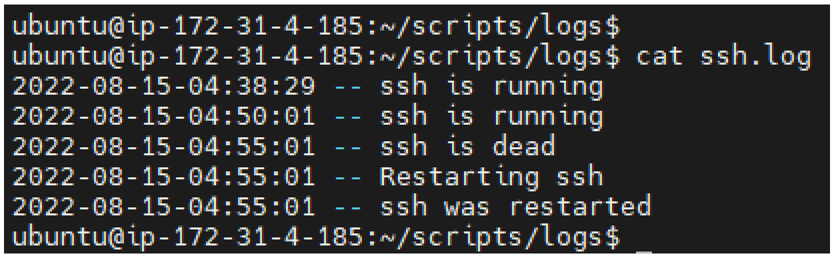
```
ubuntu@ip-172-31-4-185:~$
```


LF-AWS-RH8_VM
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
4. LF-AWS-U20_VM
5. LF-AWS-RH8_VM
User sessions
AWS_LF
LF-AWS-RH8_VM
LF-AWS-U20_VM

```
[ec2-user@ip-172-31-5-221 ~]$
```

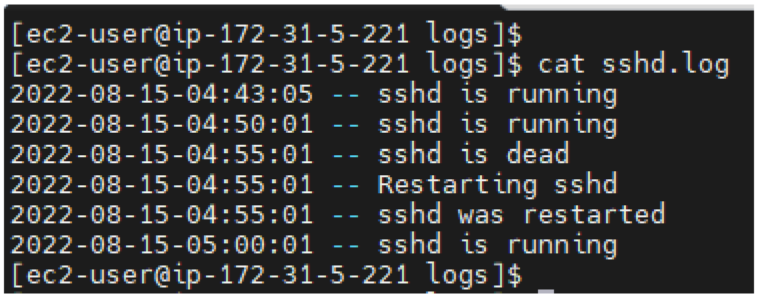
Now that I've connected to both EC2 instances, I can verify the script log files.

```
# Ubuntu 20 EC2
$ tail -n 5 /home/ubuntu/scripts/logs/ssh.log
```



```
ubuntu@ip-172-31-4-185:~/scripts/logs$
ubuntu@ip-172-31-4-185:~/scripts/logs$ cat ssh.log
2022-08-15-04:38:29 -- ssh is running
2022-08-15-04:50:01 -- ssh is running
2022-08-15-04:55:01 -- ssh is dead
2022-08-15-04:55:01 -- Restarting ssh
2022-08-15-04:55:01 -- ssh was restarted
ubuntu@ip-172-31-4-185:~/scripts/logs$
```

```
# RHEL 8 EC2
$ tail -n 5 /home/ec2-user/scripts/logs/sshd.log
```



```
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat sshd.log
2022-08-15-04:43:05 -- sshd is running
2022-08-15-04:50:01 -- sshd is running
2022-08-15-04:55:01 -- sshd is dead
2022-08-15-04:55:01 -- Restarting sshd
2022-08-15-04:55:01 -- sshd was restarted
2022-08-15-05:00:01 -- sshd is running
[ec2-user@ip-172-31-5-221 logs]$
```

The cron job ran successfully restarting the SSH/SSHD on each EC2 instance.

I hope you have enjoyed completing this tutorial and found it helpful.

If you would like to use Amazon SES (Simple Email Service) to send email notifications for successfully completed cron jobs, from an AWS EC2, my tutorial **CROND Email Notifications** is accessible **here**.

Back to Top