Linux Shell Scripting - Check Remote EC2 Services

In this tutorial I will be demonstrating how to check remote services running 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.

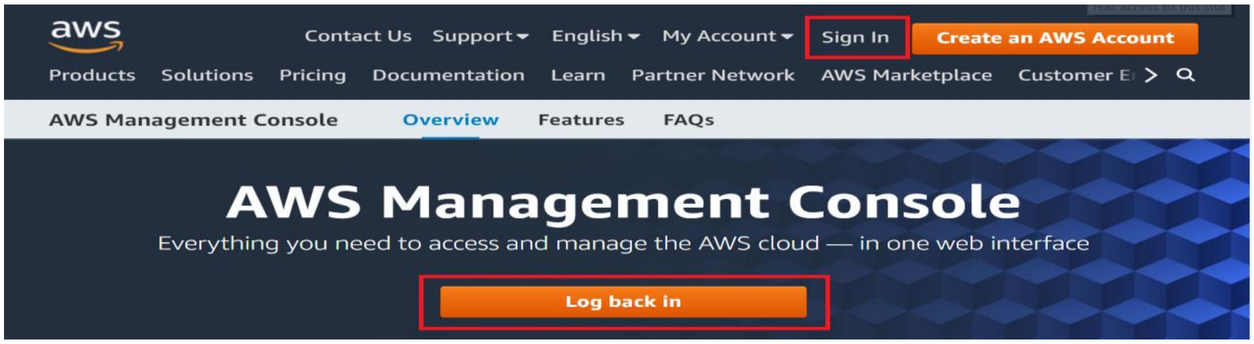
Ubuntu 20 EC2	RHEL 8 EC2
Create AWS Ubuntu 20 EC2 Instance	Create AWS RHEL 8 EC2 Instance
AWS Ubuntu 20 EC2 Nginx Install	AWS RHEL 8 EC2 Nginx Install
AWS Ubuntu 20 EC2 Postfix Install	AWS RHEL 8 EC2 Postfix Install

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.

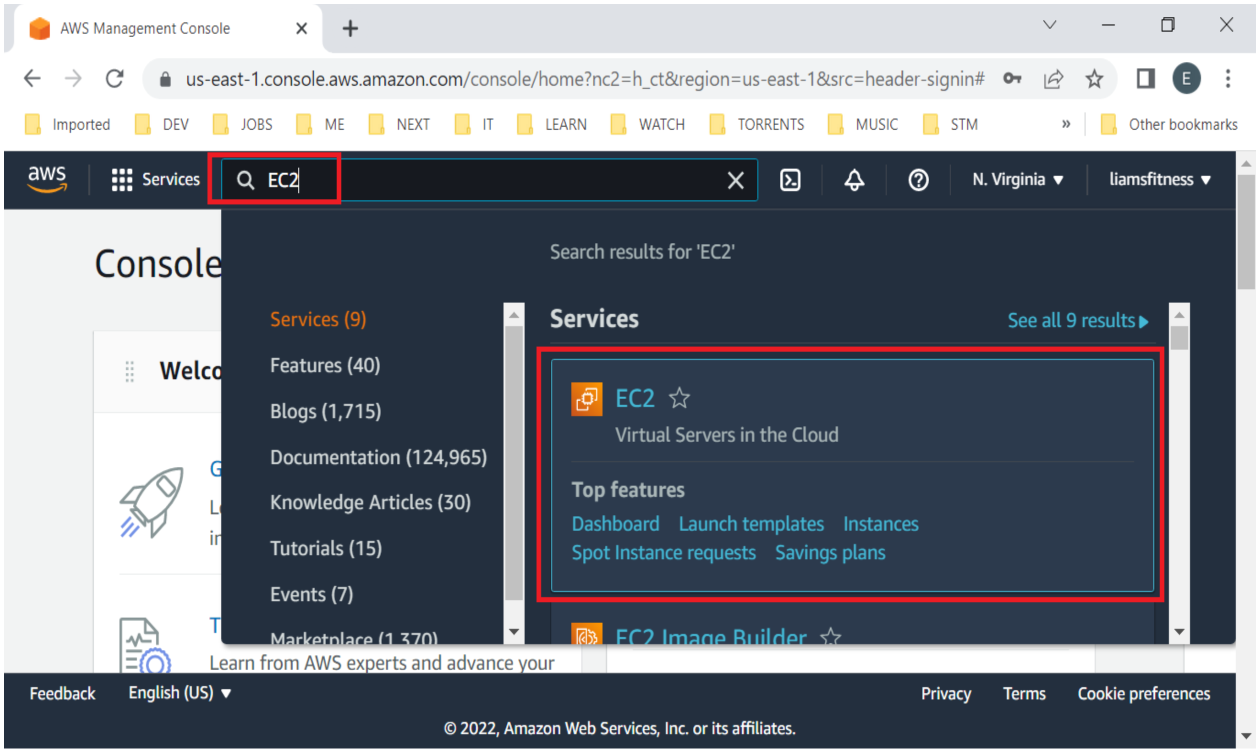
Steps to complete tutorial:

I will be running the shell script on one of my instances to check the services on my other instance. In order to do this, I will need SSH connection details for each instance: **`Public IPv4 DNS & Key Pair`**

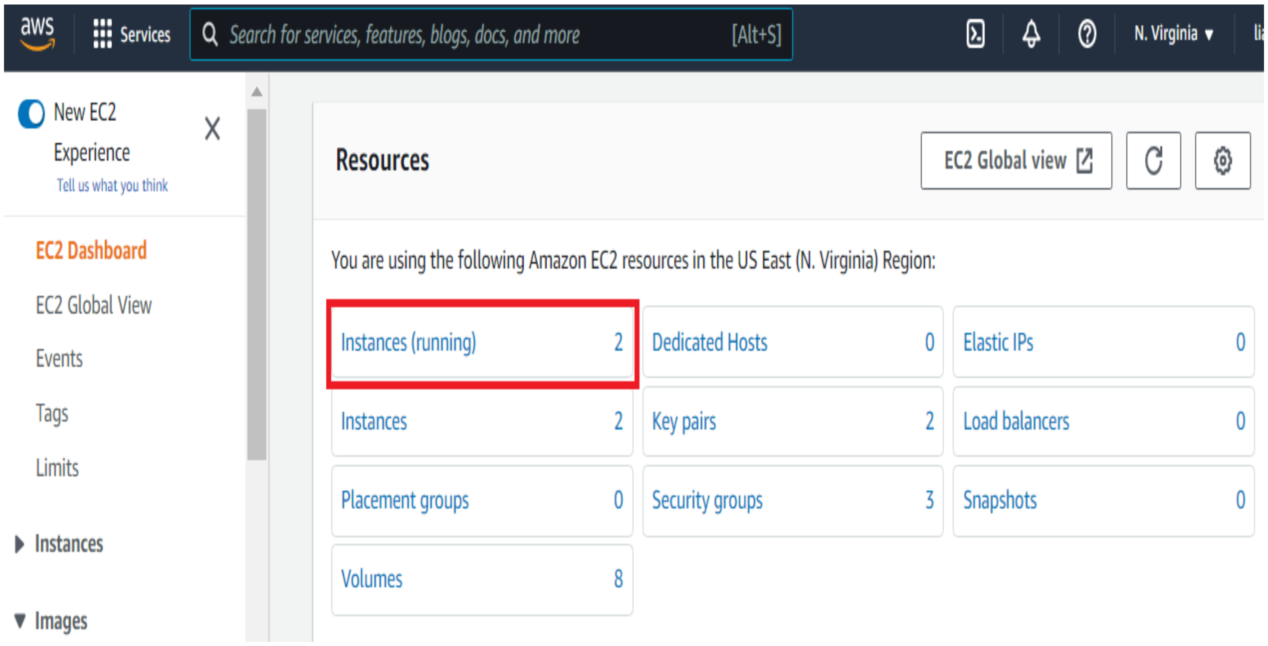
To begin, go to the following website, https://aws.amazon.com/console/ and log in to the console.



Once logged in, enter **EC2** in the search bar and select **EC2** Virtual Servers in the Cloud.



On the EC2 Dashboard, select **Instances (running)**

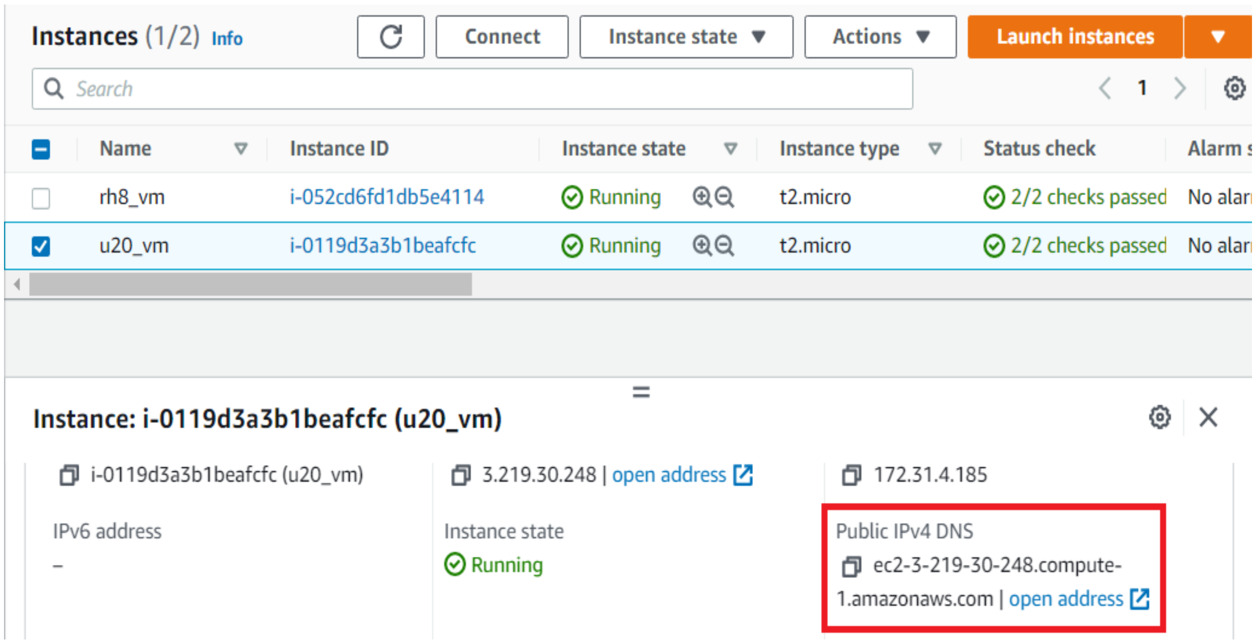


You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	2	Dedicated Hosts	0	Elastic IPs	0
Instances	2	Key pairs	2	Load balancers	0
Placement groups	0	Security groups	3	Snapshots	0
Volumes	8				

Determine EC2 Public IPv4 DNS

On the Instances screen, ensure an EC2 instance is selected and that the **Details** tab is selected. I will start with my Ubuntu 20 EC2 instance (**u20_vm**). Note the value for **Public IPv4 DNS**.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm s
☐	rh8_vm	i-052cd6fd1db5e4114	Running	t2.micro	2/2 checks passed	No alar
☑	u20_vm	i-0119d3a3b1beafcfc	Running	t2.micro	2/2 checks passed	No alar

Instance: i-0119d3a3b1beafcfc (u20_vm)

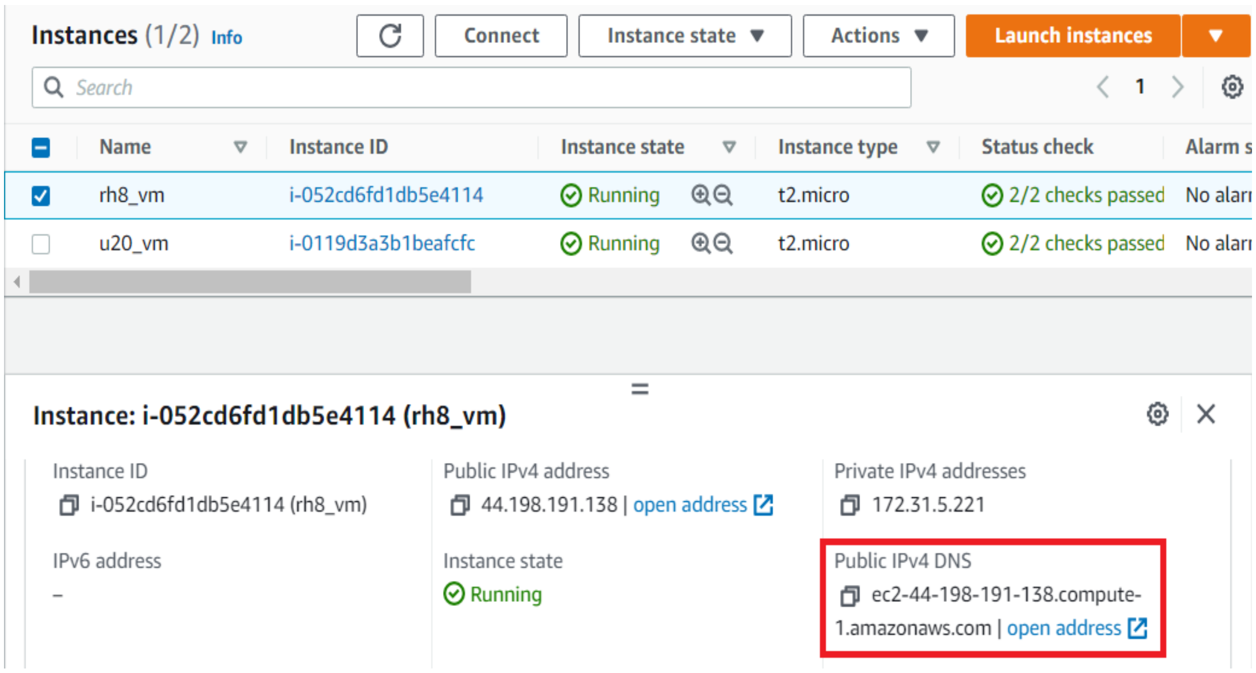
i-0119d3a3b1beafcfc (u20_vm)
3.219.30.248 | open address
172.31.4.185

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-3-219-30-248.compute-1.amazonaws.com | open address

I will now switch to my RHEL 8 EC2 instance (**rh8_vm**) and note the value for **Public IPv4 DNS**.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm s
☑	rh8_vm	i-052cd6fd1db5e4114	Running	t2.micro	2/2 checks passed	No alar
☐	u20_vm	i-0119d3a3b1beafcfc	Running	t2.micro	2/2 checks passed	No alar

Instance: i-052cd6fd1db5e4114 (rh8_vm)

Instance ID
i-052cd6fd1db5e4114 (rh8_vm)

Public IPv4 address
44.198.191.138 | open address

Private IPv4 addresses
172.31.5.221

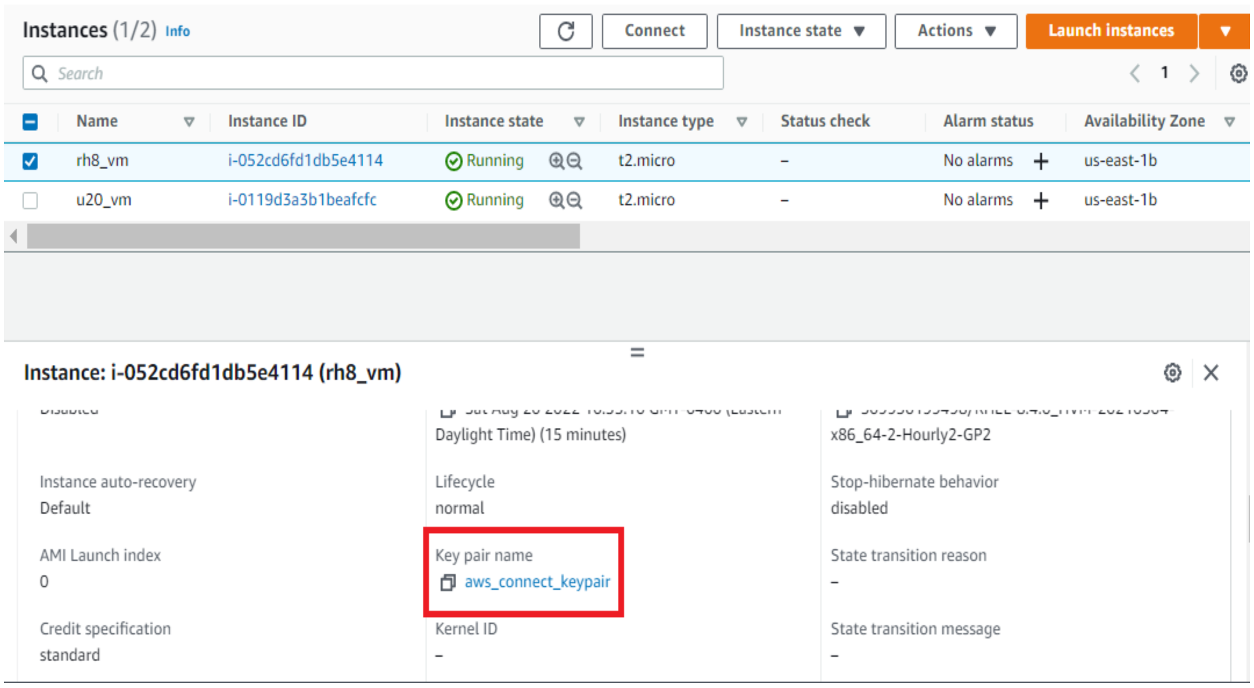
IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-44-198-191-138.compute-1.amazonaws.com | open address

Determine EC2 Key Pairs

Since my RHEL 8 EC2 is already selected, under the **Details** tab, I will scroll down until **Key pair** is visible. Then, I note the name: `aws_connect_keypair`



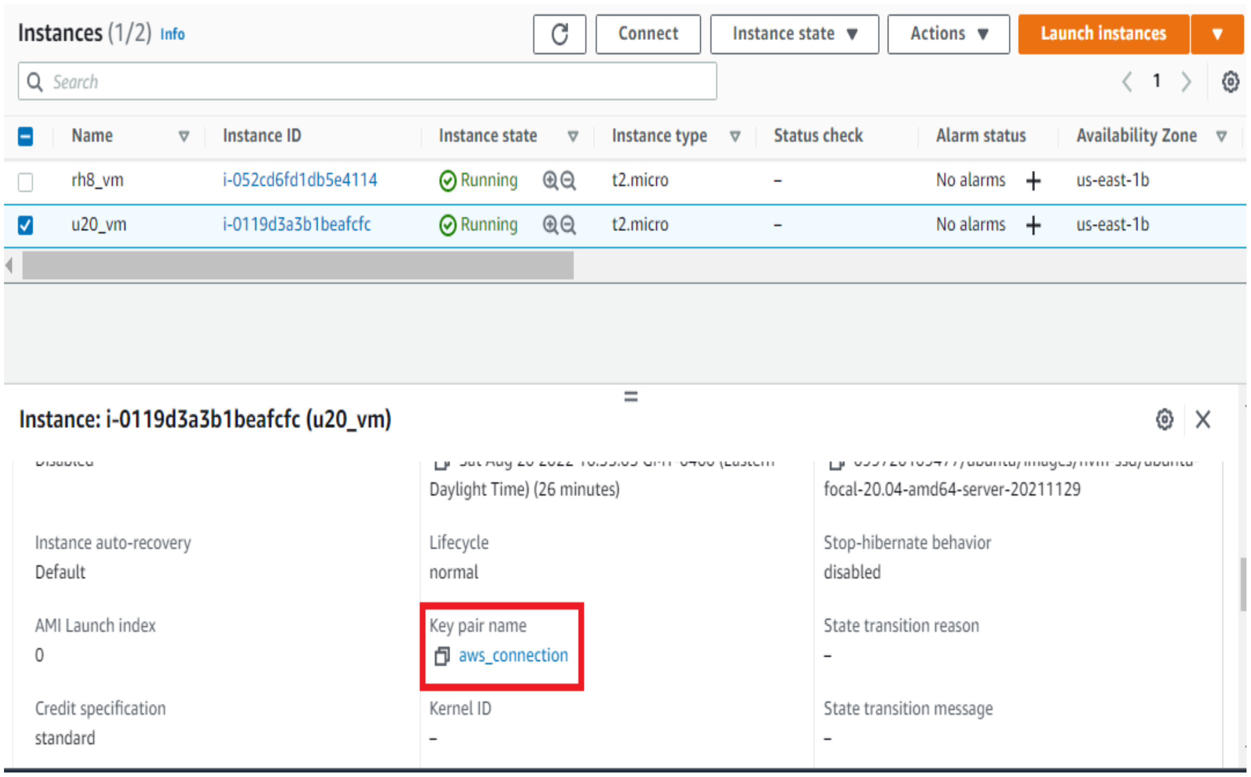
Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
rh8_vm	i-052cd6fd1db5e4114	Running	t2.micro	–	No alarms	us-east-1b
u20_vm	i-0119d3a3b1beafcfc	Running	t2.micro	–	No alarms	us-east-1b

Instance: i-052cd6fd1db5e4114 (rh8_vm)

	Daylight Time) (15 minutes)	x86_64-2-Hourly2-GP2
Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior disabled
AMI Launch index 0	Key pair name aws_connect_keypair	State transition reason –
Credit specification standard	Kernel ID –	State transition message –

I will now switch to my Ubuntu 20 EC2 instance (**u20_vm**) and under the **Details** tab, I will scroll down until **Key pair** is visible. Then, I note the name: `aws_connection`



Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
rh8_vm	i-052cd6fd1db5e4114	Running	t2.micro	–	No alarms	us-east-1b
u20_vm	i-0119d3a3b1beafcfc	Running	t2.micro	–	No alarms	us-east-1b

Instance: i-0119d3a3b1beafcfc (u20_vm)

	Daylight Time) (26 minutes)	focal-20.04-amd64-server-20211129
Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior disabled
AMI Launch index 0	Key pair name aws_connection	State transition reason –
Credit specification standard	Kernel ID –	State transition message –

When creating a Linux EC2 instance, a key pair must be specified (either newly created or an existing one can be reused). The key pair contains a public key and private key. The public key is stored on the EC2 instance, while you must store the private key on your local PC. The private key allows you to securely SSH into the EC2 instance.

To check services running on a remote EC2, I will need to execute commands using the **ssh** client. In order to do this from the source EC2, I will need a copy of the remote EC2's private key, as well as, the corresponding username and Public IPv4 DNS.

Copy Private Key to EC2 Instance

From my recent search, I know that the Ubuntu 20 EC2 key pair is **aws_connection** and the RHEL 8 EC2 key pair is **aws_connect_keypair**. During EC2 creation, I downloaded each key pair to my local machine: **aws_connection.pem** & **aws_connect_keypair.pem**

To securely copy a file from my local PC to an EC2, I use the following:

```
scp -i <remote-EC2-priv-key> <local-file-to-send> <username>@<remote-EC2-ipv4-dns>:/remote/dir
```

Please note that I am on Windows 10 and although a built-in SSH client exists, I have installed GitBash which includes an SSH client, as well as, many other Unix/Linux utilities. My tutorial, **GitBash Install** can be accessed here.

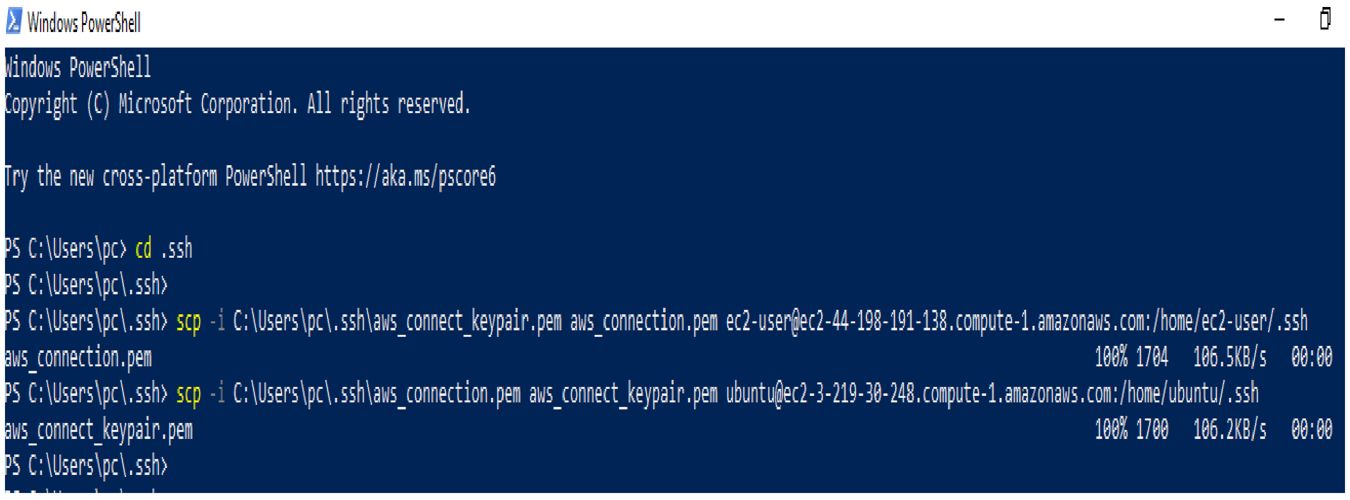
On my local machine, I will now securely copy the private key from my Ubuntu 20 EC2 key pair to my RHEL 8 EC2.

```
cd .ssh
```

```
scp -i C:\Users\pc\.ssh\aws_connect_keypair.pem aws_connection.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com:/home/ec2-user/.ssh
```

Next, I will securely copy the private key from my RHEL 8 EC2 key pair to my Ubuntu 20 EC2.

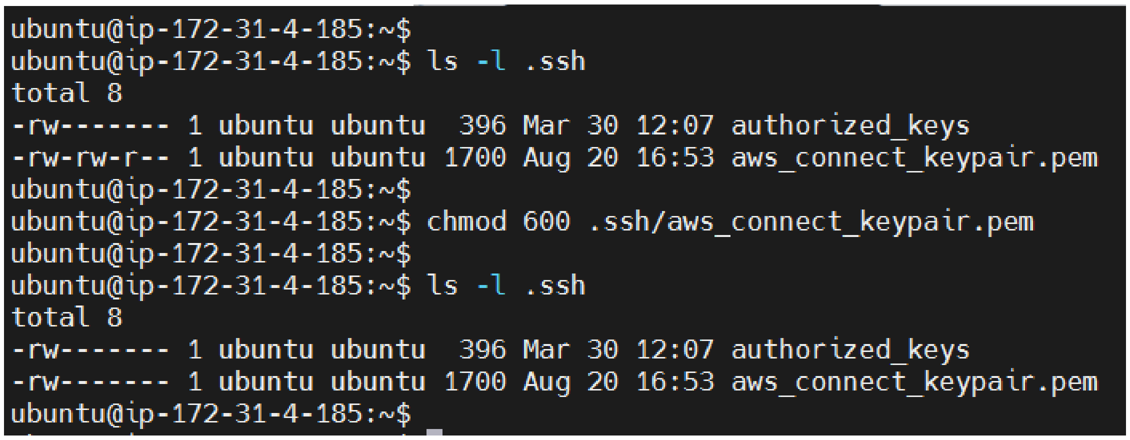
```
scp -i C:\Users\pc\.ssh\aws_connection.pem aws_connect_keypair.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com:/home/ubuntu/.ssh
```


Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pc> cd .ssh
PS C:\Users\pc\.ssh>
PS C:\Users\pc\.ssh> scp -i C:\Users\pc\.ssh\aws_connect_keypair.pem aws_connection.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com:/home/ec2-user/.ssh
aws_connection.pem                                                                                    100% 1704   106.5KB/s   00:00
PS C:\Users\pc\.ssh> scp -i C:\Users\pc\.ssh\aws_connection.pem aws_connect_keypair.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com:/home/ubuntu/.ssh
aws_connect_keypair.pem                                                                               100% 1700   106.2KB/s   00:00
PS C:\Users\pc\.ssh>
```

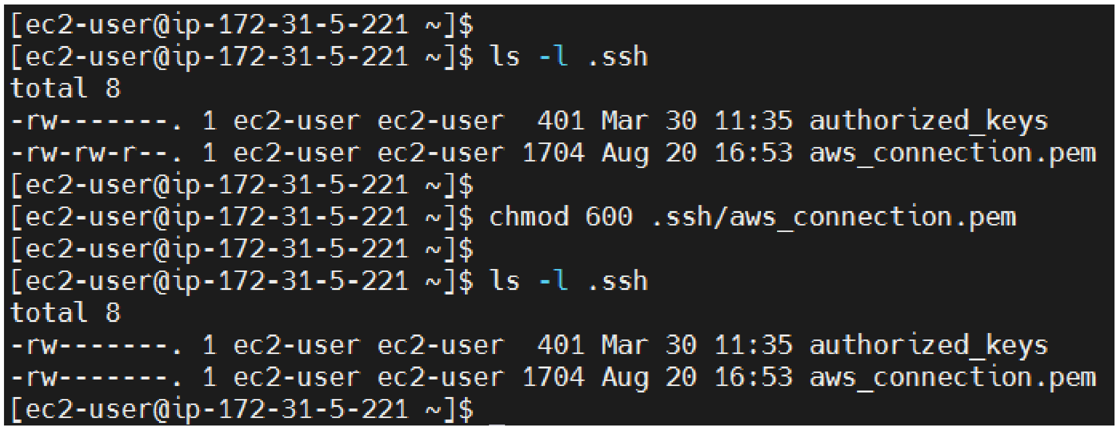
I will now confirm that the files were securely copied via my EC2 SSH connections. Then, before testing SSH connectivity between EC2 instances, I will first secure the copied private keys with restrictive permissions.

```
# on Ubuntu 20 EC2 (u20_vm)
$ ls -l .ssh
# secure RHEL 8 EC2 private key
$ chmod 600 aws_connect_keypair.pem
$ ls -l .ssh
```



```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ ls -l .ssh
total 8
-rw------- 1 ubuntu ubuntu  396 Mar 30 12:07 authorized_keys
-rw-rw-r-- 1 ubuntu ubuntu 1700 Aug 20 16:53 aws_connect_keypair.pem
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ chmod 600 .ssh/aws_connect_keypair.pem
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ ls -l .ssh
total 8
-rw------- 1 ubuntu ubuntu  396 Mar 30 12:07 authorized_keys
-rw------- 1 ubuntu ubuntu 1700 Aug 20 16:53 aws_connect_keypair.pem
ubuntu@ip-172-31-4-185:~$
```

```
# on RHEL 8 EC2 (rh8_vm)
$ ls -l .ssh
# secure Ubuntu 20 EC2 private key
$ chmod 600 aws_connection.pem
$ ls -l .ssh
```



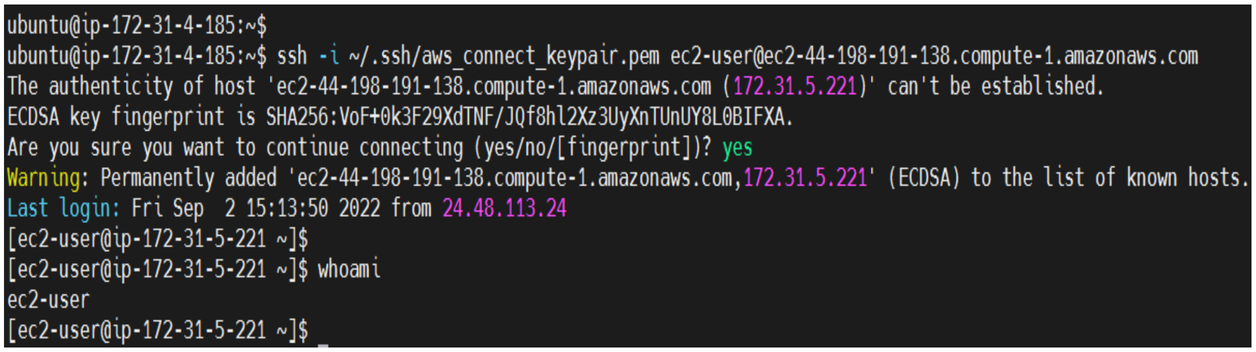
```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ ls -l .ssh
total 8
-rw-------. 1 ec2-user ec2-user  401 Mar 30 11:35 authorized_keys
-rw-rw-r--. 1 ec2-user ec2-user 1704 Aug 20 16:53 aws_connection.pem
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ chmod 600 .ssh/aws_connection.pem
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ ls -l .ssh
total 8
-rw-------. 1 ec2-user ec2-user  401 Mar 30 11:35 authorized_keys
-rw-------. 1 ec2-user ec2-user 1704 Aug 20 16:53 aws_connection.pem
[ec2-user@ip-172-31-5-221 ~]$
```

Test SSH Connection Between EC2s

I am now ready to test SSH connectivity between EC2 instances.

First, I will test an SSH connection from my Ubuntu 20 EC2 to my RHEL 8 EC2.

```
ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
```

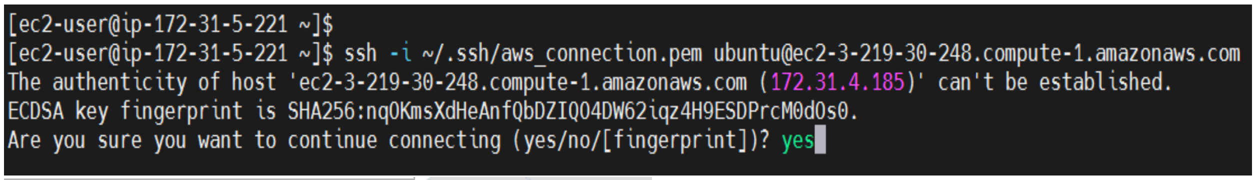


```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
The authenticity of host 'ec2-44-198-191-138.compute-1.amazonaws.com (172.31.5.221)' can't be established.
ECDSA key fingerprint is SHA256:VoF+0k3F29XdTNF/JQf8hl2Xz3UyXnTUnUY8L0BIFXA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-198-191-138.compute-1.amazonaws.com,172.31.5.221' (ECDSA) to the list of known hosts.
Last login: Fri Sep  2 15:13:50 2022 from 24.48.113.24
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-5-221 ~]$ _
```

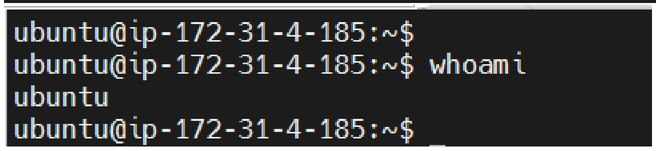
You will notice that after connecting to my RHEL 8 EC2, I was able to successfully execute a command.

Next, I will test an SSH connection from my RHEL 8 EC2 to my Ubuntu 20 EC2.

```
ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com
```



```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com
The authenticity of host 'ec2-3-219-30-248.compute-1.amazonaws.com (172.31.4.185)' can't be established.
ECDSA key fingerprint is SHA256:nqOKmsXdHeAnfQbDZIQ04DW62iqz4H9ESDPrcM0d0s0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

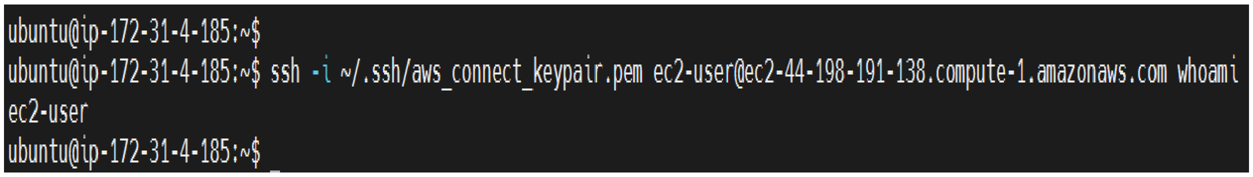


```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ whoami
ubuntu
ubuntu@ip-172-31-4-185:~$ _
```

You will notice that after connecting to my Ubuntu 20 EC2, I was able to successfully execute a command.

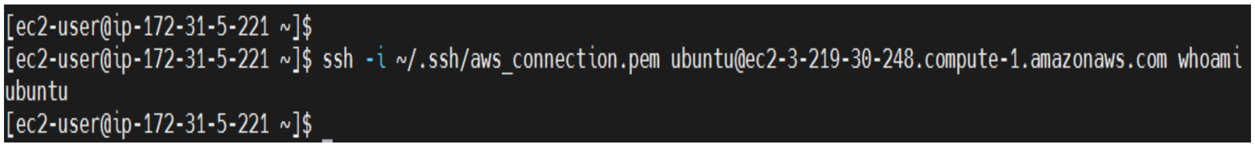
After successfully confirming SSH connectivity between EC2 instances, I will now confirm that I can execute a command remotely using **ssh**.

```
# from Ubuntu 20 EC2 to RHEL 8 EC2
ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
whoami
```



```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com whoami
ec2-user
ubuntu@ip-172-31-4-185:~$ _
```

```
# from RHEL 8 EC2 to Ubuntu 20 EC2
ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com whoami
```



```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com whoami
ubuntu
[ec2-user@ip-172-31-5-221 ~]$ _
```

I have confirmed that I can remotely execute commands using ssh from either of my EC2 instances. I am now ready to create a shell script to check remote services.

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_remote_service.sh`**

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

```
# on RHEL 8 EC2
/home/ec2-user/scripts/check_remote_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 is running remotely on an EC2 instance
```

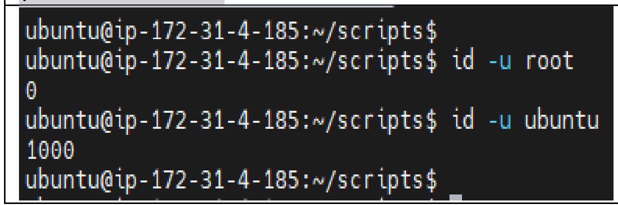
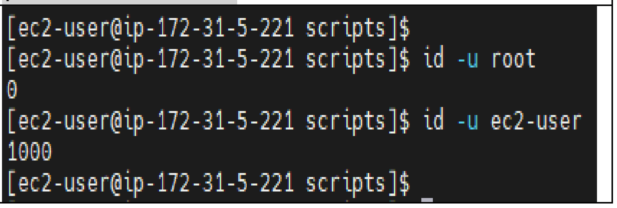
Ensure Superuser Privileges

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]$ _
```

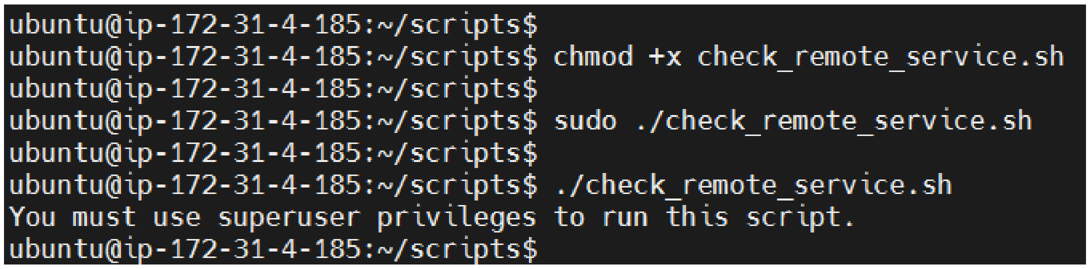
We can see from the above command outputs that the root user has a UID of 0 and the non-root users each have a UID of 1000.

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_remote_service.sh
$ sudo ./check_remote_service.sh
$ ./check_remote_service.sh
```


```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ chmod +x check_remote_service.sh
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_remote_service.sh
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ ./check_remote_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

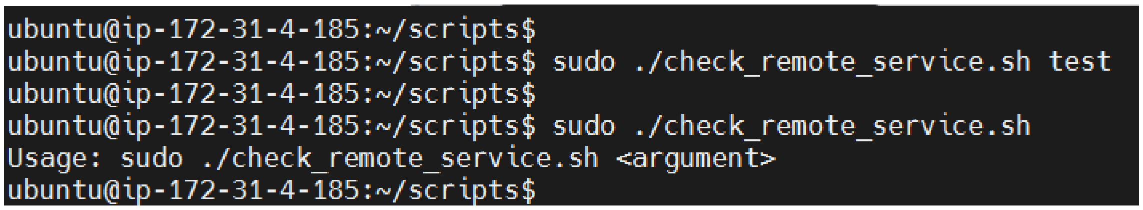
Next, I want to ensure that an argument is passed to the script.

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_remote_service.sh <argument>"
        exit 1
fi
```

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

```
$ sudo ./check_remote_service.sh test
$ sudo ./check_remote_service.sh
```


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

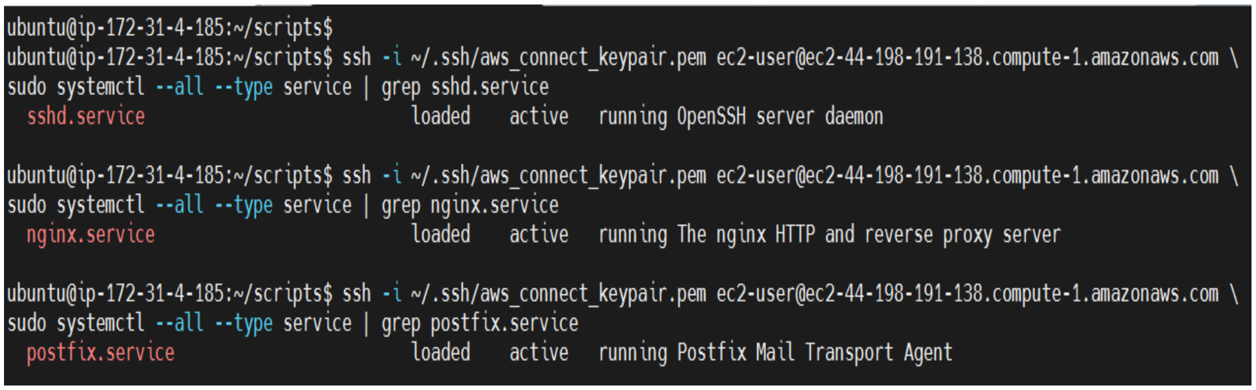
Again, you will notice that executing the script without an argument generated the helpful usage message and exited the script.

Before proceeding, I want to verify that the three remote services (**ssh, nginx, postfix**) are available on both my Ubuntu 20 & RHEL 8 EC2 instances. Again, I will use **ssh** to execute the commands remotely.

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.

```
# check RHEL 8 EC2 services from Ubuntu 20 EC2
$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep sshd.service

$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep nginx.service
```

```
$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep postfix.service
```

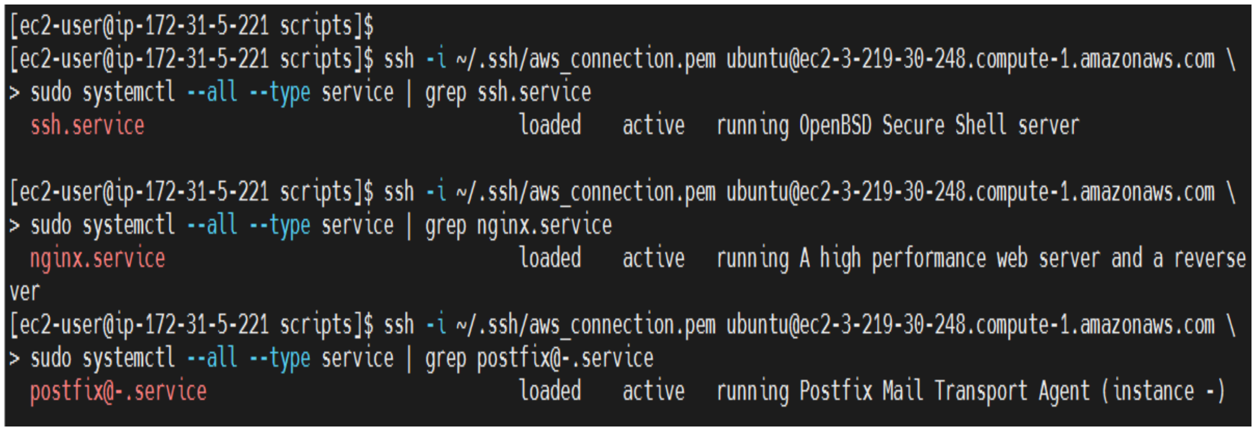


```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep sshd.service
  sshd.service                        loaded    active   running OpenSSH server daemon

ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep nginx.service
  nginx.service                       loaded    active   running The nginx HTTP and reverse proxy server

ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep postfix.service
  postfix.service                     loaded    active   running Postfix Mail Transport Agent
```

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

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

```
# check Ubuntu 20 EC2 services from RHEL 8 EC2
$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep ssh.service

$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep nginx.service

$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com \
sudo systemctl --all --type service | grep postfix@-.service
```



```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com \
> sudo systemctl --all --type service | grep ssh.service
  ssh.service                         loaded    active   running OpenBSD Secure Shell server

[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com \
> sudo systemctl --all --type service | grep nginx.service
  nginx.service                       loaded    active   running A high performance web server and a reverse
ver
[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com \
> sudo systemctl --all --type service | grep postfix@-.service
  postfix@-.service                   loaded    active   running Postfix Mail Transport Agent (instance -)
```

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

Before proceeding I will assign the remote SSH command details to variables to reduce clutter in my script moving forward. Keep in mind which EC2 script.

```
# in Ubuntu 20 EC2 script RHEL 8 EC2 connect details
KEY=/home/ubuntu/.ssh/aws_connect_keypair.pem
USER=ec2-user
DNS=ec2-44-198-191-138.compute-1.amazonaws.com
CONNECT="ssh -i ${KEY} ${USER}@${DNS}"
```

```
# in RHEL 8 EC2 script Ubuntu 20 EC2 connect details
KEY=/home/ec2-user/.ssh/aws_connection.pem
USER=ubuntu
DNS=ec2-3-219-30-248.compute-1.amazonaws.com
CONNECT="ssh -i ${KEY} ${USER}@${DNS}"
```

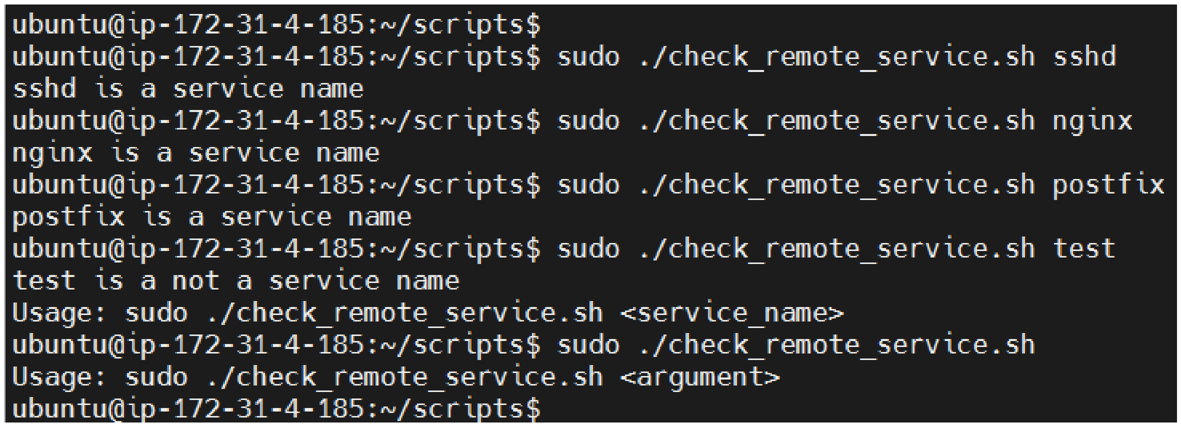
Next, I want to ensure that a single service name is provided as the argument to the script.

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

```
# Ensure a valid service name argument is passed to the script
if [[ "$#" -eq 1 ]]
then
        RESULT=$(${CONNECT} 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_remote_service.sh <service_name>"
                exit 1
        fi
else
        echo "Usage: sudo ./check_remote_service.sh <service_name>"
        exit 1
fi
```

I will now test this out, for success & failure, on my Ubuntu 20 EC2 to check remote RHEL 8 services.

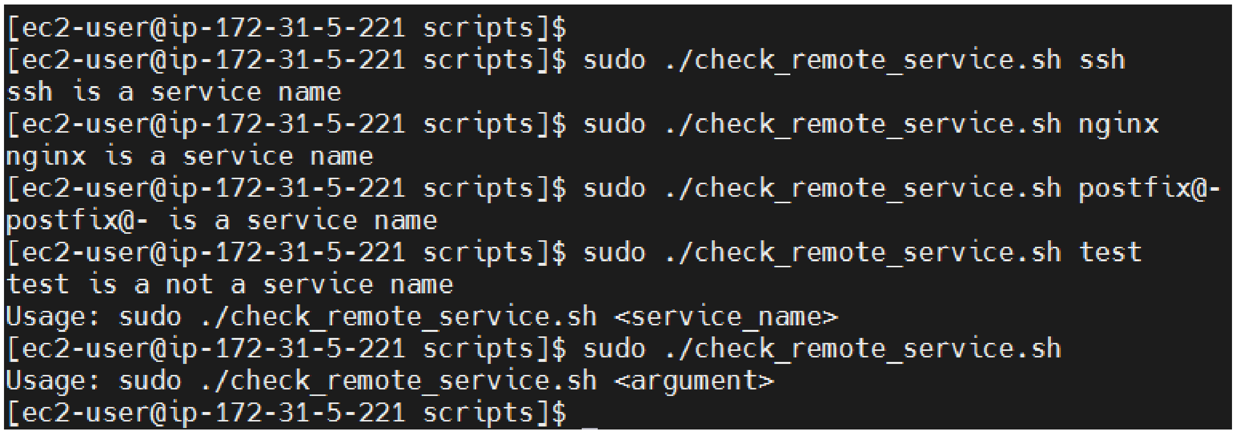
```
$ sudo ./check_remote_service.sh sshd
$ sudo ./check_remote_service.sh nginx
$ sudo ./check_remote_service.sh postfix
$ sudo ./check_remote_service.sh test
$ sudo ./check_remote_service.sh
```



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

Now, I will test this out, for success & failure, from my RHEL 8 EC2 to check remote Ubuntu 20 services.

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



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

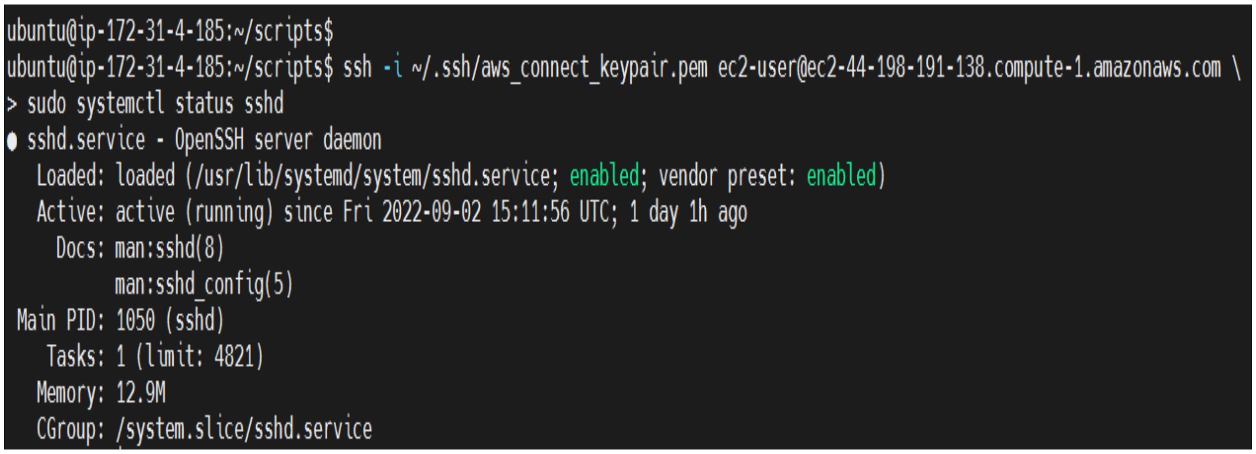
We've confirmed that a service name must be provided as an argument to the script.

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

Service Status

We could use **systemctl** to check the **status** of a remote service, but then we would have to parse the output to extract what we need and assign it to a variable.

```
# on Ubuntu 20 EC2 checking remote RHEL 8 EC2 SSHD service status
$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
sudo systemctl status sshd
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com \
> sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-09-02 15:11:56 UTC; 1 day 1h ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1050 (sshd)
    Tasks: 1 (limit: 4821)
   Memory: 12.9M
   CGroup: /system.slice/sshd.service
```

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:

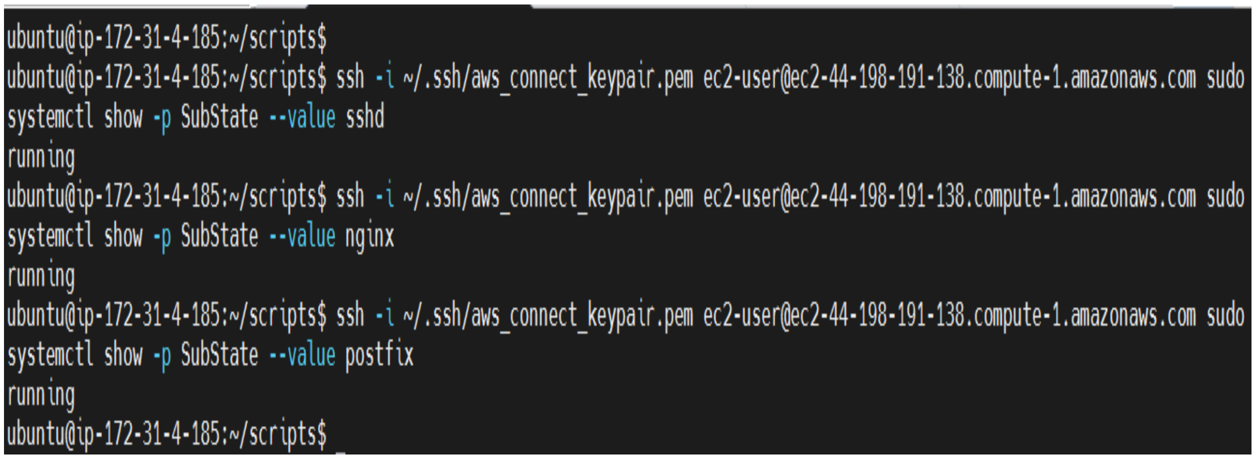
```
$ sudo systemctl show <service_name>
```

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>
```

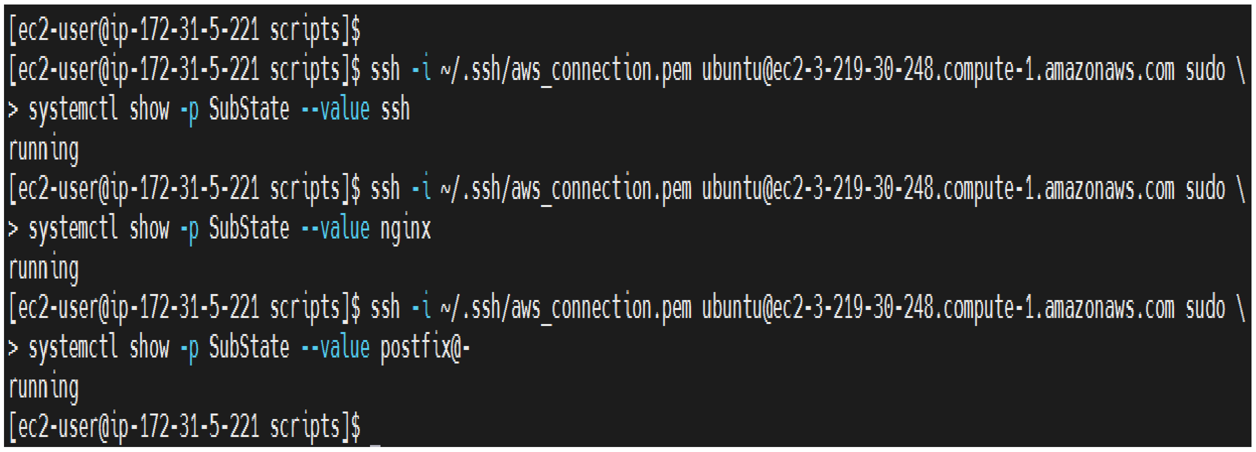
In our case, we need to determine a service unit's **SubState** property value.

```
# on Ubuntu 20 EC2 checking remote RHEL 8 EC2 service status
$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
sudo systemctl show -p SubState --value sshd

$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
sudo systemctl show -p SubState --value nginx

$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
sudo systemctl show -p SubState --value postfix
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com sudo
systemctl show -p SubState --value sshd
running
ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com sudo
systemctl show -p SubState --value nginx
running
ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com sudo
systemctl show -p SubState --value postfix
running
ubuntu@ip-172-31-4-185:~/scripts$ _
```

```
# on RHEL 8 EC2 checking remote Ubuntu 20 EC2 service status
$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com sudo \
systemctl show -p SubState --value ssh

$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com sudo \
systemctl show -p SubState --value nginx

$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com sudo \
systemctl show -p SubState --value postfix@-
```



```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com sudo \
> systemctl show -p SubState --value ssh
running
[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com sudo \
> systemctl show -p SubState --value nginx
running
[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com 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 gather a few details about the remote host. Finally, I will generate a helpful message that I will use during testing.

```
SERVICE="${1}"
STATUS="$(${CONNECT} systemctl show -p SubState --value ${SERVICE})"
HOST=$(${CONNECT} hostname)
IP_ADDR=$(${CONNECT} hostname -i)
MESSAGE="$(echo "${SERVICE} is ${STATUS} on ${HOST} at ${IP_ADDR}")"
```

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="${LOGDIR}/remote_${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
              ${CONNECT} “sudo systemctl restart ${SERVICE%@.}" &>> ${LOGFILE}
       else
              ${CONNECT} “sudo systemctl restart ${SERVICE}” &>> ${LOGFILE}
       fi

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

```
                echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
        fi
else
        echo "${MESSAGE}"
        echo "${TIMESTAMP} -- ${MESSAGE}" &>> ${LOGFILE}
fi

exit 0
```

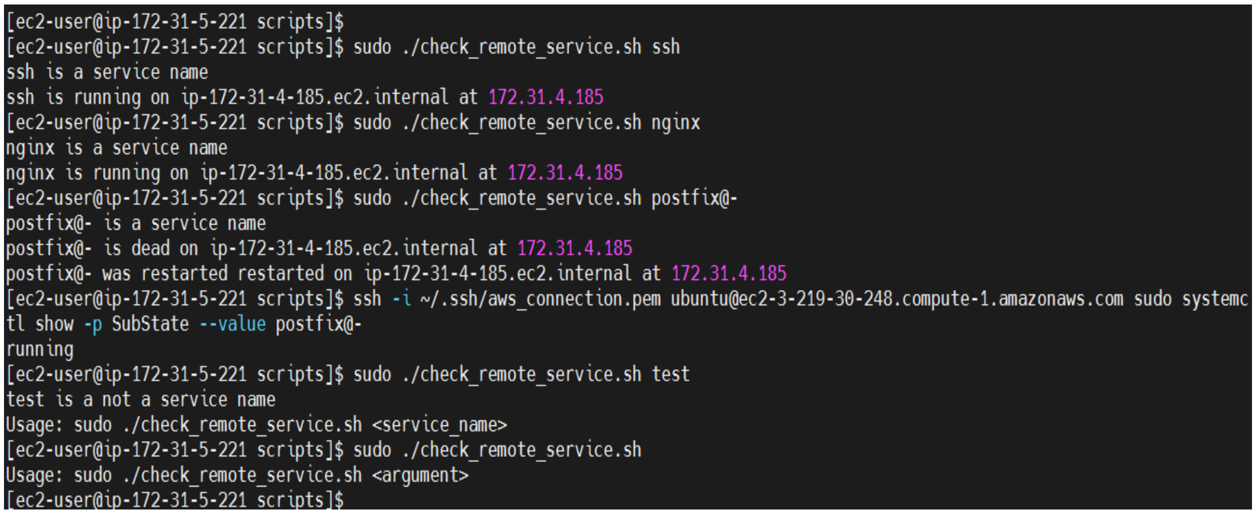
Please note that the SSH/SSHD service must be checked locally. I have included it for testing purposes.

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

```
$ sudo systemctl stop postfix
```

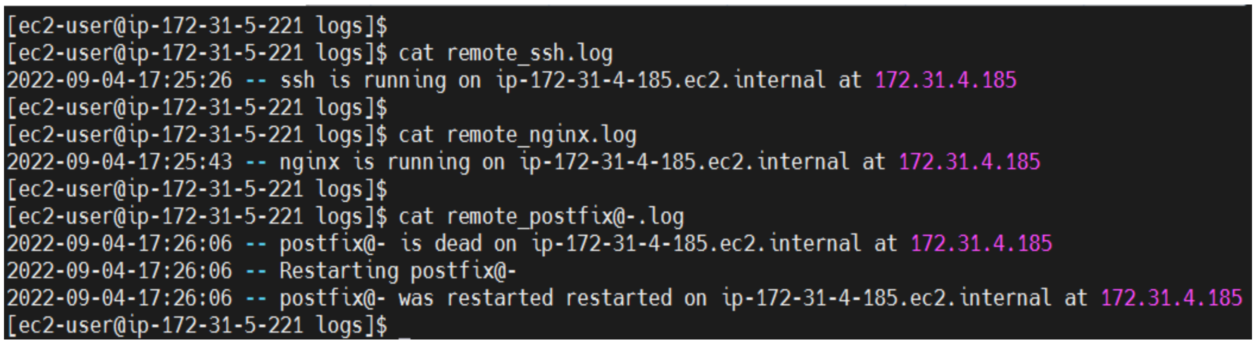
I will now test, for success & failure, from my RHEL 8 EC2 to check remote Ubuntu 20 services.

```
$ sudo ./check_remote_service.sh ssh
$ sudo ./check_remote_service.sh nginx
$ sudo ./check_remote_service.sh postfix@-
$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com
sudo systemctl show -p SubState --value postfix@-

$ sudo ./check_remote_service.sh test
$ sudo ./check_remote_service.sh
```



```
[ec2-user@ip-172-31-5-221 scripts]$
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_remote_service.sh ssh
ssh is a service name
ssh is running on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_remote_service.sh nginx
nginx is a service name
nginx is running on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_remote_service.sh postfix@-
postfix@- is a service name
postfix@- is dead on ip-172-31-4-185.ec2.internal at 172.31.4.185
postfix@- was restarted restarted on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 scripts]$ ssh -i ~/.ssh/aws_connection.pem ubuntu@ec2-3-219-30-248.compute-1.amazonaws.com sudo systemc
tl show -p SubState --value postfix@-
running
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_remote_service.sh test
test is a not a service name
Usage: sudo ./check_remote_service.sh <service_name>
[ec2-user@ip-172-31-5-221 scripts]$ sudo ./check_remote_service.sh
Usage: sudo ./check_remote_service.sh <argument>
[ec2-user@ip-172-31-5-221 scripts]$
```

I can now check the log files generated from my testing on my RHEL 8 20 EC2:



```
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat remote_ssh.log
2022-09-04-17:25:26 -- ssh is running on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat remote_nginx.log
2022-09-04-17:25:43 -- nginx is running on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 logs]$
[ec2-user@ip-172-31-5-221 logs]$ cat remote_postfix@-.log
2022-09-04-17:26:06 -- postfix@- is dead on ip-172-31-4-185.ec2.internal at 172.31.4.185
2022-09-04-17:26:06 -- Restarting postfix@-
2022-09-04-17:26:06 -- postfix@- was restarted restarted on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 logs]$ _
```

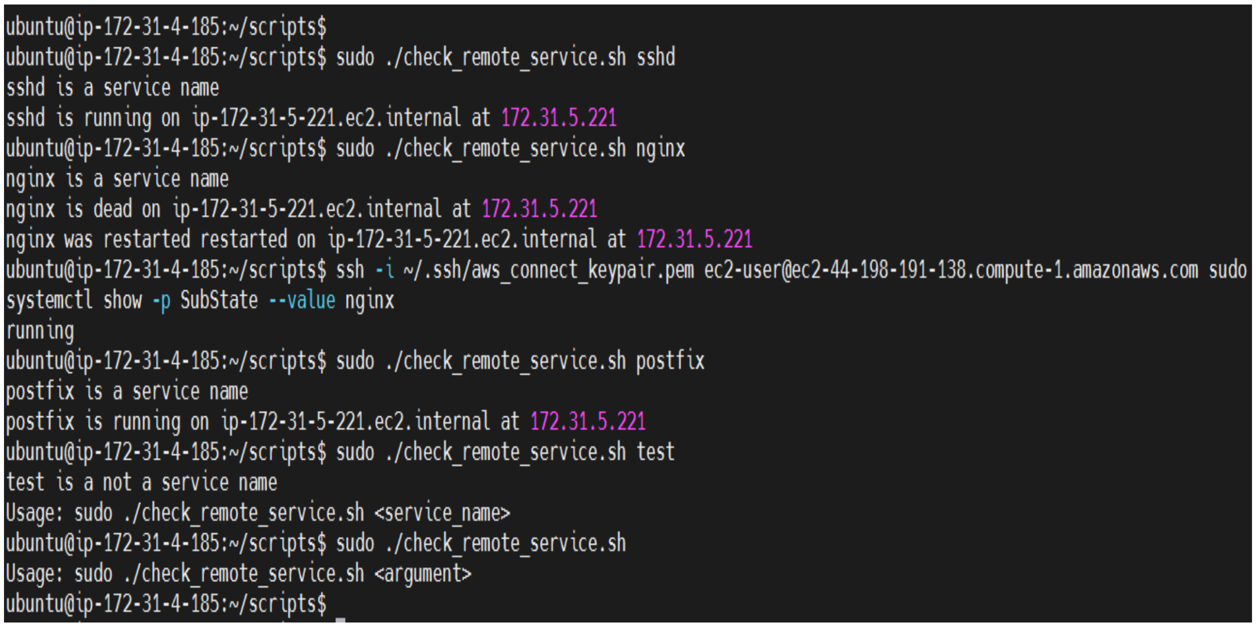
I will now manually stop the nginx service on my RHEL 8 EC2.

```
$ sudo systemctl stop nginx
```

Now, I will test this out, for success & failure, from my Ubuntu 20 EC2 to check remote RHEL 8 services.

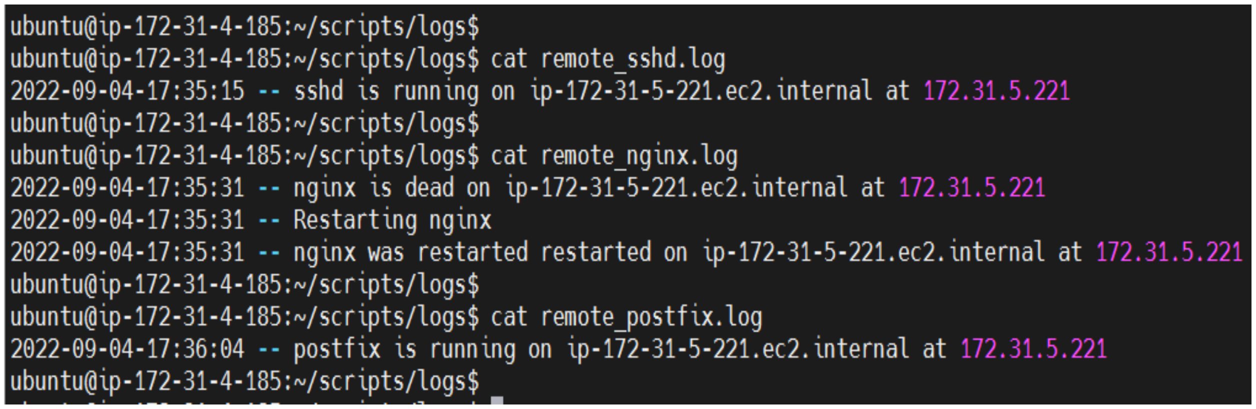
```
$ sudo ./check_remote_service.sh sshd
$ sudo ./check_remote_service.sh nginx
$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com
sudo systemctl show -p SubState --value nginx
```

```
$ sudo ./check_remote_service.sh postfix
$ sudo ./check_remote_service.sh test
$ sudo ./check_remote_service.sh
```



```
ubuntu@ip-172-31-4-185:~/scripts$
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_remote_service.sh sshd
sshd is a service name
sshd is running on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_remote_service.sh nginx
nginx is a service name
nginx is dead on ip-172-31-5-221.ec2.internal at 172.31.5.221
nginx was restarted restarted on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~/scripts$ ssh -i ~/.ssh/aws_connect_keypair.pem ec2-user@ec2-44-198-191-138.compute-1.amazonaws.com sudo
systemctl show -p SubState --value nginx
running
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_remote_service.sh postfix
postfix is a service name
postfix is running on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_remote_service.sh test
test is a not a service name
Usage: sudo ./check_remote_service.sh <service_name>
ubuntu@ip-172-31-4-185:~/scripts$ sudo ./check_remote_service.sh
Usage: sudo ./check_remote_service.sh <argument>
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 remote_sshd.log
2022-09-04-17:35:15 -- sshd is running on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~/scripts/logs$
ubuntu@ip-172-31-4-185:~/scripts/logs$ cat remote_nginx.log
2022-09-04-17:35:31 -- nginx is dead on ip-172-31-5-221.ec2.internal at 172.31.5.221
2022-09-04-17:35:31 -- Restarting nginx
2022-09-04-17:35:31 -- nginx was restarted restarted on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~/scripts/logs$
ubuntu@ip-172-31-4-185:~/scripts/logs$ cat remote_postfix.log
2022-09-04-17:36:04 -- postfix is running on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~/scripts/logs$
```

The script worked on both of my EC2 instances.

Review Script

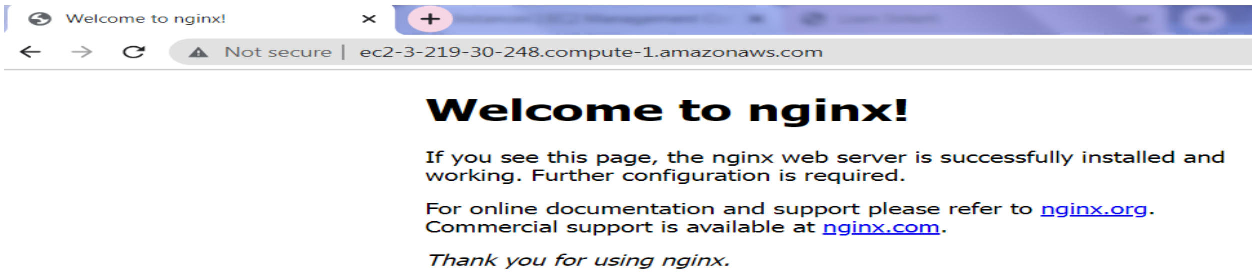
```
#!/bin/bash
#
# This script is used to check if a service is running remotely on an EC2 instance

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

# on my Ubuntu 20 EC2
#KEY=/home/ubuntu/.ssh/aws_connect_keypair.pem
#USER=ec2-user
#DNS=ec2-44-198-191-138.compute-1.amazonaws.com

# on my RHEL 8 EC2
#KEY=/home/ec2-user/.ssh/aws_connection.pem
#USER=ubuntu
#DNS=ec2-3-219-30-248.compute-1.amazonaws.com

CONNECT="ssh -i ${KEY} ${USER}@${DNS}"

# Ensure a valid service name argument is passed to the script
if [[ "$#" -eq 1 ]]
then
        RESULT=$(${CONNECT} 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_remote_service.sh <service_name>"
                exit 1
        fi
else
        echo "Usage: sudo ./check_remote_service.sh <service_name>"
        exit 1
fi

SERVICE="${1}"
STATUS="$(${CONNECT} systemctl show -p SubState --value ${SERVICE})"
HOST=$(${CONNECT} hostname)
IP_ADDR=$(${CONNECT} hostname -i)
MESSAGE="$(echo "${SERVICE} is ${STATUS} on ${HOST} at ${IP_ADDR}")"

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

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

touch "${LOGFILE}"

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
                ${CONNECT} "sudo systemctl restart ${SERVICE%@.}" &>> ${LOGFILE}
        else
                ${CONNECT} "sudo systemctl restart ${SERVICE}" &>> ${LOGFILE}
        fi

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

exit 0
```

Automate with CROND

I will create a cron job, on each of my EC2 instances, using: **`check_remote_service.sh`**. to check that the remote **nginx** service is running.

Before creating the cron job, I will confirm that the default **nginx** website is accessible from both of my EC2 instances. I will use the **Public IPv4 DNS** of each of my EC2 instances to confirm this.

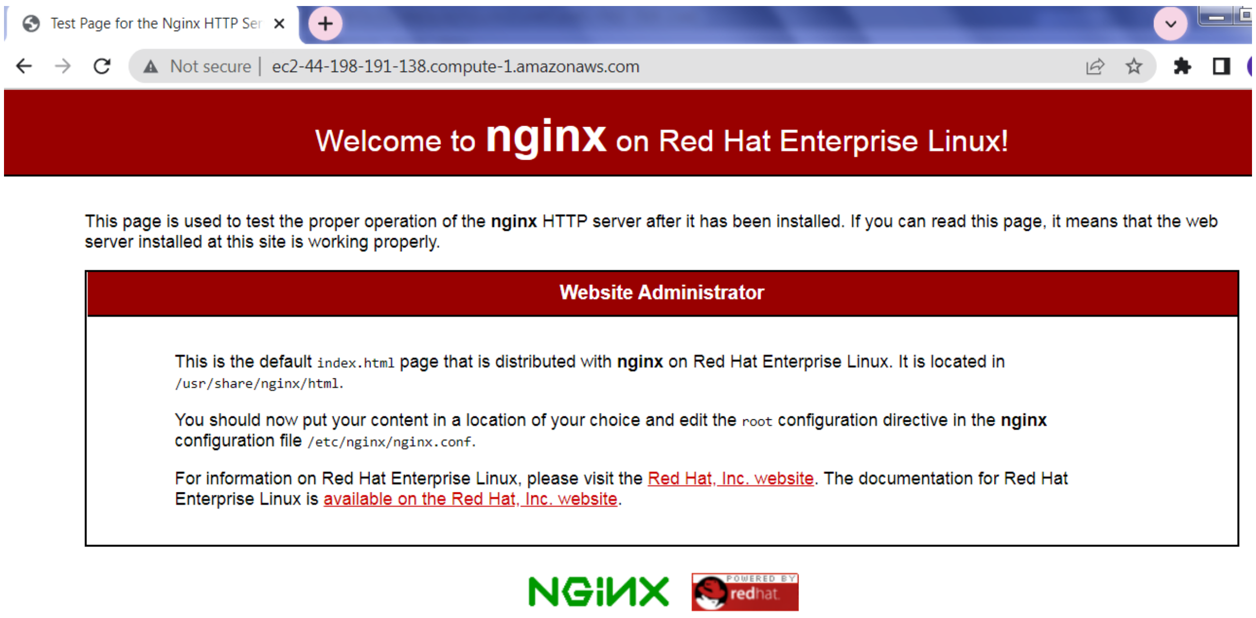


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.


Test Page for the Nginx HTTP Ser

Not secure | ec2-44-198-191-138.compute-1.amazonaws.com

Welcome to **nginx** on Red Hat Enterprise Linux!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

This is the default `index.html` page that is distributed with **nginx** on Red Hat Enterprise Linux. It is located in `/usr/share/nginx/html`.

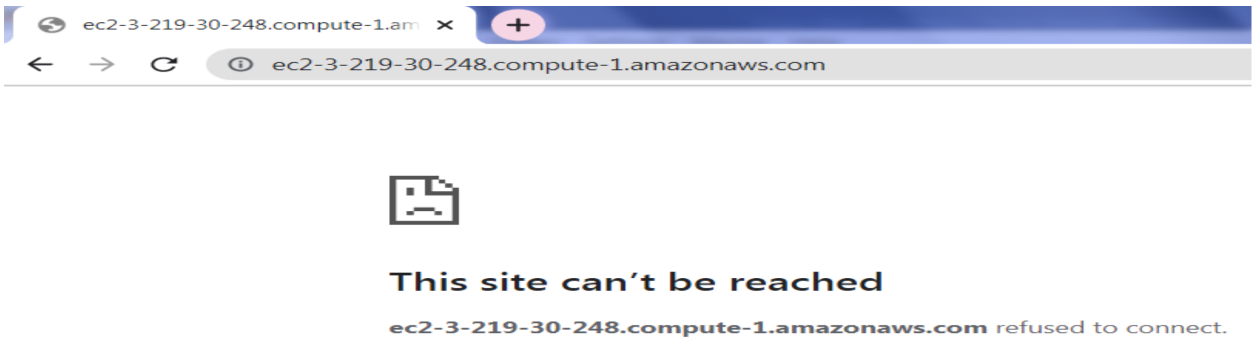
You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

For information on Red Hat Enterprise Linux, please visit the Red Hat, Inc. website. The documentation for Red Hat Enterprise Linux is available on the Red Hat, Inc. website.

Now, I will manually stop the **nginx** service on each EC2 instance.

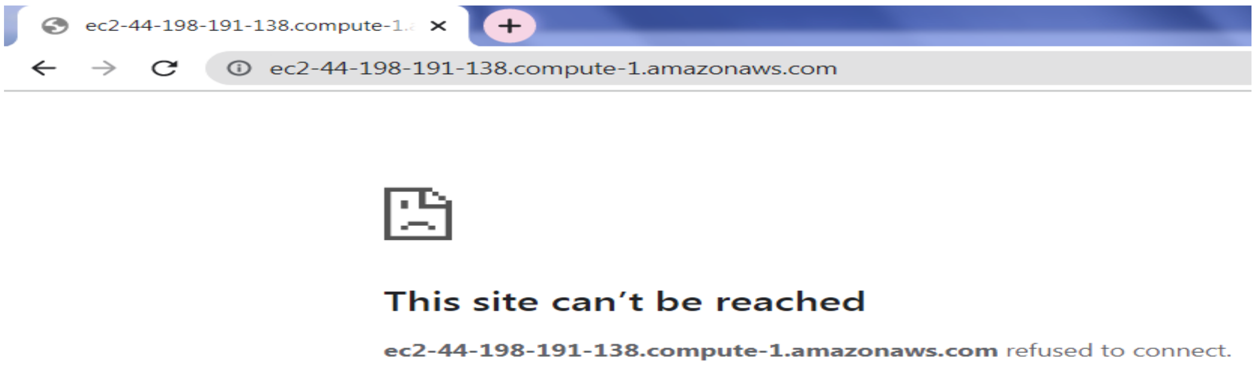
```
# Ubuntu 20 EC2
$ sudo systemctl stop nginx

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

I will now confirm that the default **nginx** website is not accessible from either of my EC2 instances.


ec2-3-219-30-248.compute-1.amazonaws.com

This site can't be reached

ec2-3-219-30-248.compute-1.amazonaws.com refused to connect.


ec2-44-198-191-138.compute-1.amazonaws.com

This site can't be reached

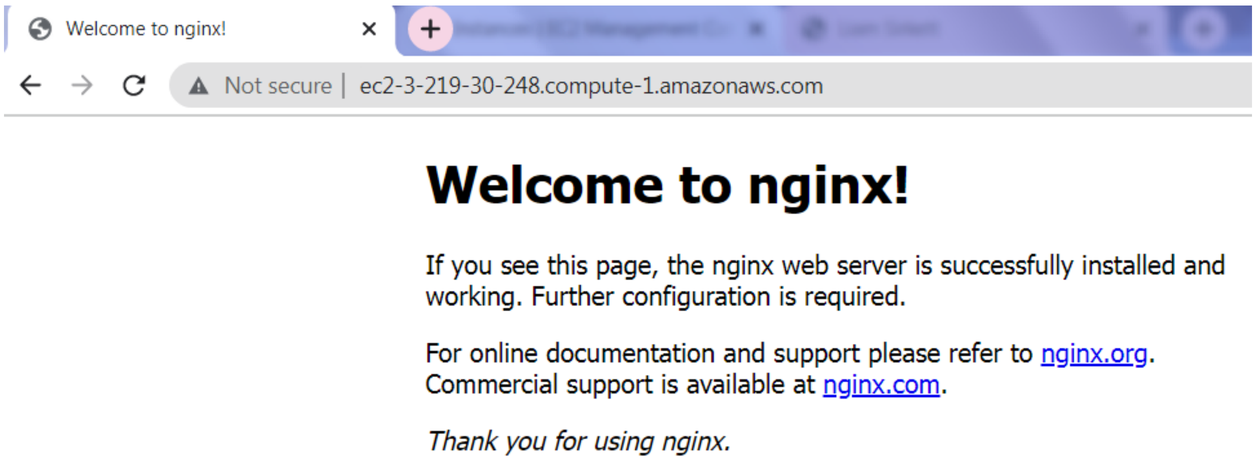
ec2-44-198-191-138.compute-1.amazonaws.com refused to connect.

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, check SSHD service on RHEL 8 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_remote_service.sh nginx &> /dev/null

# RHEL 8 EC2, check SSH service on Ubuntu 20 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_remote_service.sh nginx &> /dev/null
```

After cron job creation, I will wait five minutes and then try accessing the default **nginx** websites again.

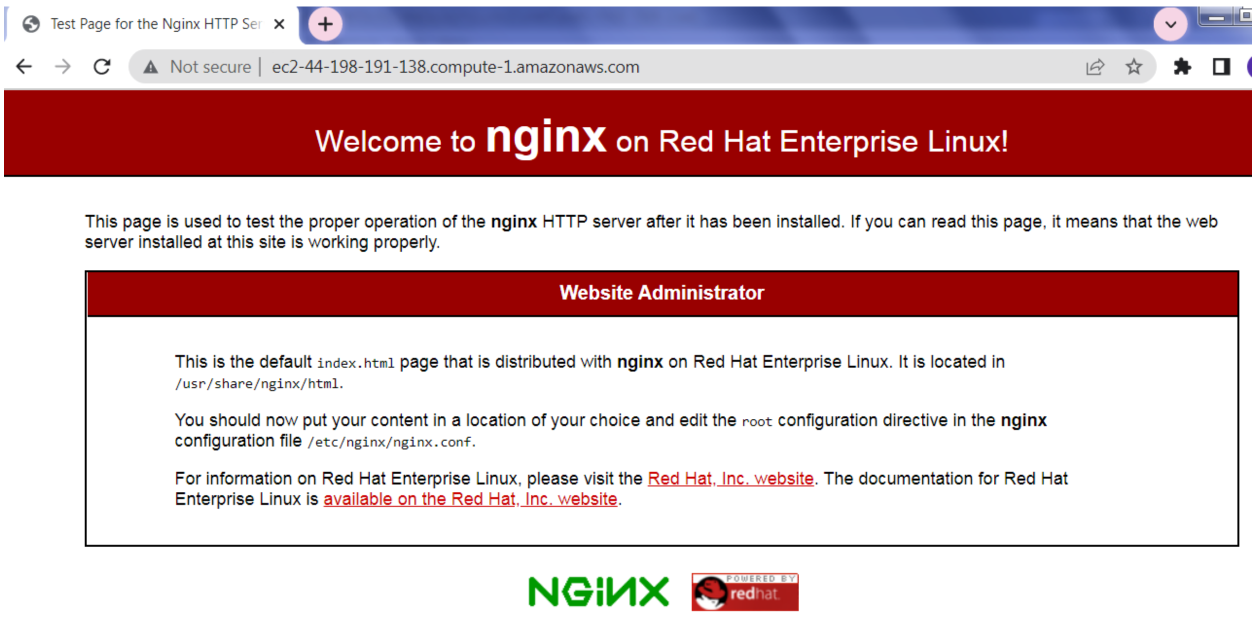


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.


Test Page for the Nginx HTTP Ser
Not secure | ec2-44-198-191-138.compute-1.amazonaws.com

Welcome to **nginx** on Red Hat Enterprise Linux!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

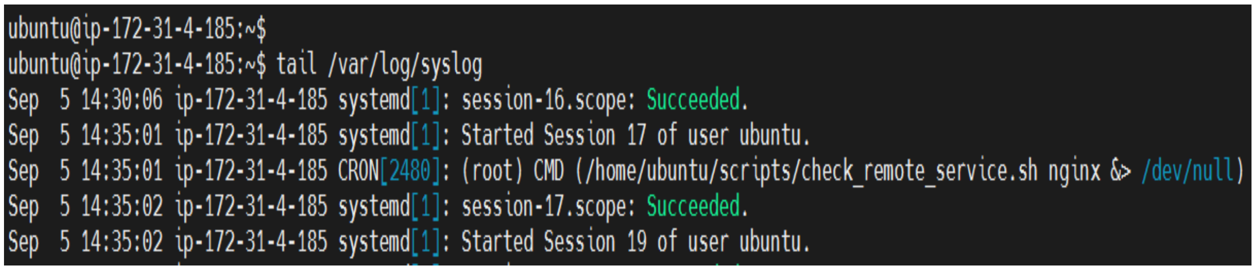
This is the default `index.html` page that is distributed with **nginx** on Red Hat Enterprise Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

For information on Red Hat Enterprise Linux, please visit the Red Hat, Inc. website. The documentation for Red Hat Enterprise Linux is available on the Red Hat, Inc. website.

I can check the system logs to ensure that the cron job ran on my Ubuntu 20 EC2.

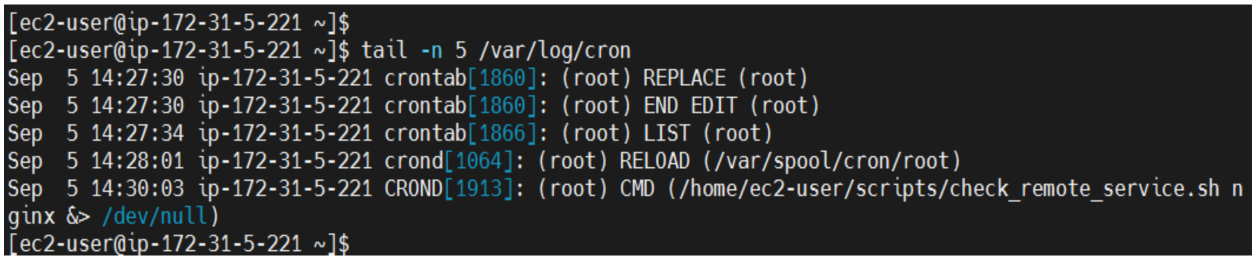
```
$ tail /var/log/syslog
```



```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ tail /var/log/syslog
Sep  5 14:30:06 ip-172-31-4-185 systemd[1]: session-16.scope: Succeeded.
Sep  5 14:35:01 ip-172-31-4-185 systemd[1]: Started Session 17 of user ubuntu.
Sep  5 14:35:01 ip-172-31-4-185 CRON[2480]: (root) CMD (/home/ubuntu/scripts/check_remote_service.sh nginx &> /dev/null)
Sep  5 14:35:02 ip-172-31-4-185 systemd[1]: session-17.scope: Succeeded.
Sep  5 14:35:02 ip-172-31-4-185 systemd[1]: Started Session 19 of user ubuntu.
```

Then, I will check the cron logs to ensure that the cron job ran on my RHEL 8 EC2.

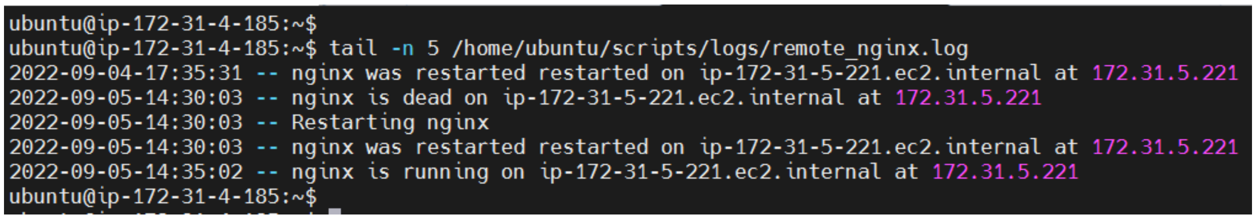
```
$ tail -n 5 /var/log/cron
```



```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ tail -n 5 /var/log/cron
Sep  5 14:27:30 ip-172-31-5-221 crontab[1860]: (root) REPLACE (root)
Sep  5 14:27:30 ip-172-31-5-221 crontab[1860]: (root) END EDIT (root)
Sep  5 14:27:34 ip-172-31-5-221 crontab[1866]: (root) LIST (root)
Sep  5 14:28:01 ip-172-31-5-221 crond[1064]: (root) RELOAD (/var/spool/cron/root)
Sep  5 14:30:03 ip-172-31-5-221 CROND[1913]: (root) CMD (/home/ec2-user/scripts/check_remote_service.sh n
ginx &> /dev/null)
[ec2-user@ip-172-31-5-221 ~]$
```

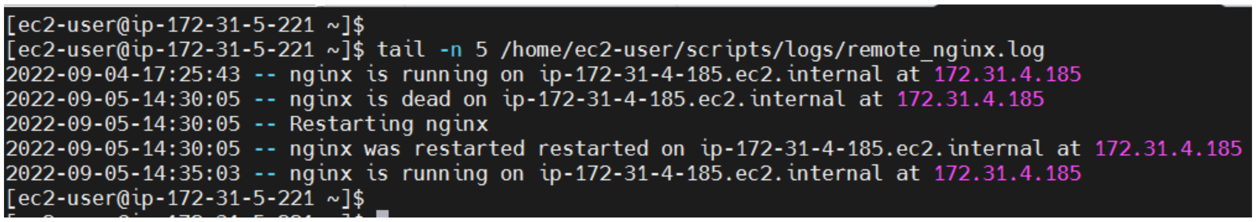
Finally, I can verify the script log files on each EC2 instance.

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



```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ tail -n 5 /home/ubuntu/scripts/logs/remote_nginx.log
2022-09-04-17:35:31 -- nginx was restarted restarted on ip-172-31-5-221.ec2.internal at 172.31.5.221
2022-09-05-14:30:03 -- nginx is dead on ip-172-31-5-221.ec2.internal at 172.31.5.221
2022-09-05-14:30:03 -- Restarting nginx
2022-09-05-14:30:03 -- nginx was restarted restarted on ip-172-31-5-221.ec2.internal at 172.31.5.221
2022-09-05-14:35:02 -- nginx is running on ip-172-31-5-221.ec2.internal at 172.31.5.221
ubuntu@ip-172-31-4-185:~$
```

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



```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ tail -n 5 /home/ec2-user/scripts/logs/remote_nginx.log
2022-09-04-17:25:43 -- nginx is running on ip-172-31-4-185.ec2.internal at 172.31.4.185
2022-09-05-14:30:05 -- nginx is dead on ip-172-31-4-185.ec2.internal at 172.31.4.185
2022-09-05-14:30:05 -- Restarting nginx
2022-09-05-14:30:05 -- nginx was restarted restarted on ip-172-31-4-185.ec2.internal at 172.31.4.185
2022-09-05-14:35:03 -- nginx is running on ip-172-31-4-185.ec2.internal at 172.31.4.185
[ec2-user@ip-172-31-5-221 ~]$
```

The cron job ran successfully on each EC2 instance restarting the remote NGINX service.

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