Cimitra Linux Agent Install

Cimitra Agent Setup

A Cimitra Agent must be installed on any machine that you want to run scripts or other commands on. So for example, if you have a Linux server that you have some BASH and Python scripts on that you want to run from the Cimitra Client, you need to define and deploy a Cimitra Agent specific to that Linux server. This document explains how to do so from Cimitra Administration.

1.	**Login As An Admin User** **NOTE:** First consider making a user other than the Admin user in the Cimitra System. You can give this other user Admin level rights also. To create a user select **Admin | Users**

NOTE: When you log into the Cimitra Client, do not access it from a web browser on the Cimitra Server using an address such as https://locahost. Make sure you are using an IP address that is available to the rest of your network, or better yet a DNS address you have assigned to the Cimitra Server. The address you use to access the Cimitra Server from a web browser; is embedded into the Cimitra Agent when you download it to be used on a particular computer where the Cimitra Agent will run.

2.

Create Agent

1. In the Cimitra Client select **Admin | Agents** on the top menu bar.
2. Then select the **Add New Agent** button
3. Give the Cimitra Agent a descriptive name that works for you. For example: **LINUX_SERVER_ONE**
4. Select the platform: **Linux**
5. Select the **Create** button

3.

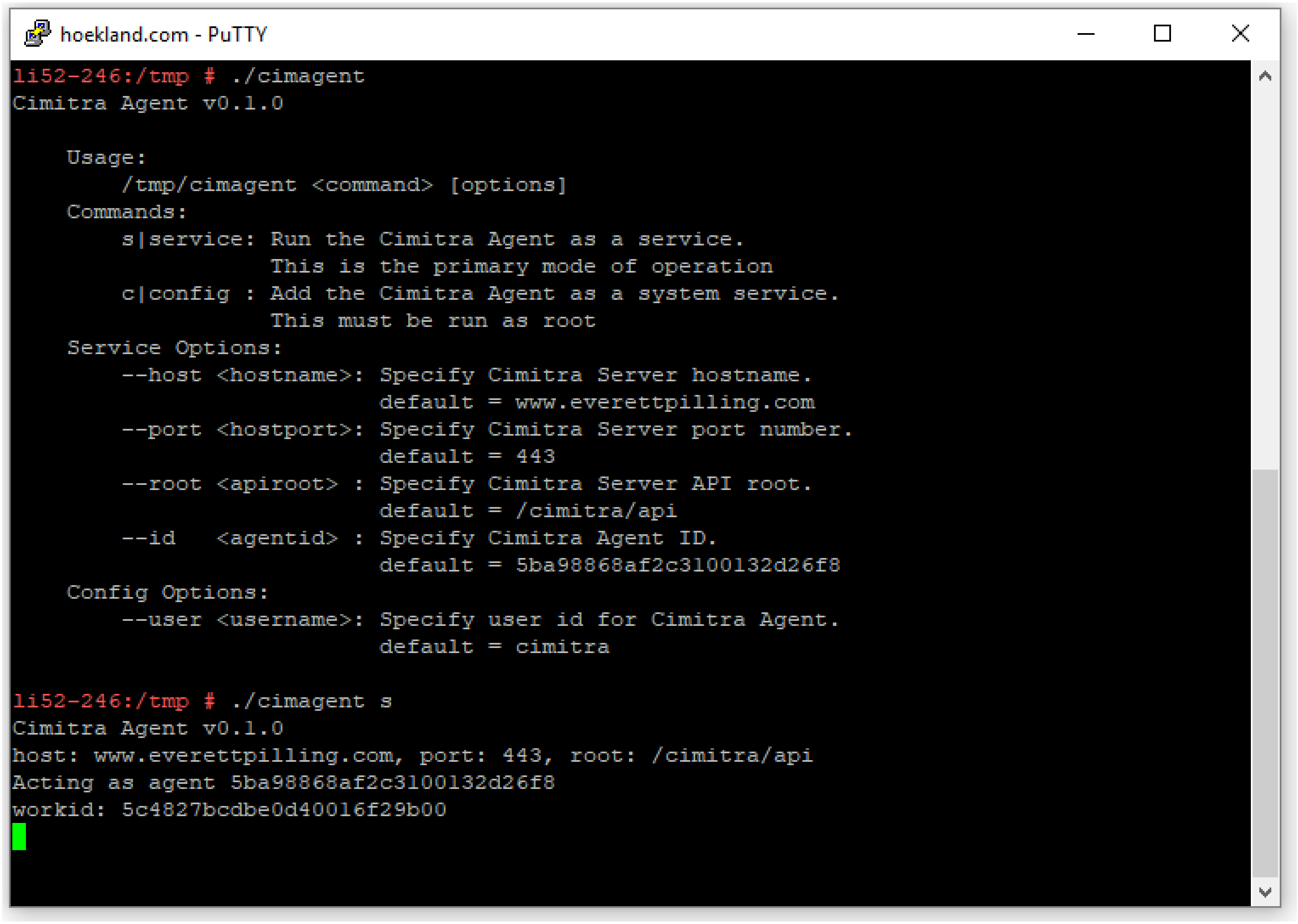
Download Agent

1. Click on the Cimitra Agent object that you just created.
2. Click the **64-bit or 32-bit button** to download the correct Cimitra Agent architecture for your Linux box.

NOTE: The download may take a minute or so, since an executable file is being generated which contains the Cimitra Agent along with

	configuration settings. 3. Create a directory on your Linux box, such as **/cimitra** 4. Place the **cimagent** that was downloaded into the **/cimitra** directory.

4.	**Install Agent**
	
	The Cimitra Agent on Linux is very easy to run. First you will see how to run it in the foreground, which is very helpful for troubleshooting purposes. And then you will see how to configure the Cimitra Agent as a Linux Service.
	
	From a terminal prompt, run the Cimitra Agent in this manner:
	1. First, make sure to make the Cimitra Agent software **executable**.
	
	`chmod +x ./cimagent`
	
	2. Then run the Cimitra Agent with the “s” service command
	
	`./cimagent s`



```
li52-246:/tmp # ./cimagent
Cimitra Agent v0.1.0

    Usage:
        /tmp/cimagent <command> [options]
    Commands:
        s|service: Run the Cimitra Agent as a service.
                   This is the primary mode of operation
        c|config : Add the Cimitra Agent as a system service.
                   This must be run as root
    Service Options:
        --host <hostname>: Specify Cimitra Server hostname.
                           default = www.everettpilling.com
        --port <hostport>: Specify Cimitra Server port number.
                           default = 443
        --root <apiroot> : Specify Cimitra Server API root.
                           default = /cimitra/api
        --id   <agentid> : Specify Cimitra Agent ID.
                           default = 5ba98868af2c3100132d26f8
    Config Options:
        --user <username>: Specify user id for Cimitra Agent.
                           default = cimitra

li52-246:/tmp # ./cimagent s
Cimitra Agent v0.1.0
host: www.everettpilling.com, port: 443, root: /cimitra/api
Acting as agent 5ba98868af2c3100132d26f8
workid: 5c4827bcdbe0d40016f29b00
```

To stop the Cimitra Agent you can select **Ctrl-C**

3. Now that you have proven that the Cimitra Agent is running, you should set it up as a Linux Service so that the Cimitra Agent can run in the background and so that it restarts when the Linux box reboots. You might have your own procedure or utility for making a Linux Service. If so you can use that method. However, the Cimitra Agent makes it really easy to configure. Just use the "**Cimitra Agent Configure**" command:

```
./cimagent c
```

NOTE: The Cimitra Agent Configure feature attempts to register the Cimitra Agent as a Linux Service via the **"systemd" Linux Initialization Service.**

This method of registering a Linux service is likely only supported on the most modern releases of Linux operating systems.

If you get an error when running the `./cimagent c` command read the alternative configuration instructions for assuring that the Cimitra Agent will run on reboot of the Linux box, by reading this document... **CLICK HERE**

4. To authorize the Cimitra Agent to be registered to restart when/if the Linux server restarts type in this command:

```
systemctl enable cimitra-agent
```

5. To start the Cimtra Agent for the first time, type in the command:

```
cimitra start
```

This invokes a BASH shell script that the Cimitra Agent created when you ran the cimagent /c command.

5.

Troubleshooting

The Cimitra Agent can be started and stopped with the commands:

```
cimitra start
cimitra stop
cimitra restart
cimitra status
```

If ever you need to troubleshoot the Cimitra Agent, then **stop the Cimitra Agent** and then find the Cimitra Agent binary file, which is likely in the directory…

```
 /usr/bin
```

So the command would be:

```
/usr/bin/cimitra s
```

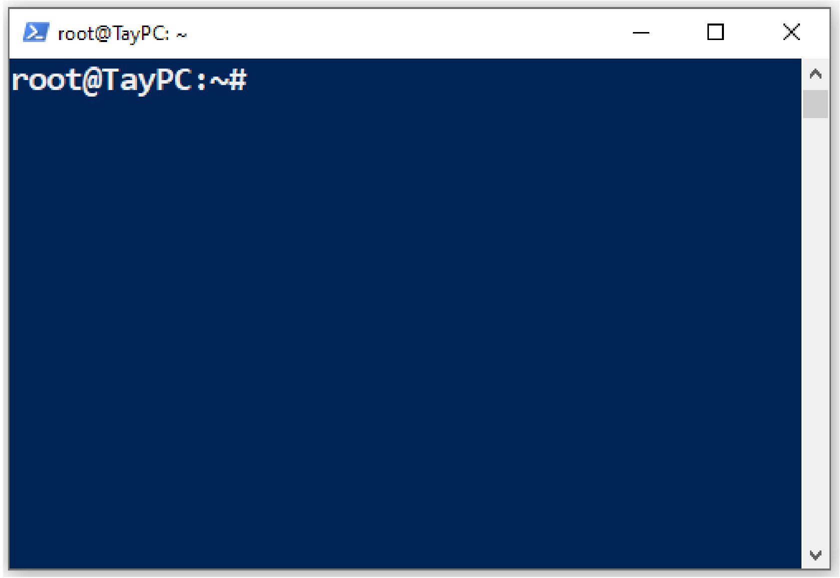
Cimitra generates Linux binary executables for Linux variants that are X86 processor-based. So if you have a version of Linux that runs on an ARM processor, Cimitra cannot generate a Cimitra Agent binary executable file for that platform. For this reason, Cimitra has created a **Node.js** based Cimitra Agent for deployment on any platform that you can get **Node.js** on.

So for example, if you have a Raspberry Pi, Cimitra can support that platform through its Node.js based Agent. The Node.js client is what is made for download when you create a Cimitra Agent with the **Platform of "Other"**.

So basically, if you can get Node.js on a machine, the Cimitra Agent can run on that machine.

Running “Agentless” Through SSH Key Exchange

You can install a Cimitra Agent in one location, and then through SSH key exchange, you can allow a Cimitra Agent to run remote commands to another Linux box via ssh.

Linux Box 1	**Linux Box 2**
Cimitra Agent is **already deployed** to this box	Cimitra Agent will **not be deployed** to this box
IP Address: 192.168.99.**1**	192.168.99.**2**
Generates SSH Public **Key**	**Receives** Linux Box 1’s SSH **Key** into it’s **authorized keys**
 root@TayPC: ~ `root@TayPC:~#`	hoekland.com - PuTTY `li52-246:~ #`

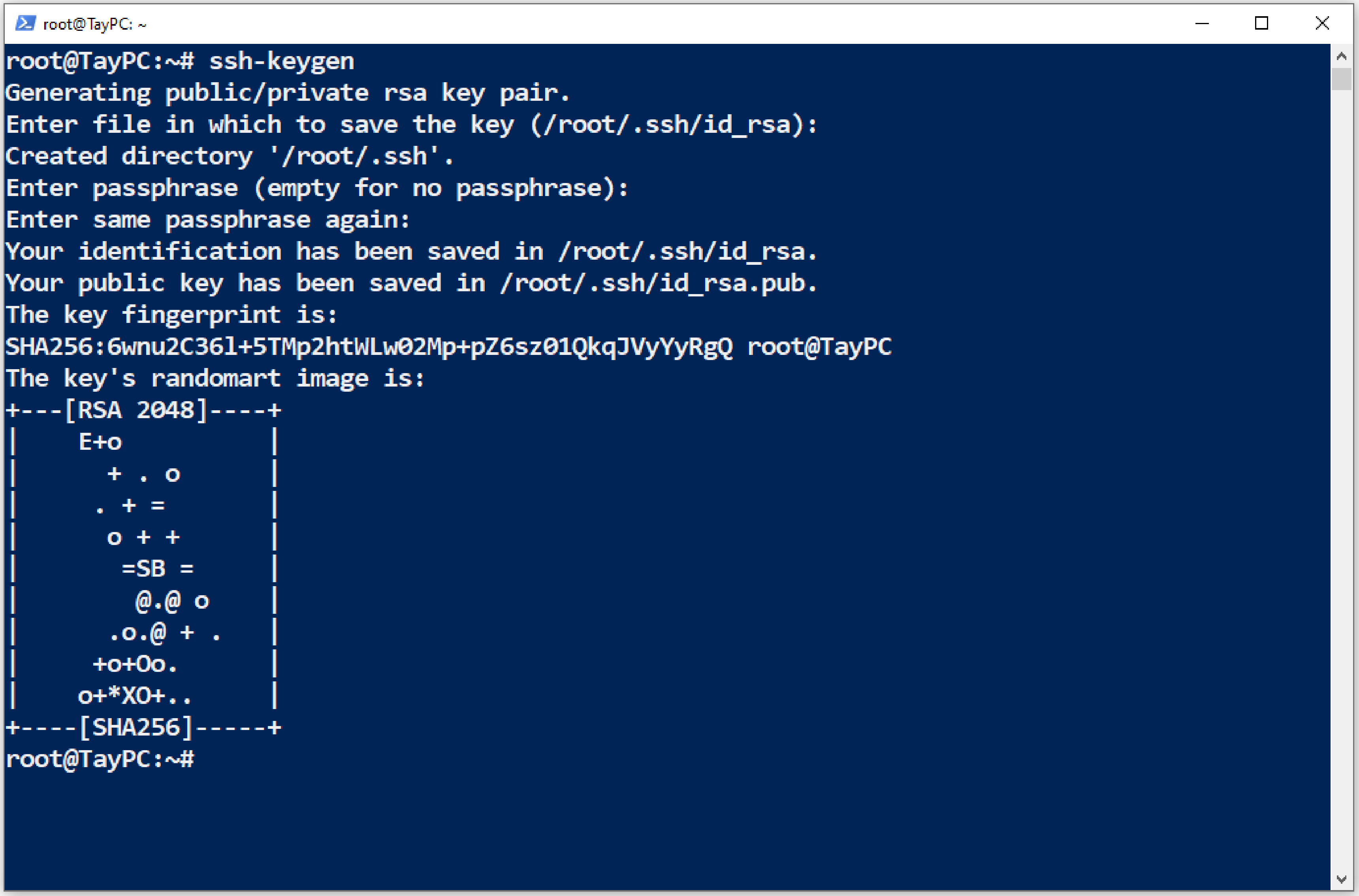
On **Linux Box 1** - Where the Cimitra Agent is Deployed

1. Login is as the root user

2. Generate an SSH Public Key

```
ssh-keygen
```

NOTE: Follow the prompts, but if a key already exists you will be prompted to replace it, generally you **DO NOT WANT TO REPLACE** the SSH-Key. You can just press the **[Enter**] key if you do not want to use a passphrase. See the screenshot below.



root@TayPC: ~

```
root@TayPC:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6wnu2C36l+5TMp2htWLw02Mp+pZ6sz01QkqJVyYyRgQ root@TayPC
The key's randomart image is:
+---[RSA 2048]----+
|    E+o          |
|      + . o      |
|     . + =       |
|      o + +      |
|       =SB =     |
|        @.@ o    |
|      .o.@ + .   |
|     +o+Oo.      |
|    o+*XO+..     |
+----[SHA256]-----+
root@TayPC:~#
```

3. The contents of the `id_rsa.pub` file **on Linux Box 1** need to get appended to a file called `authorized_keys` file on **Linux Box 2**. From **Linux Box 1** issue the following command to append the contents of the **id_rsa.pub** file to the **authorized_keys** file on Linux Box 2.

```
cat /root/.ssh/id_rsa.pub | ssh root@192.168.99.2 "cat >>
/root/.ssh/authorized_keys"
```

   a. When prompted, enter the root password for **Linux Box 2**.

NOTE: You can use some other user other than the **root** user. So for example, if you had a user with the id of **tkratzer** on both Linux boxes, you could use a command such as:

```
cat /home/tkratzer/.ssh/id_rsa.pub | ssh tkratzer@192.168.99.2 "cat >>
/home/tkratzer/.ssh/authorized_keys"
```

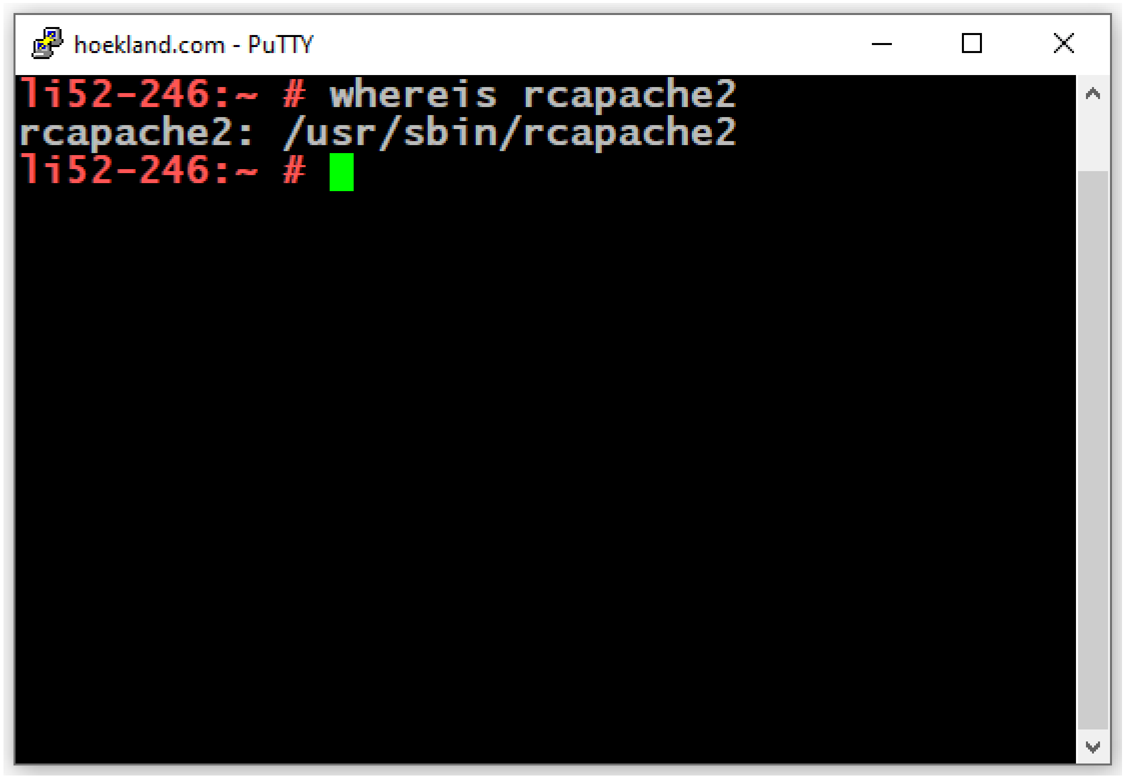
4. From **Linux Box 1**, issue a command on **Linux Box 2** using the following syntax:

```
ssh root@192.168.99.2 ls /tmp
```

Or

```
ssh tkratzer@192.168.99.2 ls /tmp
```

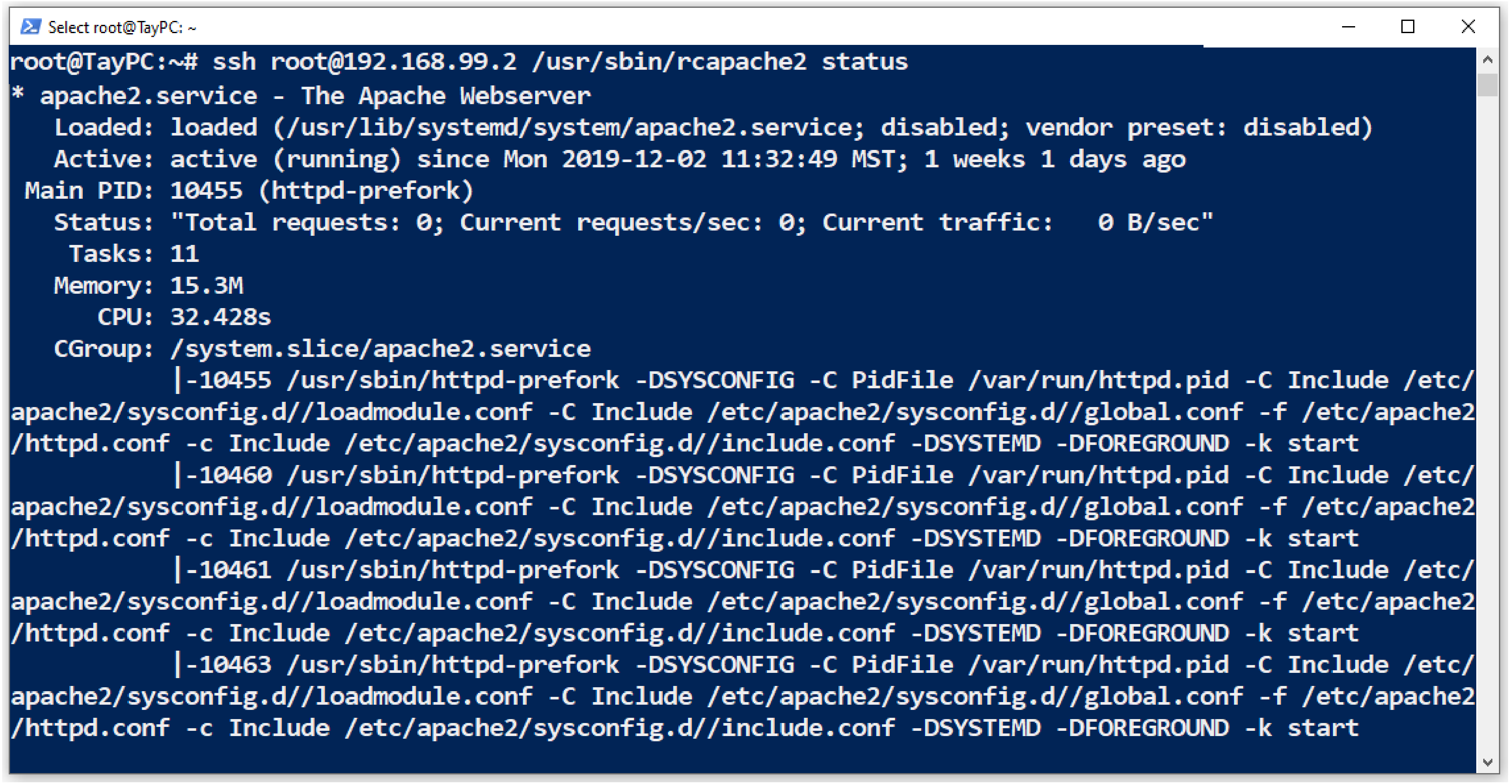
5. If you need to call a command that is in a search path on Linux Box 2, you have to indicate the entire path to the command. So for example, to get the status of the Apache server, we first need to discover where the **rcapache2** command exists on Linux Box 2. You use the **whereis** command to determine the location of a command:



```
li52-246:~ # whereis rcapache2
rcapache2: /usr/sbin/rcapache2
li52-246:~ #
```

So the command you would issue from **Linux Box 1** to execute a command on **Linux Box 2** would be as follows:

```
ssh root@192.168.99.2 /usr/sbin/rcapache2
```


Select root@TayPC: ~

```
root@TayPC:~# ssh root@192.168.99.2 /usr/sbin/rcapache2 status
* apache2.service - The Apache Webserver
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2019-12-02 11:32:49 MST; 1 weeks 1 days ago
 Main PID: 10455 (httpd-prefork)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
    Tasks: 11
   Memory: 15.3M
      CPU: 32.428s
   CGroup: /system.slice/apache2.service
           |-10455 /usr/sbin/httpd-prefork -DSYSCONFIG -C PidFile /var/run/httpd.pid -C Include /etc/
apache2/sysconfig.d//loadmodule.conf -C Include /etc/apache2/sysconfig.d//global.conf -f /etc/apache2
/httpd.conf -c Include /etc/apache2/sysconfig.d//include.conf -DSYSTEMD -DFOREGROUND -k start
           |-10460 /usr/sbin/httpd-prefork -DSYSCONFIG -C PidFile /var/run/httpd.pid -C Include /etc/
apache2/sysconfig.d//loadmodule.conf -C Include /etc/apache2/sysconfig.d//global.conf -f /etc/apache2
/httpd.conf -c Include /etc/apache2/sysconfig.d//include.conf -DSYSTEMD -DFOREGROUND -k start
           |-10461 /usr/sbin/httpd-prefork -DSYSCONFIG -C PidFile /var/run/httpd.pid -C Include /etc/
apache2/sysconfig.d//loadmodule.conf -C Include /etc/apache2/sysconfig.d//global.conf -f /etc/apache2
/httpd.conf -c Include /etc/apache2/sysconfig.d//include.conf -DSYSTEMD -DFOREGROUND -k start
           |-10463 /usr/sbin/httpd-prefork -DSYSCONFIG -C PidFile /var/run/httpd.pid -C Include /etc/
apache2/sysconfig.d//loadmodule.conf -C Include /etc/apache2/sysconfig.d//global.conf -f /etc/apache2
/httpd.conf -c Include /etc/apache2/sysconfig.d//include.conf -DSYSTEMD -DFOREGROUND -k start
```