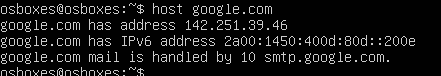
**Host**

**Host 8.8.8.8**

****

**host google.com**

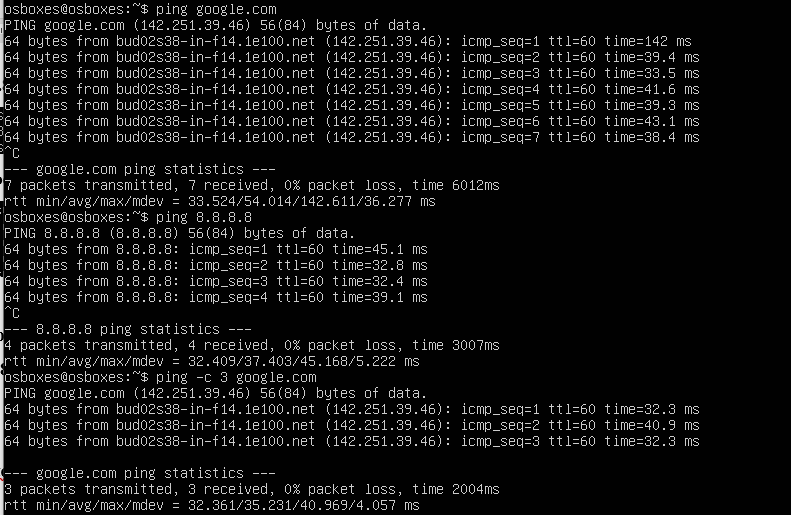
****

**Ping**

**ping google.com**

**ping 8.8.8.8**

**ping -c 3 google.com**

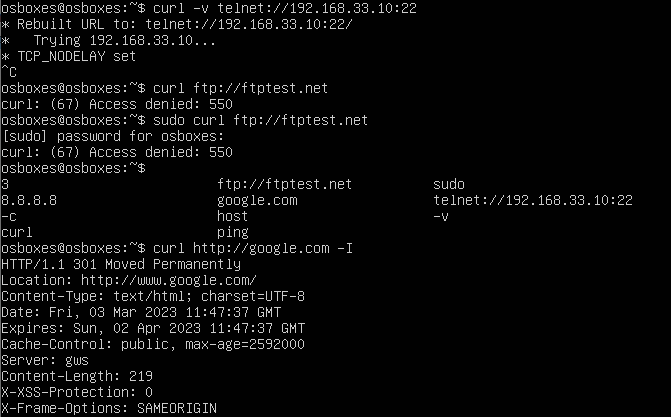
****

**Curl**

**curl -v** [**telnet://192.168.33.10:22**](telnet://192.168.33.10:22)

**curl** [**ftp://ftptest.net**](ftp://ftptest.net)

**curl http://google.com -I**

****

**nc (netcat)**

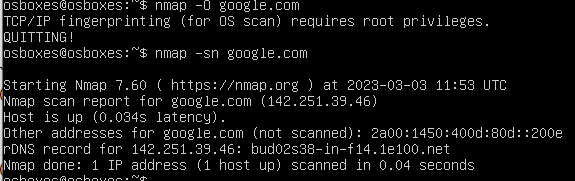
**nc -v -n 192.168.33.10 22**

****

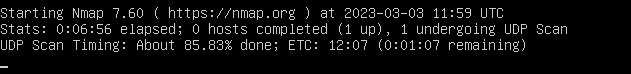
**Nmap**

**sudo nmap -sn <Your-IP>**

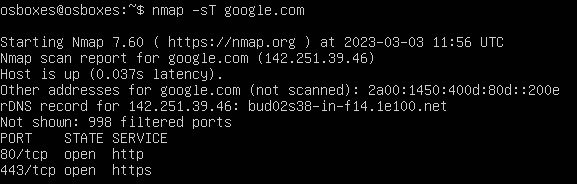
**sudo nmap -O <Your-IP>**

****

**sudo nmap -sS -sU -PN <Your-IP>**

****

**sudo nmap -sT <Your-IP>**

****

**sudo nmap -A -T4 <Your-IP>**

****

**telnet**

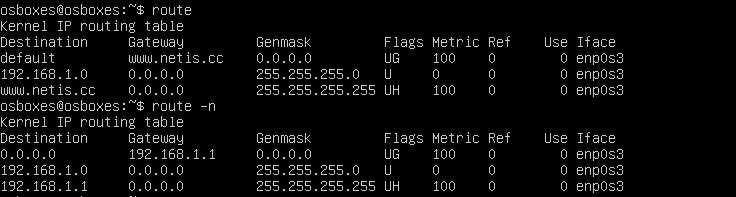
**telnet 142.251.39.14**

****

**route**

**route**

**route -n**

****

**wget**

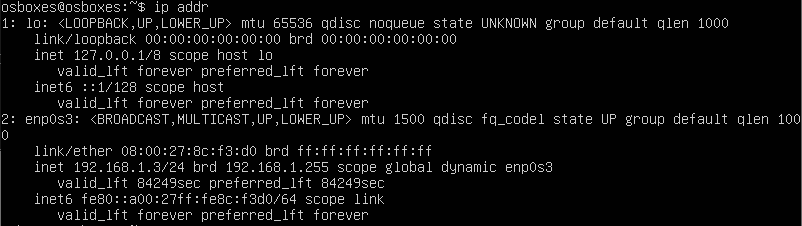
**wget -e use\_proxy=yes http\_proxy=<proxy\_host:port>** [**http://externalsite.com**](http://externalsite.com)

**wget** [**www.google.com**](http://www.google.com)

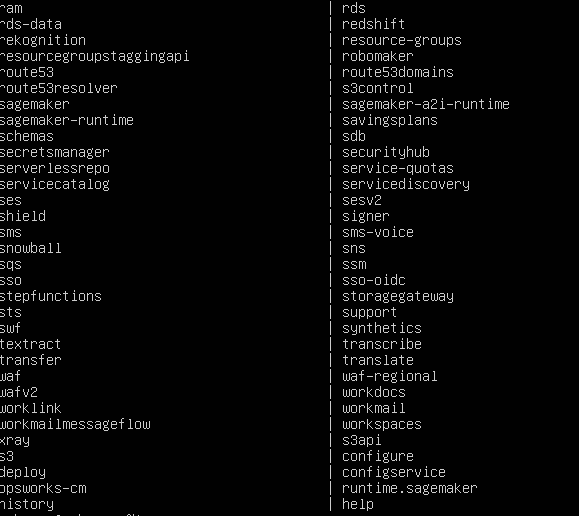
****

**ip (ifconfig)**

**ip addr**

****

**ip a | grep eth0 | grep "inet" | awk -F" " '{print $2}'**

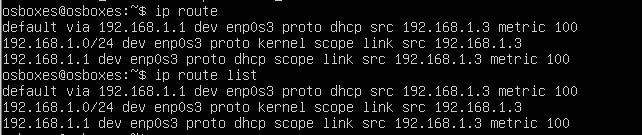
****

**ip a show enp0s3**

****

**ip route**

**ip route list**

****

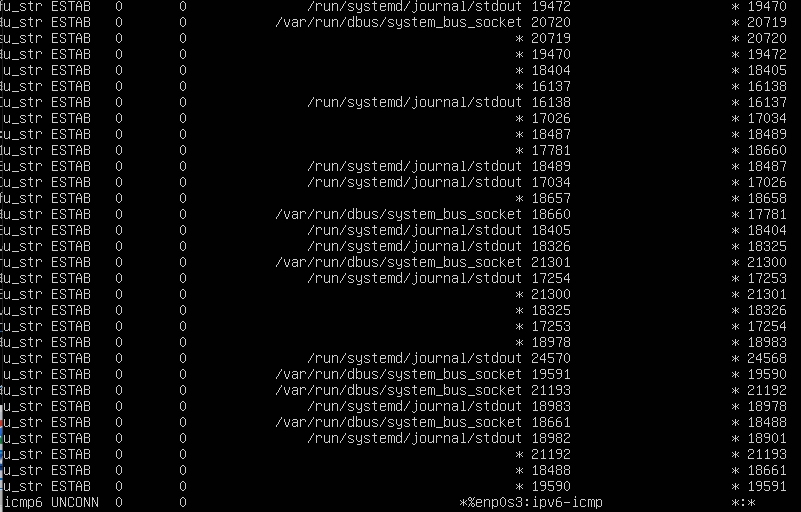
**arp**

**arp**

****

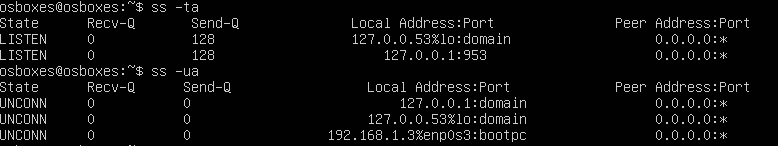
**ss (netstat)**

**ss**

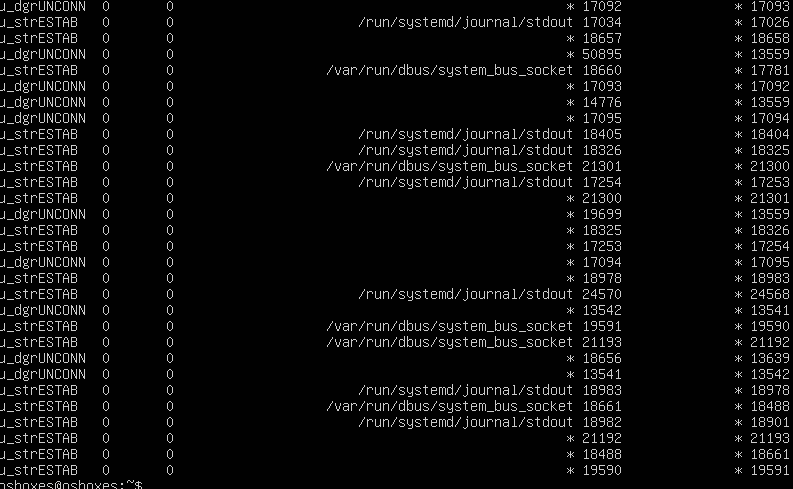
****

**ss -ta**

**ss -ua**

****

**ss -xa**

****

**ss -lt**

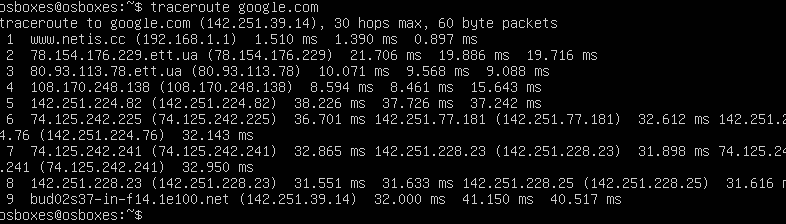
**ss -t -r state established**

**ss -t -r state listening**

****

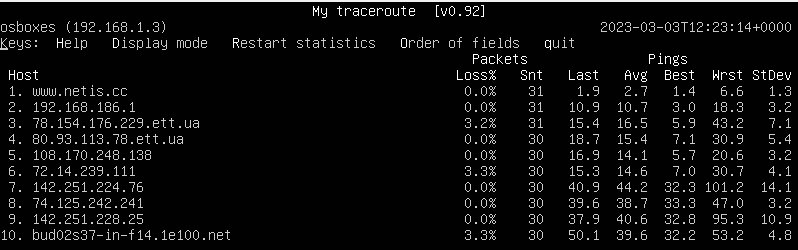
**traceroute**

**traceroute google.com**

****

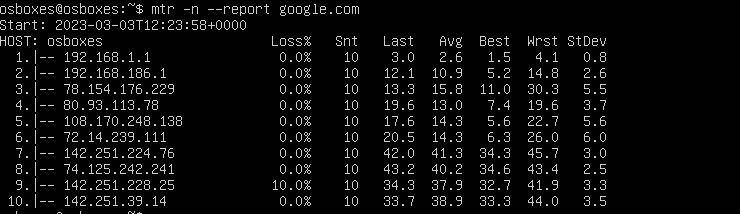
**mtr**

**mtr google.com**

****

**mtr report**

**mtr -n --report google.com**

****

**dig**

**dig twiter.com ANY**

**dig google.com ANY +short**

**dig www.google.com A +short**

**dig google.com CNAME +short**

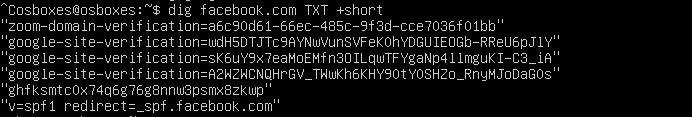
**dig google.com MX +short**

****

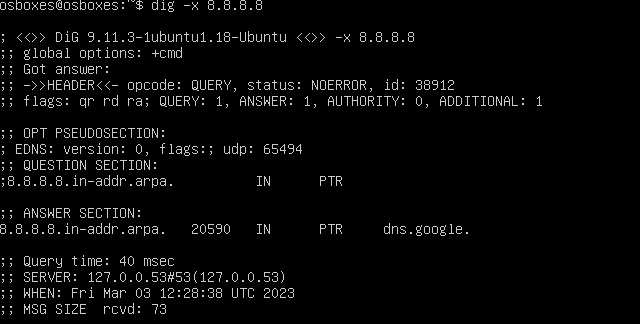
**dig google.com NS +short**

****

**dig facebook.com TXT +short**

****

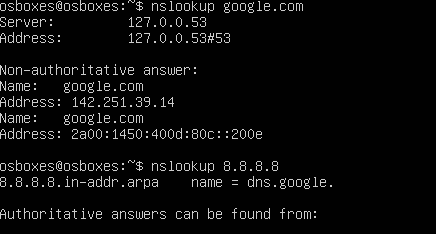
**dig -x 8.8.8.8**

****

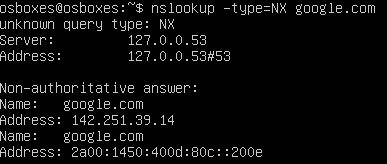
**nslookup**

**nslookup google.com**

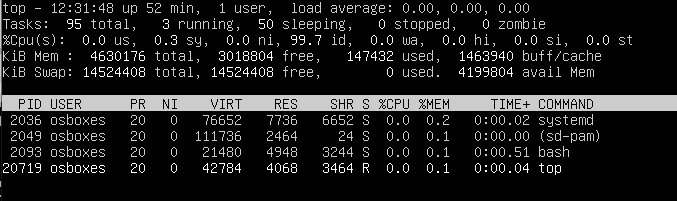
**nslookup 8.8.8.8**

****

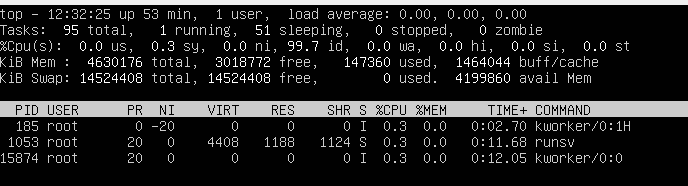
**nslookup -type=any google.com**

****

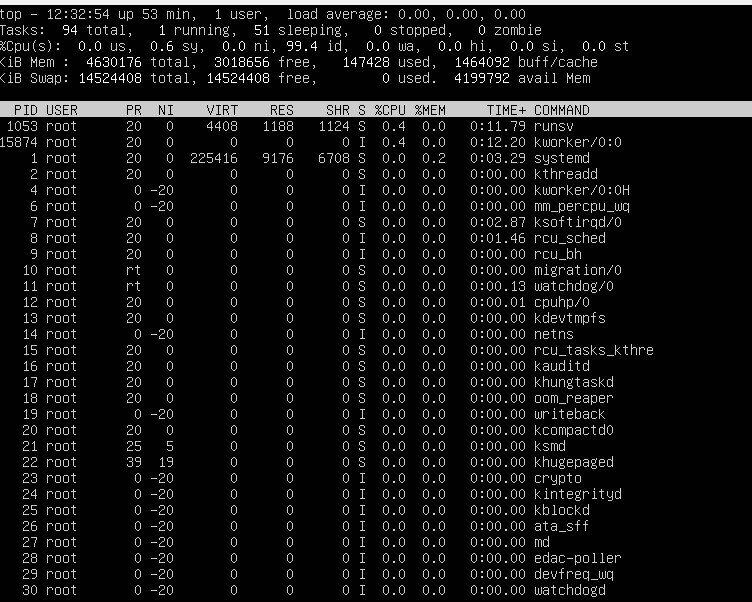
**Top**

**top -u 'username'**

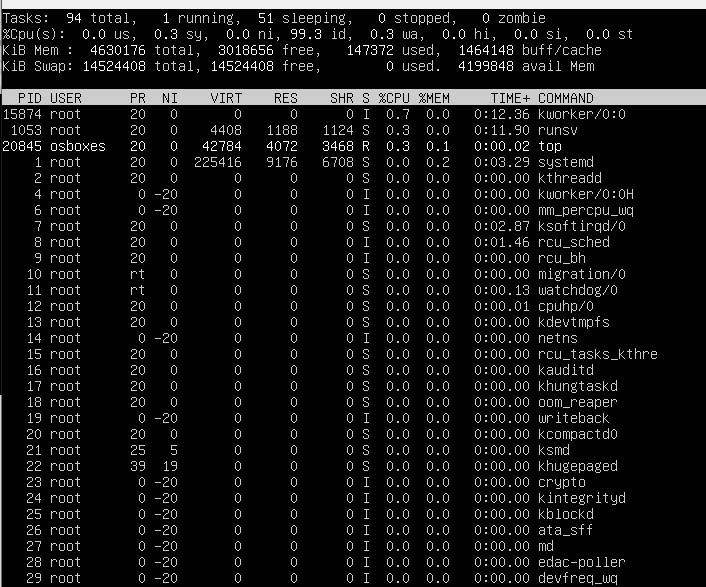
**top -i**

****

**top -d 5**

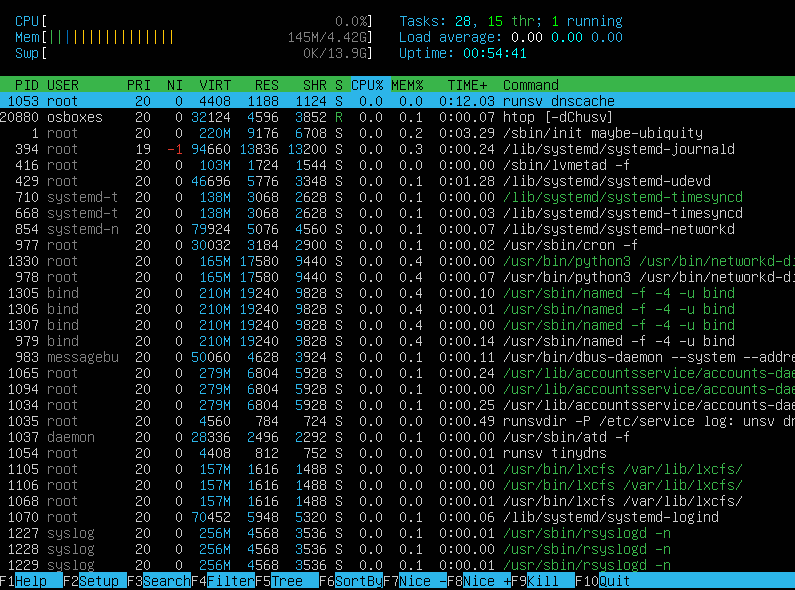
****

**top -n 2**

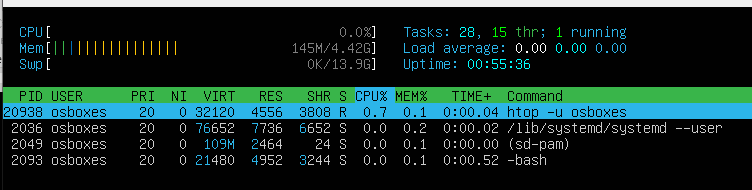
****

**Htop**

**htop [-dChusv]**

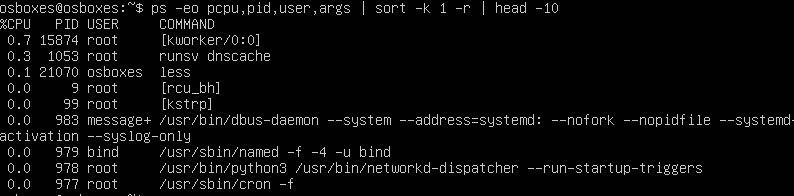
****

**htop -u username**

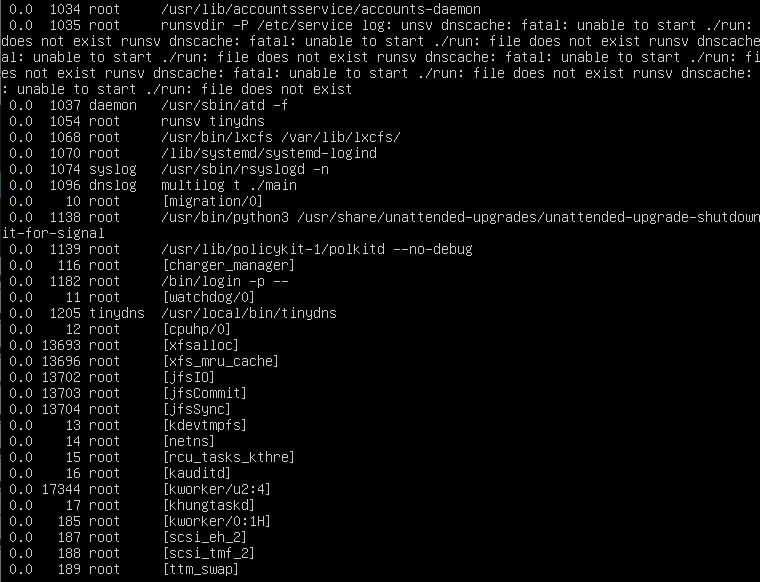
****

**Ps**

**ps -eo pcpu,pid,user,args | sort -k 1 -r | head -10**

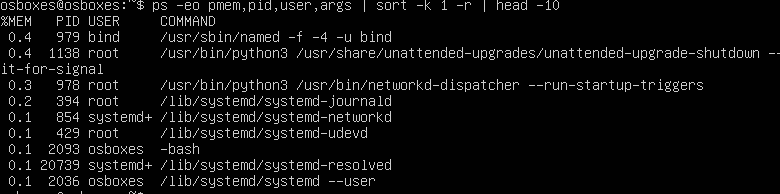
****

**ps -eo pcpu,pid,user,args | sort -r -k1 | less**

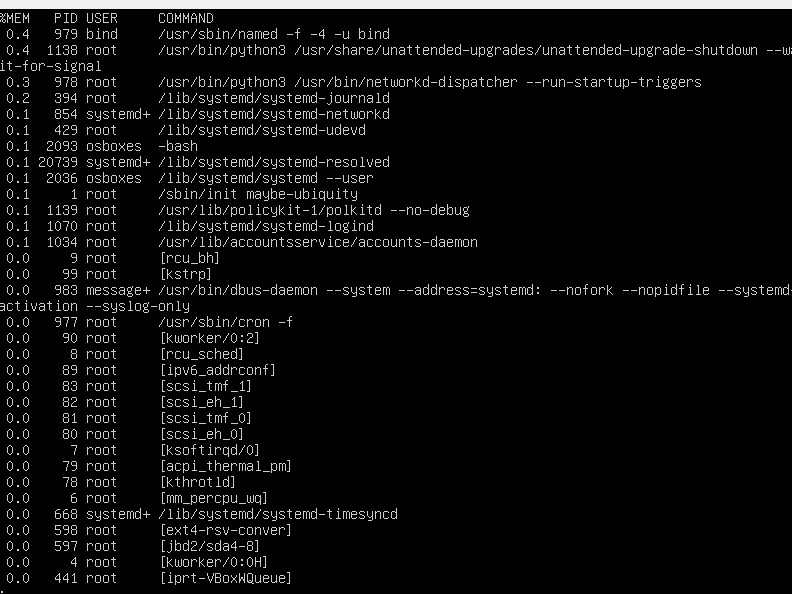
****

**or**

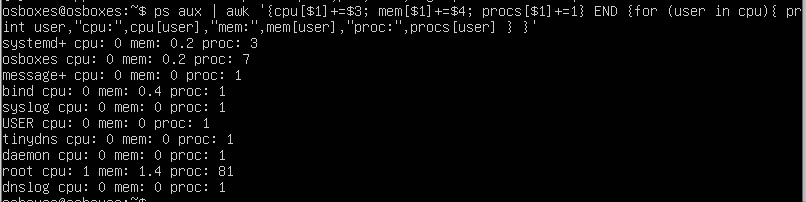
**ps -eo pmem,pid,user,args | sort -k 1 -r | head -10**

****

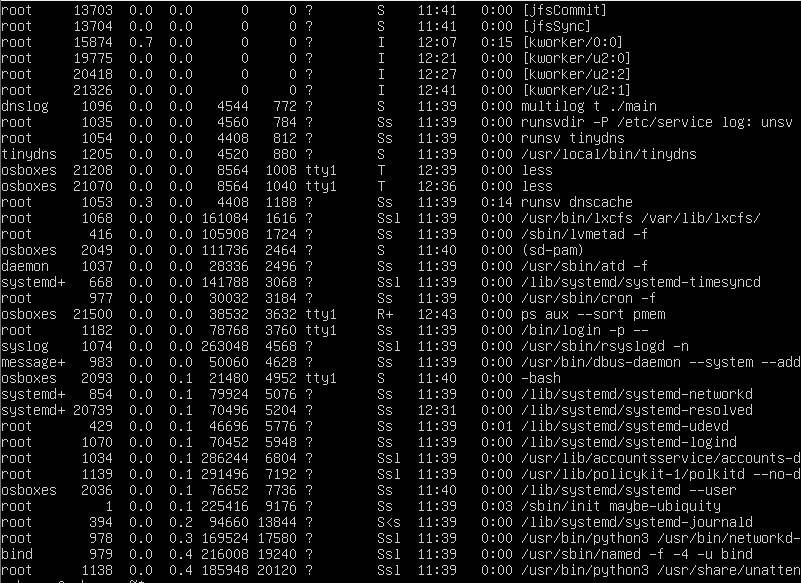
**ps -eo pmem,pid,user,args | sort -r -k1 | less**

****

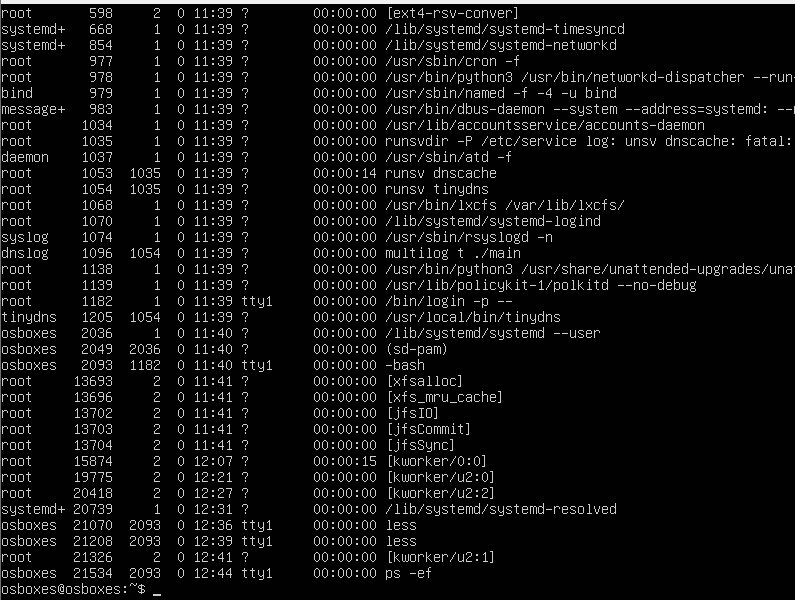
**ps aux | awk '{cpu[$1]+=$3; mem[$1]+=$4; procs[$1]+=1} END { for (user in cpu){ print user,"cpu:",cpu[user],"mem:",mem[user],"proc:",procs[user] } }'**

****

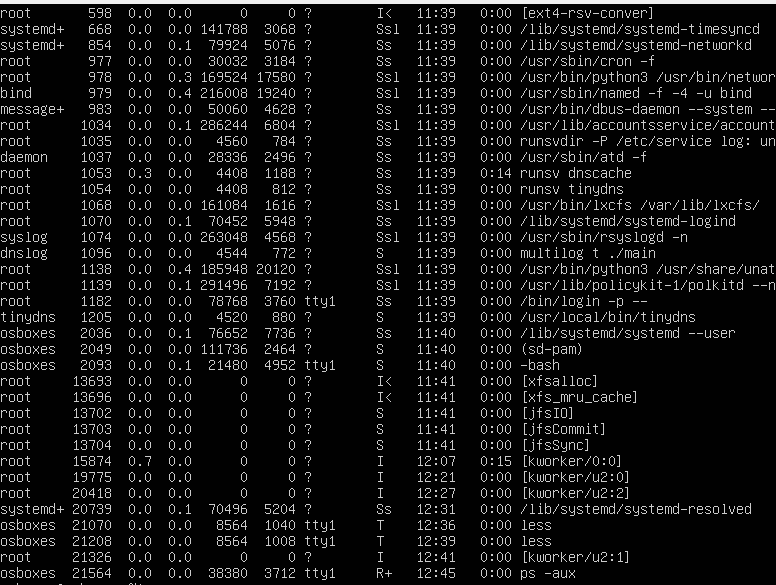
**ps aux --sort pmem**

****

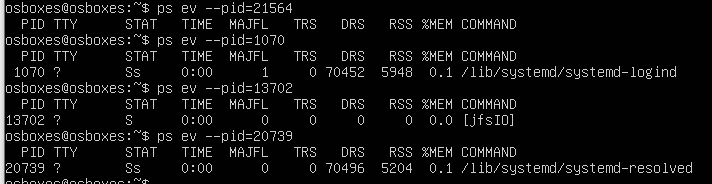
**ps -ef**

****

**ps -aux**

****

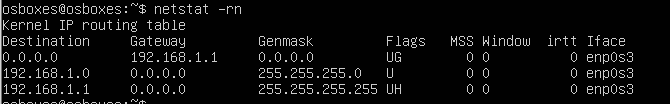
**ps ev --pid=[HighestEnterPID]**

****

**Netstat**

Using the netstat command with the -r option lists the kernel routing information in the same way as with the route command.

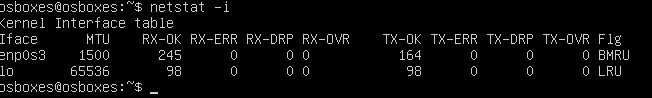
**netstat -rn**

****

Note that the additional -n option is used to disable hostname lookup. It configures the netstat command to display addresses as dot-separated quad IP numbers instead of host and network names in the form of symbols.

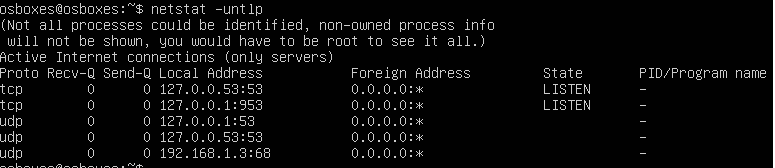
The -i option configures the netstat command to display network interface statistics. By including the -a option, we’ll include all interfaces present on the kernel in the output, not just those currently configured.

**Netstat -i**

****

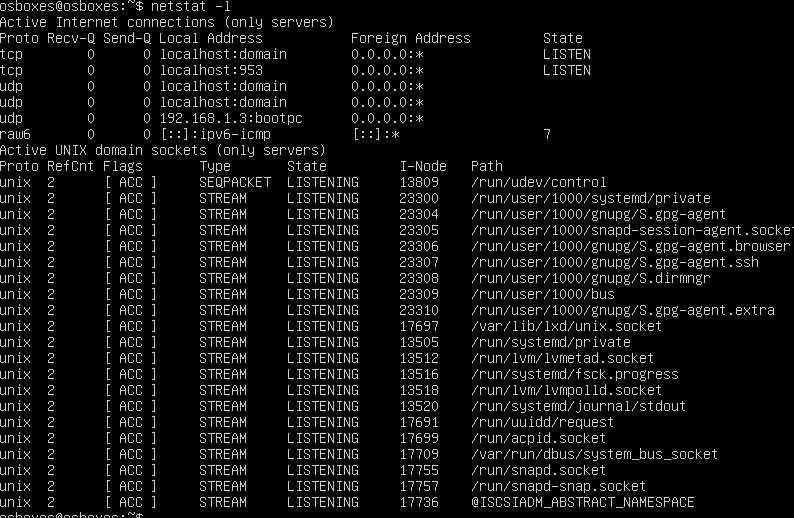
This command displays anything listening for incoming traffic and the port it is listening on. Breaking this command down, the first parameter, -t, identifies your request for information pertaining to TCP. Next, -u represents UDP; -l requests listening sockets; -p attempts to show the name of the program; and -n shows numeric values. Putting it all together, you get netstat -unltp:

**netstat -unltp**

****

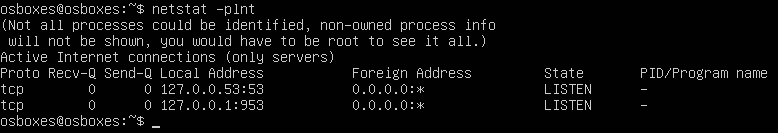
Use the netstat command and the -l option to listen only to active ports,

**Netstat -l**

****

You can also list the ports that are being listened to. This works for root users on a Linux machine:

**Netstat -plnt**

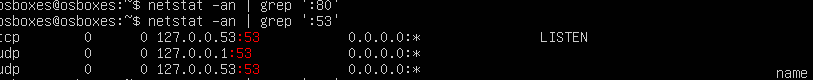
****

**Troubleshooting with netstat**

Troubleshooting with netstat involves identifying, diagnosing, and solving network problems by adding options to the command. In addition to the -untlp option above, you can use the grep option for troubleshooting issues.

**Using netstat + grep**

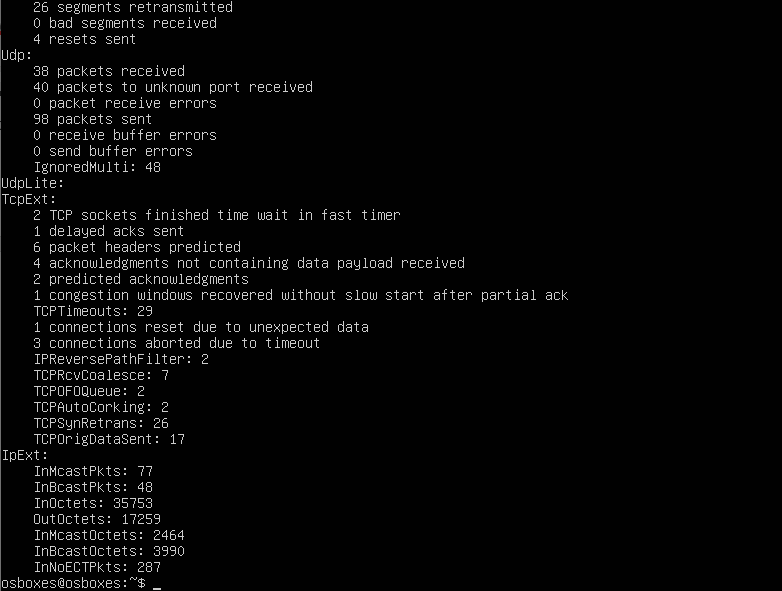
To see what process is occupying a specific port, you can use the grep option.



**Listing raw network statistics only**

Network statistics are displayed using the -s option with the netstat command:

**netstat -s**

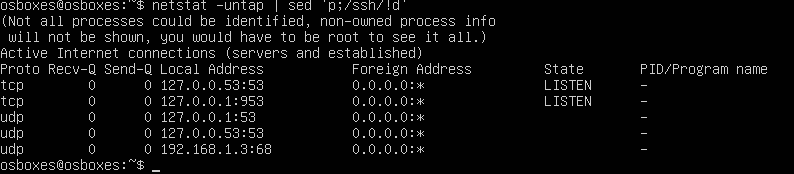


**Monitoring logins in an SSH Server**

Let’s say you’re running a public server such as an SSH web server. The SSH server will open a port in the server system for users to access and log in.

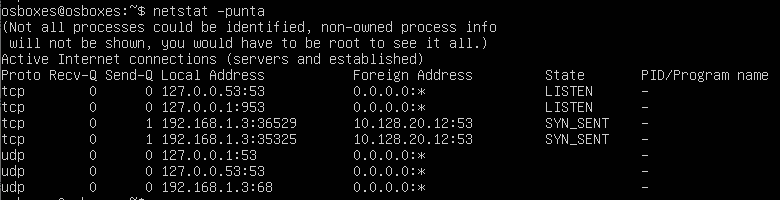
The default port for sshd is TCP port 22. With the netstat command, you can monitor all open ports, using the options together with the command, as shown below:

**netstat -untap | sed ‘p;ssh/!d’**



The netstat command can also be used to monitor your internet browsing session:

**netstat -punta**

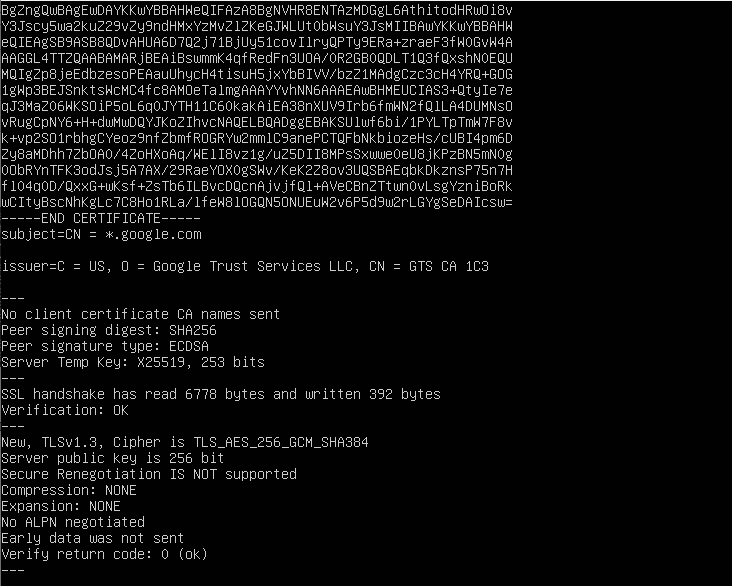


**Openssl s\_client**

Troubleshooting HTTP connections

Web server testing is a very common troubleshooting scenario. With openssl, you can open a secure connection to a remote server on port 443, and then send raw HTTP commands. For example, the following text shows an exchange between an openssl client and a remote web server. Text in red represents commands typed by the user**:**

**openssl s\_client -connect google.com:443**

****

**Troubleshooting IMAP and POP connections**

Similar to the procedure for web server troubleshooting, you can test secure POP (port 995) and IMAP (port 993) connectivity.

The following text shows a sample exchange between an openssl client and a remote IMAP server. Text in red represents commands typed by the user:

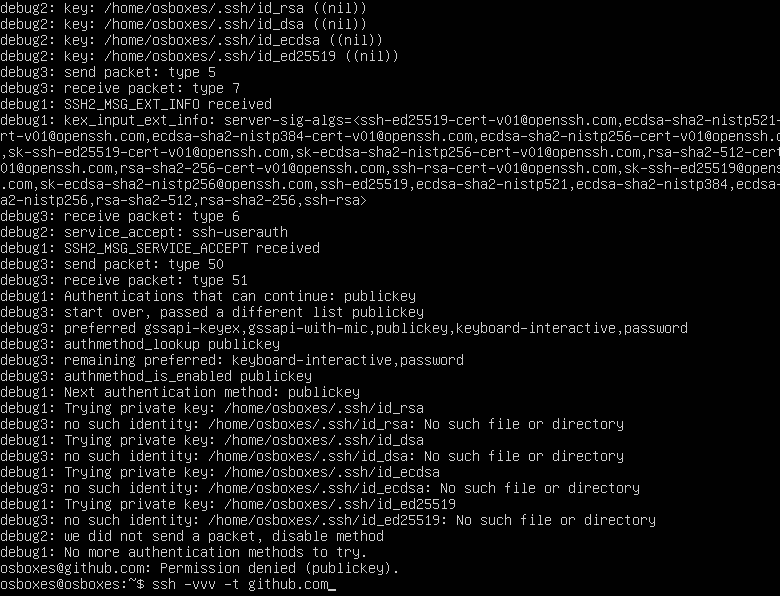
**openssl s\_client -connect google.com:993**

****

**SSH**

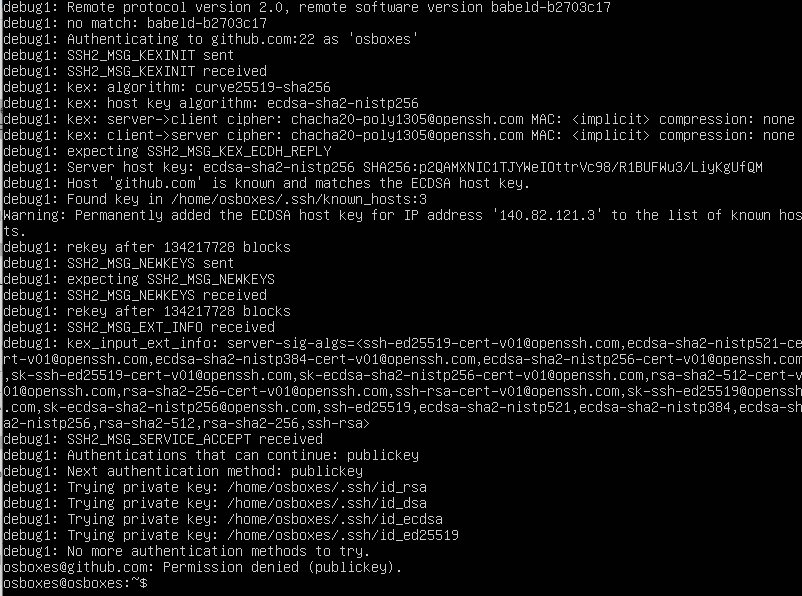
This is the command we will be using to get debug data from your client. The -v is one level of verbose logging. The -T is to avoid an interactive shell. This command can output lots of data.

**ssh -vvv -t github.com**

****

**Analize Debug**

**ssh -v -t github.com**

****

**Rsync**

If you are having difficulty using rsync to transfer files to your CDN storage account, please check the following items:

Verify that the user attempting rsync access has the Rsync privilege.

Verify that a recent version of rsync is installed on your local machine.

Verify that the specified hostname corresponds to the correct storage location (e.g., vny, cpm, or ams).

Verify the command used to transfer your content matches the syntax defined below.

**rsync -avp {Local Path} {User Name}@rsync.yyy.AN.{Base Domain}:{Destination Path}**

Verify that Local Path points to an existing folder on your local computer.

Verify that Destination Path starts with a forward slash.

Example:

**/DestinationFolder**

If you have trouble establishing a connection, try using a backslash to escape the @ symbol in the email address specified as your MCC user name.

**rsync -avp Local Path joe\@cdn.com@rsync.ams.0001.{Base Domain}:Destination Path**

**Theorical understanding of Anycast**

**Anycast is a network addressing and routing method where one IP address can be used jointly by devices in several locations, which is implemented in the IPv6 protocol. The method works like this, it searches for the nearest machine from the location of the user request. This reduces latency and increases redundancy. The use of this technology provides an increase in the level of reliability, DNS security, and fault tolerance.**