How to use ss command on Linux

The ss command is a new alternative to the classic netstat. You can use it on Linux to get data about network connections. Here's how to work with this helpful tool.

Ss and netstat commands

As an alternative to the outdated netstat command, the ss command gives you information about communication between your computer and other computers, networks and other services.

The ss command will display metrics for Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Unix (interprocess) and raw sockets. The raw socket operates at the OSI level network, which means that TCP and UDP protocols must be handled by software, not by the transport layer. Internet Control Message Protocol (ICMP) messages and the ping utility use the raw socket.

Use the ss command

You do not have to install the ss command, it is available on the latest versions of Linux distributions. However, the result of this command may be both long and wide.

A result after using the ss command

List the network connections

Type the following command:

```
ss
```

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process u_str ESTA
```

The columns will appear as follows:

1. Netid: Types of sockets.
2. State: Status of the socket.
3. Recv-Q: Number of packets received.
4. Send-Q: Number of packets sent.
5. Local Address: Port: Local address and port (or equivalent values ??for Unix socket).
6. Peer Address: Port: Remote address and port (or equivalent value for Unix socket).

For UDP sockets, the State column is usually left blank. With a TCP socket it could be one of the following:

1. LISTEN: Only for server side. Socket is waiting for connection request.
2. SYN-SENT: Only for client side. This socket makes a connection request and waits to see when it is accepted.
3. SYN-RECEIVED: Only for server side. This socket waits for connection recognition after the connection request is accepted.
4. ESTABLISHED: For server and client. An active connection is established between the server and the client, allowing data to be transferred between the two parties.
5. FIN-WAIT-1: For server and client. This socket is waiting for a connection request from the remote socket, or recognizes a connection request from the previous one.
6. FIN-WAIT-2: For server and client. This socket is waiting for connection request from the remote socket.
7. CLOSE-WAIT: For Server and client. This socket is waiting for connection requests from local users.
8. CLOSING: This socket is waiting for a connection request to identify it from the remote socket.
9. LAST-ACK: For server and client. This socket is waiting for an identifier requesting the connection it sends to the remote socket.
10. TIME-WAIT: For server and client. This socket sends an identifier to the remote socket to indicate it has received a request from the remote socket. Now it is waiting to make sure this identity has been accepted.
11. CLOSED: There was no connection so the socket was canceled.

List of active sockets

To see active sockets, add -l (listening) to the command:

```
ss -l
```

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process nl UNCONN
```

All of these sockets are disconnected and working. 'rtnl' means routing netlink, used to transfer information between kernel and userspace processes.

List all sockets

To list all sockets, you can use the -a (all) option:

```
ss -a
```

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process nl UNCONN
```

The result will include all sockets regardless of status.

List the TCP sockets

You can use a filter, which only takes sockets to be displayed. Here, we use the -t (TCP) option, only TCP sockets will be listed:

```
ss -a -t
```

List the UDP sockets

The -u option (UDP) is the same filter. Use if you only want to see the UDP socket:

```
ss -a -u
```

```
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process UNCONN 0 0 0.0.0
```

List the Unix sockets

To see Unix sockets, you can add -x (Unix) to the command line as follows:

```
ss -a -x
```

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process u_str ESTA
```

List the raw sockets

To filter raw sockets, add -w (raw):

```
ss -a -w
```

Lists IP socket version 4

Sockets using TCP / IP version 4 can be listed using option -4 (IPV4):

```
ss -a -4
```

Lists IP socket version 6

You can use -6 (IPV6) to find the IP socket version 6:

```
ss -a -6
```

List sockets by status

You can list sockets by status. It will be divided into categories as established, active, or closed.

Use the following command if you want to find established TCP connections, the ss command will list by name:

```
ss -t -r state established
```

The four listed connections are all in established state. The hostname, ubuntu20-04, has been resolved and 'ssh' has been shown instead of 22 for the second SSH connection.

List sockets by protocol

You can list sockets using a special protocol like dport and sport, corresponding to the destination port and the source port.

Enter the following to list the sockets under the HTTPS protocol on an established connection (note opening spaces after parentheses and before closing):

```
ss -a state established '( dport = :https or sport = :https )'
```

You can use the protocol name or the port usually connected to the protocol. The default port for SSH is port 22.

List connections to special IP addresses

With dst again, you can list connections to a certain IP address.

Use the following command:

```
ss -a dst 192.168.4.25
```

Define the process

To view the processes that are using the socket, you can use the process option (-p), as shown below (note that sudo must be used):

```
sudo ss -t -p
```

```
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process ESTAB 0 0 192.1
```