COURSEWARE

Screens in Linux

Contents

Overview

Screen is a terminal Multiplexer for linux allowing long-lived unsupervised sessions.
There will be many times when the common work pattern is:

1. connect to a remote server
2. run some process on the remote server

As seen, `ssh` offers a perfectly respectable way to complete this in a single command. We may run into problems, however, if the process we wish to run on the remote server is particularly long-running.

When we `ssh` into a remote server, our user is authenticated and we begin a session and (crucially) all user processes are bound to this session. If your laptop were to drop wifi and the connection supporting the session were broken, linux will end the session and any processes belonging to it. This often happens because of a timeout, and if you are in the middle of a long-running process, will cause you no end of frustration.

`screen` is a "Terminal Multiplexer" which will keep a session open as long as the system is booted or the session explicitly killed.

Starting a screen session

To start a screen session simply type `screen` in the shell and you will be forked into a persistent session with a clean history. To name the session you can use the `-S` flag:

```
$ screen -S my-long-running-process-screen
```

Detach from a screen session

Whilst interacting with a screen session you are `attached` to the session, if anything should interrupt your connection or you simply wish to leave the screen session running, you can `detach` from it using the `action` input: `Ctrl+a` and then pressing `d` for `detach`.

The processes running within the screen session will continue to run as normal.

List the currently running screen sessions

screen may be running multiple sessions so to list them use the `-ls` flag:

```
$ screen -ls

There are screens on:
    18290.pts-0.my-long-running-process-screen   (Detached)
    20789.pts-0.bitcoin-miner   (Detached)
```

Attach to a running screen session

To attach to a specific screen session use the `-r` flag (for `restore`) along with the name/identifier of the session you wish to restore, if used without the screen name you will be attached to the most recently detached screen session.

```
$ screen -r 18290
```

Customising a screen session

Just like with `bash`, `screen` can be customised using `/etc/screenrc` for global customisations and `~/screenrc` for user level customisations.

Common uses for these files are for highlighting useful facts while on the command line such as:

- username
- hostname
- directory names
- repo and branch highlighting (very useful)

Tutorial

This tutorial will take you through a very simple use of `screen` to keep a process running independent of the ssh connection

Prerequisites

- an Ubuntu or Debian Virtual Machine

Virtual Machine Configuration

There will need to be the following configured on the VM:

- nginx needs to be installed and running

Our "long running process" for the purposes of this tutorial will be a `tail` of the logs output by the `nginx` service. While debugging issues with servers it is often worthwhile looking into the logs, and `tail` will allow us to follow the log so that we can see each new line the service sends to it.

Run the following commands on the virtual machine to configure this:

```
# install nginx
sudo apt install nginx
```

Test that the service is up and running

```
$ sudo systemctl status nginx

 nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
enabled)
   Active: active (running) since Sun 2019-04-21 13:57:01 PDT; 5min ago
     Docs: man:nginx(8)
  Process: 4491 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 -
-pidfile /run/nginx.pid (code=exited, status=0/SUCCESS)
  Process: 4502 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
  Process: 4492 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process
on; (code=exited, status=0/SUCCESS)
 Main PID: 4504 (nginx)
    Tasks: 3 (limit: 2319)
   CGroup: /system.slice/nginx.service
           |-4504 nginx: master process /usr/sbin/nginx -g daemon on;
master_process on;
           |-4516 nginx: worker process
           `-4517 nginx: worker process
```

Check the service is available

```
$ curl -i http://127.0.0.1/nginx_status
HTTP/1.1 200 OK
Server: nginx/1.11.1
Date: Tue, 01 Dec 2020 07:45:43 GMT
Content-Type: text/plain
Content-Length: 97
Connection: keep-alive

Active connections: 1
server accepts handled requests
3 3 3
Reading: 0 Writing: 1 Waiting: 0
```

Now, open a screen session and follow the nginx logs:

```
$ screen -S nginx
$ sudo tail -f /var/log/nginx/access.log /var/log/nginx/error.log
```

You should be able to verify that the logs are written to, that you are following the correct logs and that you are sending requests to the webserver, as they should contain a line detailing the http request we just sent. You will also be able to see as new requests are handled.

Next, we need to `detach` from the current screen session using the action control `Ctrl+a` and then pressing `d` which will kick you out of the screen session and return you to the original shell session. To verify that the screen is still running:

```
$ screen -ls
There are screens on:
    18290.pts-0.nginx   (Detached)
```

Send a couple requests to the web server (some that we expect to fail):

```
$ curl -i http://127.0.0.1/
$ curl -i http://127.0.0.1/test
$ curl -i http://127.0.0.1/error
```

Then we can reconnect to the screen to see how that has affected the log files:

```
$ screen -r
```

Although this is a trivial example - there is no real advantage to watching a log file in a screen - this illustrates the utility of the screen function. If, instead, we were running a shell script against a very large log file looking for instances of a

certain IP address on a remote server, we would almost certainly want to do so in a screen session because otherwise (and for many reasons) our script could be interrupted and processing stopped without warning.

Exercises

There are no exercises for this module.