My team and I regularly deal with remote SSH connections to AWS and other third party services, and part of that requires several concurrent terminal sessions running scripts, processes, etc.

Frequently, we encounter the dreaded:

```
Write failed: broken pipe.
```

There are a couple ways to mitigate this:

SSH Configuration

```
sudo vim ~/.ssh/config
```

Add the following:

```
Host *
ServerAliveInterval 120
```

Screen

> *Screen is a full-screen window manager that multiplexes a physical terminal between several processes, typically interactive shells. Each virtual terminal provides the functions of the DEC VT100 terminal and, in addition, several control functions from the ISO 6429 (ECMA 48, ANSI X3.64) and ISO 2022 standards (e.g. insert/delete line and support for multiple character sets). There is a scrollback history buffer for each virtual terminal and a copy-and-paste mechanism that allows the user to move text regions between windows. When `screen` is called, it creates a single window with a shell in it (or the specified command) and then gets*

> *out of your way so that you can use the program as you normally would. Then, at any time, you can create new (full-screen) windows with other programs in them (including more shells), kill the current window, view a list of the active windows, turn output logging on and off, copy text between windows, view the scrollback history, switch between windows, etc. All windows run their programs completely independent of each other. Programs continue to run when their window is currently not visible and even when the whole screen session is detached from the user's terminal.*

If your session terminates or fails for some reason, using **Screen** you can quickly continue back where you left off with the previous session:

```
screen -d
screen -r
```