Lecture 3 - Introduction to Remote Computing

DSE 512

Drew Schmidt
2022-02-01

From Last Time

- Assignment 1 is graded
  - Very diverse computing backgrounds
  - Not many strong opinions about what we should cover/emphasize
- Questions?

Remote Computing Basics

Ways to Interact with a Remote Resource

- Terminal commands (ssh)
- Graphical programs over ssh (X forwarding)
- Web services (may require tunneling)

Remote Tools

- `ssh` - secure shell
- `sftp` - secure ftp (file transfer)
- `scp` - secure copy
- `rsync` - "file copying tool"

File Editing

- Editing files is a major part of the job
- Major text editors
  - vim
  - emacs
  - nano/pico/whatever

Local Editor Remote Dev

- VSCode https://code.visualstudio.com/docs/remote/ssh-tutorial
- Atom https://atom.io/packages/remote-ssh
- RStudio + remoter https://cran.r-project.org/web/packages/remoter/index.html
- PyCharm https://www.jetbrains.com/help/pycharm/configuring-remote-interpreters-via-ssh.html

Some Conventions

- A leading `$` means we are at the shell
- We will use ssh config

ssh Basics

- Connects you to remote computer
- Runs a shell (sh, dash, bash, csh, ksh, ...)
- I recommend bash but it's a free country

Connecting with ssh

```
$ ssh username@hostname
```

Example:

```
$ ssh mschmid3@acf-login.acf.tennessee.edu
```

ssh config

Add a similar line to `~/.ssh/config`

```
Host isaac
HostName acf-login.acf.tennessee.edu
User mschmid3
ServerAliveInterval 30
Port 22
```

Then connect via:

```
$ ssh isaac
```

ssh Keys

- Allow you to log in without a password
- Better in every way than logging in with a password
- Some systems don't support this
  - AWS *requires* it
  - ISAAC *bans* it
  - Most systems let you choose

scp/sftp Basics

- Transfer files between local and remote system
- `scp` like `cp`
  - `scp localfile username@hostname:remotepath`
  - Scriptable, but keyless ssh requires logins every time
- `sftp` like `ftp`
  - `sftp username@hostname`
  - Brings up interactive prompt - not scriptable

Graphical Applications over ssh

- Possible if you forward X11
- Usually very slow
- Graphical applications rarely exist on HPC resource in the first place
- Mac:
  - Install XQuartz
  - `ssh ... -o "XAuthLocation=/opt/X11/bin/xauth"`
- Windows:
  - Putty: Check "Enable X11 forwarding" (Connection -> SSH -> X11)
  - WSL: ???

Web Applications and Tunnels

- Web servers can use web ports (80/443)
- Other ports need to be *opened* or *redirected*
- Single tunnel not so bad
- Multiple tunnels is painful…

Tunneling

Tunnel listen on *your laptop* on port `1234` to a remote machine listening on port `5678`

```
ssh user@remote_machine -L 1234:localhost:5678 -N
```

- Usually use the same port for each
- Can add line to ssh config: `LocalForward 1234 localhost:5678`

Using AWS

EC2

- Hardware is entirely managed
- Software environment is *unmanaged*
- Have administrative privileges

Creating an Instance

1. Log in https://aws.amazon.com/
2. Navigate to EC2 panel
3. Create Ubuntu 20.04 t2.micro instance in EC2 with 30GB of storage (this should qualify for the free tier)
4. Set up ssh keys, get instance IP
    - We'll assume key is in `~/.ssh/mykey.pem`
    - We'll call the ip `x.x.x.x`

Logging in with ssh

```
$ ssh -i ~/.ssh/mykey.pem ubuntu@x.x.x.x
```

ssh config

Add a similar line to `~/.ssh/config`

```
Host ec2
HostName x.x.x.x
User ubuntu
IdentityFile ~/.ssh/mykey.pem
Port 22
ServerAliveInterval 30
ForwardX11 yes
```

```
$ ssh ec2
```

Managing the Software Stack

- Your problem, buddy
- Changing system files requires root (administrative privilege)
  - `some_command` - run as user
  - `sudo some_command` - run as root
  - `sudo su` (or just `su` on non-Ubuntu distributions) - open root shell
- Use the software repo when you can
  - `sudo apt install r-base-dev`

Live Demo