Mark's Guide to Amazon EC2 (and MS Azure) for R and Julia

Mark Agerton

2017-06-12

This is a guide on how to set up an Amazon EC2 cluster for use with RStudio and/or Julia. There are also instructions for setting up an MS Azure instance.

For EC2, there are a few ways to get data on and off the servers in addition to the Dropbox directions provided by http://www.louisaslett.com/RStudio_AMI/. Renting EC2 space can be quite cheap, especially if you use a Spot Instance. Pricing is a continuous uniform-price auction, and if you are outbid, your instance is terminated w/out warning. Azure does not have spot-pricing like Amazon.

The guide assumes you have some basic familiarity with using a linux-like terminal (e.g., navigating the file structure, copying, moving, ssh, etc). Note that Windows 10 now includes the "Windows Subsytem for Linux" (WSL), which provides a very nice terminal environment (MSDN setup guide). The Git Bash terminal will also work nicely.

If you have suggestions, pull requests & edits are welcome!!

Launching an EC2 instance

1. Sign up for Amazon AWS account. Sign up at github.com for an education pack if eligible and you may get some free Amazon AWS credits.
2. Get your SSH keys fixed on your computer so you can log in to your EC2 instances.
    - You may need to add a file called `config` to local `~/.ssh` folder. For example it could be `IdentityFile ~/.ssh/github_rsa IdentityFile ~/.ssh/Magerton_Key_Pair.pem`
    - Make sure permissions are correct for SSH folder & keys. See Stack-exchange. If using symlinked directory in Windows Subsystem for Linux (WSL) on Windows 10, might need to change permissions to read only on Windows side if can't do this in WSL.
    - What files are:

    - `github_rsa` and `*.pem` are private keys. KEEP SECURE–these are like your password
    - `*.ppk` is PuTTy version of private key (StackOverflow)
    - `*.pub` files are public keys. These are placed on server-side and used to verify that your private key is correct.
3. Spin up an Ubuntu 16.04 or 18.04 image. ~~Go to http://www.louisaslett.com/RStudio_AMI/ and click on the AMI you want (first time only to set up)~~
    - If permanent setup, use a smaller, cheaper one to get set up. You can save money by using a Spot instance, but could get booted if your maximum willingness to pay (bid) is too low. Otherwise, get as much power as you need.
    - Make sure that you have enough drive storage (16Gb should be fine), and that the HTTP Protocol (port 80) is open in addition to SSH (port 22).
4. SSH into the instance (right click on it & hit "connect" to get the terminal command, should be something like `ssh ubuntu@ec2-54-196-121-83.compute-1.amazonaws.com`). Note that the RStudio AMI has superuser `ubuntu`, not `root` as Connect page suggests.

Software installation

1. On the remote, the directory `~/.ssh` has a file `authorized_keys`, which contains the public key counterpart for your private (local) `.pem` key. You'll want to add a github private key to this folder, and also deposit `authorized_keys` and your github private key in other uses (such as `rstudio`) with appropriate permissions. `ssh-copy-id` might be a better option to put a public key in another user's folder.
    - On the local machine, navigate to your directory w/ relevant keys (usually `~/.ssh` or `%USERPROFILE%/.ssh`).
    - Use `sftp` to put your `github_rsa` private key (and possibly also `config`) on the remote server
    - Exit `sftp`, and then `ssh` into the remote
    - Move the private key into .ssh: `mv github_rsa .ssh/`
    - Check that the permissions are correct: `ls -al .ssh`. See stackexchange.
    - You may need to make a new config file: `shell echo IdentityFile ~/.ssh/github_rsa > .ssh/config chown ubuntu .ssh/config chmod 700 .ssh/config`
    - If you need to access git or other services, copy over .ssh files to `~/.ssh`. `chmod` the directory to 700 and files to 600.
2. Install needed programs
    - Add `lftp` to machine to connect to Box.net accounts. `shell sudo apt-get update sudo apt-get install lftp`
    - Add `git-lfs`.

    - See install guide. I had to use PackageCloud to install from command line.

```
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh
sudo apt-get install git-lfs
git lfs install  # only run once for initial install
```

Julia setup

- Install julia
    - `wget` the file on https://julialang.org/downloads/index.html
    - `tar -xvzf [download name]`
    - Symlink to `/usr/local/bin` by running `sudo ln -s <where you extracted the julia archive>/bin/julia /usr/local/bin/julia`. Note, you'll want to use the FULL path of the directory julia got extracted to (eg, `/home/ME/juliaarchive/bin/julia`)
- Update pkgs `sudo apt-get update`
- Install build tools: `sudo apt-get install build-essential`
- Install `sudo apt-get install hdf5-tools` from command line
- Open up a julia prompt and install packages into the default folder

```
]add AxisAlgorithms BenchmarkTools Calculus CategoricalArrays DataFrames Distributions

dev ssh://git@github.com/magerton/CountPlus.git
dev ssh://git@github.com/magerton/Halton.git
dev ssh://git@github.com/magerton/JuliaTex.jl.git
dev ssh://git@github.com/magerton/GenGlobal.jl.git
dev ssh://git@github.com/magerton/MarksRandomEffects.git
dev ssh://git@github.com/magerton/OrderedResponse.jl.git
dev ssh://git@github.com/magerton/MarkovTransitionMatrices.jl.git
dev ssh://git@github.com/magerton/ShaleDrillingModel.jl.git
dev ssh://git@github.com/magerton/ShaleDrillingData.jl.git
dev ssh://git@github.com/magerton/ShaleDrillingEstimation.jl.git
dev ssh://git@github.com/magerton/ShaleDrillingPostEstimation.jl.git
```

- ~~Initialize package repo with `Pkg.init()` in julia~~
- ~~Bulk install by updating `REQUIRE` in `~/.julia/v0.x/REQUIRE` and running `Pkg.resolve()`. You may need to run julia as `sudo` with elevated priveleges, but hopefully not.~~

Setting up GUI/ remote desktop (RDP) for remote machine via secure SSH tunnel

- Remote Desktop
  - Install `xrdp` and `xcfe4` software as per https://docs.microsoft.com/en-us/azure/virtual-machines/linux/use-remote-desktop. We'll connect over SSH, so no need to open a special RDP port.
  - On local machine, create ssh tunnel to remote with port forwarding `ssh -L [LOCALPORT]:localhost:[3389] username@remoteip`. Can do this with terminal, Bash for Windows, or Putty.
  - After connecting to remote instance, set a password on the remote machine so that the RDP can log in `sudo setpasswd [yourname]`
  - Open Remote Desktop Connection (search for mstsc.exe on Windows) & log in to `localhost:[LOCALPORT]`
  - To be able to reconnect to the same desktop, see http://c-nergy.be/blog/?p=5305 and https://askubuntu.com/questions/133343/how-do-i-set-up-xrdp-session-that-reuses-an-existing-session. Basically, the idea is to edit the xrdp ini file to allow this. Run `sudo [gedit/pico/vim] /etc/xrdp/xrdp.ini` and change section `[xrdp1]` where it says `port=-1` to `port=ask-1`. When logging in for the first time, leave the port as `-1` and note the port number you get (will default to `5910`). Then on subsquent logins, change the port to whater the previous one was (I it *should* default to `5910`). Sessions seem to persist even when the SSH tunnel is closed.
- Install the gnome terminal `sudo apt-get install gnome-terminal`, or something better than the `xcfe` terminal. This should swap out automatically if you open a new terminal window
- Install unzip (at least if on Azure): `sudo apt-get install unzip` so that Julia can build `HttpParser` for Atom
- Fix tab-completion by following https://www.starnet.com/xwin32kb/tab-key-not-working-when-using-xfce-desktop/
- Install Firefox using `sudo apt-get install firefox`
- Installing Atom
  - Download .deb file from https://atom.io/
  - attempt to install with `sudo dpkg -i atom-amd64.deb`
  - After error, run `sudo apt-get install -f`
  - Then again `sudo dpkg -i atom-amd64.deb`
  - Follow https://github.com/atom/atom/issues/4360#issuecomment-205122828 to get Atom to run. You can find the file with `bash dpkg -L libxcb1 # to find the file   cd /usr/share/atom cp /usr/lib/x86_64-linux-gnu/libxcb.so.1   sudo sed -i 's/BIG-REQUESTS/_IG-REQUESTS/' libxcb.so.1`

Subsequent work using terminal or REPL

1. Launch your AMI from the EC2 console: `AMI > Select on your AMI > Under "Actions," select "Spot Request"` Request a big instance, and set the MAX price you are willing to pay per hour (usually it's much lower than this)
2. Once your AMI is running (can take a bit), SSH into it & get to work or point your browser to the relevant IP address.
3. To save intermediate log-files, you could use a `cron` job to run a script (named like `myscript.sh`, has a shebang w/ your shell path at the top– google it) that would ssh into the server & pipe log file to your home computer. (Might also be something as simple as a command – no script needed) See https://www.cyberciti.biz/faq/howto-use-tar-command-through-network-over-ssh-session/

Using LFTP

- Alternately, transfer using `lftp` and Box.net. Note that special characters in password may have to be escaped or translated to HTML. `shell lftp -p 990 -u "netid@rice.edu,PASSWORD" ftps://ftp.box.com mirror [project_dir_on_box]   [remote_project_dir]`