R on Amazon EC2

Jay Emerson, Susan Wang, and Dan Snyder

4/17/2018

Jay and Susan, Department of Statistics, Yale University, 2013 (original version); 2016 (revised); February 2017 (revised again)

This guide is divided into several sections. The first section should help you create an EC2 instance on your own from one of the Amazon basic images. The second section describes customizations related to R. This process is essentially what Bioconductor (link) would have done in creating its image(s) with customizations. It may be there are some images for `R` and `shiny-server` that are ready to go on EC2, but we haven't looked recently.

Finally, we show how to take this basic process to create an instance for the purpose of building an on-demand cluster (not checked in 2016 or 2017 revisions). With only a few extra steps this cluster can be used with R's `foreach` (via `doSNOW`) for parallel computing.

Step-by-step instructions for setting up a (possibly free) T1 Micro instance

1. Sign up for an account (if you haven't)! Set up the account for whatever region is appropriate (I used something in the Eastern USA). Start here: http://aws.amazon.com/console/.
2. Create a security group. From the `EC2 Dashboard` choose `Security Groups` under `Network and Security`. I added a group with a VPC, and added rules for inbound traffic. Make the obvious choices (SSH and HTTP on ports 22 and 80).
   If you will be using Shiny, also open up port 3838 and (possibly) 10187 for default use of parallel programming tools (see below).
3. From the `EC2 Dashboard` under `Instances` choose `Launch Instance`. My account is in the US East (N. Virginia) region. This will begin the process of creating the virtual machine you will log into.
4. Select the AMI (Amazon machine image) you would like to launch. I selected `Ubuntu Server 16.04 LTS` with the 64-bit version. This is eligible for the free tier model. It is bare-bones; choose something bigger if you need it.
5. I'll create one (1) instance of the free tier eligible `T2 Micro` instance type. Don't press the blue `Review and Launch` button yet… be careful here. Then choose `Next: Configure Instance Details`. In `Step 3: Configure Instance` set `Auto-assign Public IP` to Disable. This will allow you to assign an Elastic IP address after launch. In `Step 6: Configure Security Group` you should select the security group you created earlier. Now you are ready to go.
6. After selecting the correct security group for your instance, click the blue "Launch" button in the lower right corner of the page. If you haven't created a key pair yet, choose create a public/privite key pair option. These are the keys to the server. If you lose them, you'll be locked out. Download the key file and keep it in your `~/.ssh` directory (on Linux / Mac OS X systems). If you already created a key pair, select the "Choose an existing key pair" option. Next, click `Launch Instances`! After this step, there will be links that direct you to areas of the dashboard where you can set up billing alerts or monitor the status of your instance. The instance should report it as launching pretty quickly. You can watch from `EC2 Dashboard`. Wait until the state is given with a green circle as `running` and it indicates the initialization is complete.
7. If the public IP is automatically assigned, the address will change after every launch. We would like to assign the same public IP address to our instance every time it is launched. So we will create an Elastic IP address: an IP address that can be reassigned to different EC2 instances. Go to `Network &`

`Security` and click on `Elastic IPs`. Click `Allocate New Address` and follow the steps to pull an IP address. In the next step we will attach this address to your instance's network interface `eth0`. The name `eth0` is the Linux term for the primary internet connection to the computer.

8. Next go to `Network Interfaces` under `Networking and Security`. If your instance launched, you should see the network interface for your instance listed here. This is the interface to which we will bind the Elastic IP address. With the desired network interface selected, choose `Associate Address` from `Actions` menu at the top of the console. The first drop down should make your Elastic IP address available for association with your instance's network interface. The `resassociate` check box allows for this assignment to override any previous assignments (because the Elastic IP can only associate to one network interface at a time)
9. Optionally, on your local computer, we will configure an alias for the ec2 instance. Doing this will make it much easier to connect to your instance. Say that your instance's public IP address (the Elastic IP address you assigned to your intance's network interface) is 19.20.21.22. Also, assume that your public key file is `ec2yale.pem`. In the `~/.ssh/config` file, add an entry that looks like this (using a plain text editor such as vim).

```
Host ec2
    Hostname 19.20.21.22
    IdentityFile ~/.ssh/ec2yale.pem
    User ubuntu
```

10. To connect to your instance, you'll use a command line interface. Log into the instance using the `ssh` command (secure shell). On your local machine (Mac OS X or Linux), ensure that your key file `ec2yale.pem` is in the `~/.ssh/` directory. If you lost the key file, you will need to terminate the instance and start over. The key file permission must give access to your user only. Key file permissions should be set to 400 (user execute, hidden to all others). To ensure this configuration, execute: `chmod 400 ec2yale.pem`. The two trailing zeros in `400` signify that the key is not accessible to the group or the world. If you did not set up the hose alias described in the previous step, the log in ssh command will look like:

```
ssh -i ~/.ssh/ec2yale.pem ubuntu@19.20.21.22
```

If you've set up the ssh config file specifying alias name `ec2`, the command will be:

```
ssh ec2
```

You may be asked if you are sure you want to connect to the remote machine at hte IP address specified. To allow this, type `yes`.

Windows? You'll need a Windows SSH/SFTP client. Check on Yale's software page (link). You'll need to set it up to use this same keyfile. I've never tested this, so if it doesn't work or of you discover something that should be added to this document, please let me know.

11. Voila! We're in.

A little about Linux

In this context, everything you do will be at the command line, typing. Your mouse is useless. You don't need to know much to pull this off. Some basic navigation:

```
pwd
ls
ls -lat
mkdir Stat662
ls -lat
cd Stat662
```

```
pwd
ls
cd ..
pwd
ls
```

A command reference (link) that some have found useful. Editing a text file or a script? I've pointed students at the following introduction to vi (link), although some students also use emacs, with a Yale help page (link) or something from Colorado State CS (link). I should really add more to this section.

Customization of the instance

We provide several scripts: a shell script and some R scripts. These basically install R and add-on packages. The scripts sets up Shiny server from R Studio; this part of the installation might require that you hit `return` at some point.

To do this customization, SSH into the instance, and pick up the shell script (if you want to run my version without modification). Otherwise, edit `InstallR.sh` to use your own R scripts (at your own risk). Some possible edits would be the flags at the beginning of the file that say whether or not to install *extras* and *mysql*. The extras include shiny server (web hosting and data visualization) and foreach (parallel computing). The mysql option will allow you to setup a database server.

```
instance-prompt$ sudo su
instance-prompt$ wget http://www.stat.yale.edu/~jay/EC2/InstallR.sh
instance-prompt$ chmod +x InstallR.sh
instance-prompt$ ./InstallR.sh
```

Once this is complete, you're good to go. On the way you may need to answer 'y' or 'Y' or something, or press `Return` on request. Just do it.

Testing the Web Server and Shiny

The main web server is at your IP address (here's an example):

```
http://54.210.104.9
```

and may be customized by editing `/var/www/html/index.html`. Shiny Server should be running, try visiting

```
http://54.210.104.9:3838
```

You should see a working Shiny app in the top right, and a Shiny Doc in the bottom right. The folder for Shiny apps seems to be `/src/shiny-server`, so if you use `sftp` to move a folder (say, `ESG` for my example) there you can run it at

```
http://54.210.104.9:3838/ESG/
```

Building an on-demand cluster for parallel computing

This was not checked as of the 2016 revision.

This section simply extends material discussed above. You can build a compute cluster (CC) of any number of nodes of a certain type. Amazon's CC1 instance is 4 cores, and a CC2 instance is 8 cores (CHECK, EXPAND). Options may be limited depending on the Amazon region (we are using `N. Virginia`.

1. On your Dashboard, click `AMIs` under `Images`. Modify the filter for public images, and search for `ami-0745d26e`. Select it, and launch. Choose your instance type (we use `cc1.4xlarge` which may be 4 core, 23 GB RAM, with over 800 GB of dedicated – not shared – disk space). You need to specify a

placement group; we called ours `CC1-cluster`. Click through stuff as above (including your key pair), specify your security group, and launch.

2. Back in the dashboard, click `Volumes` under `Elastic Block Store`. Create a 200 GB standard volume. Be sure that the volume resides in the same `Availability Zone` as the instance. Attach the volume to the instance, picking `dev/sdh` as the mount point. This will be shared across your cluster.

3. Login and sudo su.

4. Format and mount (note that the "s" in the path has been replaced by "xv") > mkfs.ext4 /dev/xvdh
   > mkdir -m 000 /vol
   > echo "/dev/xvdh /vol auto noatime 0 0" | sudo tee -a /etc/fstab
   > sudo mount /vol

5. Install stuff (InstallR.sh) as explained above.

6. Deal with SSH keys. To do this, do **not** be logged in as root! The command `sudo su` logs you in as root user. Files created while logged in as root user will not have the correct file permissions. You are user `ubuntu`. Just hit `return` when asked any question about a passphrase. > su ubuntu
   > cd
   > ssh-keygen -t dsa
   > cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
   > chmod 644 ~/.ssh/authorized_keys

7. Go back to EC2 Dashboard, click on instance and under Actions, select "Create Image".

8. Maybe name your image something like `CC1-image` and then `Yes, Create`. This will add the image to `Images` under `AMIs`, and will take a little while (rebooting your master instance in the process).

9. Select the image, click Launch. Select the CC1 instance type with quantity 1. If we wanted, say, 10 slaves, then we would pick 10 here. Make sure this instance shares the same security group and placement group as the master.

10. Log into the master, and copy the `Public DNS` names of the slave machines into a new file `/vol/nodelist`, one per line. Do not include the master.

11. Edit Security Group to have port 10187 opened (could reconsider this later).

12. Fire up R. We will now do the same sort of exercise as before, making use of 8 cores, 4 of which reside in the master "localhost", and 4 others that come from the slave in the "nodelist" file.

Here, we show a simple calculation done in parallel via `doSNOW` with 8 cores on the 2 nodes:

```
library(doSNOW)
library(itertools)
setDefaultClusterOptions(port=10187)
machines <- readLines("/vol/nodelist")
machines <- rep(c("localhost", machines), each = 4)
cl <- makeCluster(machines, type = "SOCK")
registerDoSNOW(cl)
N <- 1e6
a <- matrix(1:N, nrow=4)
system.time({
    b <- foreach(j=isplitIndices(ncol(a), chunks=length(machines)),
                .combine=c) %dopar% (apply(a[,j], 2, sum))
})
stopCluster(cl)
```

Terminating your cluster

This is important, otherwise you'll get billed about $2.50 an hour for these two nodes! In the Dashboard, select all the machines and terminate. Alternatively, you may just `stop` and we're trying to figure out what the billing implications are of having this stuff stopped (not running) but available. This could be a nice option if you use this regularly.