Set up Shiny on AWS server

(<http://trestletechnology.net/2013/02/deploying-shiny-server-on-amazon-ec2/>)

Setting up Shiny Server on [Amazon’s Elastic Compute Cloud](http://aws.amazon.com/ec2/) (EC2) can be a bit of a process if you’re unfamiliar with Linux system administration. Thankfully, we’ve done all the hard work for you! If you’re looking to get Shiny Server running in EC2 without delay, just boot up an instance of our public Amazon Machine Image (AMI): [ami-e2a3358b](https://console.aws.amazon.com/ec2/v2/home?region=us-east-1" \l "Images:filter=all-images;platform=all-platforms;visibility=public-images;search=ami-e2a3358b" \t "_blank) named “ShinyServer.”

If you’re interested in the details, you can view the complete description below.

If you want to recreate this image yourself, you can follow these steps, but you’ll need a running instance on EC2 first. If you’re unfamiliar with EC2, read up on how to create and access EC2 instances [here](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html" \t "_blank). Go do that now.

…

Welcome back! I’ll now assume that you’re able to create and access EC2 instances. For this tutorial, you’ll likely want to create a fairly powerful instance — the process of building R and Node.js from scratch can take 20-30 minutes even on powerful hardware — it could take hours or even be impossible due to memory constraints on a smaller instance. Of course, it should only take an hour or two to get everything installed. At that point, you can create an image of your system and later boot it up with whatever size instance you care to use.

Note that Amazon EC2 instances use volatile storage — this means that any changes you make to the system (including changes written to the hard drive) will be lost the moment you terminate your instance! Cloud instances are treated as disposable and are no more reliable than a physical server (i.e. can crash at a moment’s notice).

Once you’ve created a system you’d like to keep, be sure to create an image of that system so that you can boot up other instances of that system at a later date. If you’re new to EC2, take a few minutes to play around with this practice before diving into this guide — I don’t want to have the hours you lost to an EC2 instance that vanished weighing on my conscience…

Go ahead and create a Large (or bigger) instance and log in. Once we’ve logged in, it’s always a good idea to ensure that we’re using the latest packages Amazon provides. We can update with

sudo yum update -y

## Install R & Shiny

We can now install the packages we’ll need in order to be able to build R from source:

sudo yum install gcc gcc-c++ gcc\_gfortran readline-devel make texinfo \ libX11-devel libX11-devel.\* libICE-devel.\* libSM-devel.\* libdmx-devel.\* \ libx\* xorg-x11\* libFS\* libX\* tetex mesa-libGLU-devel libjpeg-devel \ libpng-devel cairo-devel pango-devel -y

Once we have those packages, we’re ready to build R. Download the latest version available from the [R Project home page](http://www.r-project.org/). At the time of writing, that’s 2.15.2 which we can download and untar using the following commands:

wget http://cran.rstudio.com/src/base/R-2/R-2.15.2.tar.gz tar -xvf R-2.15.2.tar.gz cd R-2.15.2

We’re now in the untar’d directory containing the source code for R. We can configure it to not expect X, since we’re running on a headless server, then compile it using make:

./configure make make check # technically optional

Depending on the size of instance you chose, this could take quite some time. On a large instance, configuration took a minute or two and compiling using make took 20 minutes. The check process took only 2-3 minutes . If you chose a larger EC2 image which has more than one core and sufficient memory to run both builds simultaneously (“Large” or above), you can actually begin the compilation process of Node.js (see below) so the longest steps run concurrently. Now go get some tea.

If everything was able to build correctly, then you can install R to the regular locations using

sudo make install

Confirm that R installed successfully using

R --version

which should return the version of R you downloaded and installed. We’re now ready to install the Shiny package from CRAN. To install the shiny package into R globally across the system, run the following command:

sudo su --c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""

to install the Shiny package. It make take a couple of minutes to grab the package and all of its dependencies. When that’s ready, you can exit R so we can  proceed with setting up the Node.js environment.

## Setup Node.js

Node.js requires Python in order to install properly; we haven’t installed that yet, so we’ll need to:

sudo yum install python27 -y

Shiny Server expects Node.js with a version of at least 0.8.16. The latest stable release at the time of writing is 0.8.19 which can be downloaded from GitHub to assemble from source.

wget https://github.com/joyent/node/archive/v0.8.19.tar.gz tar -xvf v0.8.19.tar.gz cd node-0.8.19

Now you’re ready to configure and compile Node.

./configure make

Just like R’s compilation, the make step will take some time. On a large instance, it took about 10 minutes . If it was able to finish compiling without errors, go ahead to install the binaries in the usual spots:

sudo make install

You should now be ready to go with Node!

## Install Shiny Server

We’ll first need to grab the latest code for Shiny. At this point, RStudio’s just keeping it on GitHub, so we’ll install git and clone the repository. If you’re opposed to installing git, you could just download a version of the code using wget.

sudo yum install git -y

Now we can clone their repository and begin the installation process:

git clone https://github.com/rstudio/shiny-server.git sudo npm install -g shiny-server

It’s possible that you’ll get an error of “npm: command not found” because npm isn’t on the root user’s path. If this happens, just find npm and run the command after providing the explicit location. For instance, mine looked like this:

[ec2-user@ip-1-1-1-1 ~]$ sudo npm install -g shiny-server/ sudo: npm: command not found [ec2-user@ip-1-1-1-1 ~]$ whereis npm npm: /usr/local/bin/npm [ec2-user@ip-1-1-1-1 ~]$ sudo /usr/local/bin/npm install -g shiny-server

which should reoslve the issue. You now have an Amazon image with Shiny Server installed! There are just a few minor steps to get everything working with the default configuration.

Shiny Server will attempt to run as the user named “shiny” by default. So we’ll need to create that user and the directories Shiny Server will attempt to use by default for logging and web files.

sudo useradd -r shiny sudo mkdir -p /var/shiny-server/www sudo mkdir /var/shiny-server/log

Note that you can omit these steps if you plan to configure Shiny Server to run as a different user or use different directories. If you plan to customize the configuration, then copy the provided configuration file to the expected system location:

sudo mkdir -p /etc/shiny-server/ sudo cp shiny-server/config/default.config /etc/shiny-server/shiny-server.conf

Then make whatever configuration changes you’d like to the file at /etc/shiny-server/shiny-server.conf (You can view all of the various configuration options online at [https://github.com/rstudio/shiny-server/blob/master/config.html](http://htmlpreview.github.com/?https://github.com/rstudio/shiny-server/blob/master/config.html" \t "_blank)).

You’re now ready to test Shiny Server by deploying a sample application. You can use an application of your own or use some of our sample apps.

cd /var/shiny-server/www sudo git clone https://github.com/trestletech/shiny-sandbox.git .

(Don’t forget the “.” at the end of line 2.) You should now have a “sampleapp” application, among others, within Shiny Server’s jurisdiction. Start up Shiny Server and make sure everything comes online properly.

sudo shiny-server

This may not work, again due to path issues (“shiny-server command not found” or “node: command not found”). It’s probably easiest to just add whatever path is missing to the PATH variable explicitly for now.

For instance, if “whereis shiny-server” produced a location of “/usr/local/bin/shiny-server”, you’d want to add “/usr/local/bin” to the PATH variable via:

sudo -i export PATH=\$PATH:/usr/local/bin #to do this automatically when you login as root echo "export PATH=\$PATH:/usr/local/bin" >> ~/.bashrc shiny-server

Now that shiny-server and node are on your path, you should be able to start the server and see something like the following:

[2013-02-08 03:38:07.115] [INFO] shiny-server - Shiny Server v0.3.0 (Node.js v0.8.19) [2013-02-08 03:38:07.117] [INFO] shiny-server - Using config file "/usr/local/lib/node\_modules/shiny-server/config/default.config" [2013-02-08 03:38:07.202] [INFO] shiny-server - Starting listener on 0.0.0.0:3838

Which shows that, if you haven’t changed the configuration file, shiny will listen for connections from any machine on port 3838. You can try visiting your Amazon instance on port 3838 by looking up its public DNS (the same one you used when SSHing into the machine) and visiting the URL of the format"http://<your amazon machine name>:3838/" which will show a list of all deployed applications on your Shiny Server. You can click any of them to visit them. Note that, if you’re using our sample apps, there are some additional packages and configuration required to use some of these applications — try “sampleapp” to start off with.

If you are having trouble connecting, first check that your Amazon EC2 Instance is accessible on port 3838 universally (see [this site](http://cloud-computing.learningtree.com/2010/09/24/understanding-amazon-ec2-security-groups-and-firewalls/" \t "_blank) for more details). Also check the shiny-server is currently running on your server.

### Start on Boot

The final step is optional, but if you want shiny-server to start automatically when the machine boots up, then you can use the Upstart system in order to make the server start on boot and also refresh itself if it ever crashes. An upstart script comes with Shiny, but some modifications are required in order to make it work with the default Amazon Linux AMI. Below is the file that is working on my instance:

# shiny-server.conf description "Shiny application server" #start on stopped networking start on stopped rc RUNLEVEL=[S3] stop on runlevel [016] limit nofile 1000000 1000000 exec /usr/local/bin/shiny-server >> /var/log/shiny-server.log 2>&1 respawn

Copy this text into a file saved at “/etc/init/shiny-server.conf” Then reboot the image to see if shiny-server restarts itself when the machine boots.

sudo reboot

If you’re still able to access you Shiny applications, then you should be in good shape. **Remember that Amazon EC2 instances use volatile storage, so all of the files on the hard drive will disappear the moment you terminate this image!** If you were hoping to save all of this work to use again later, you can create an AMI of your instance to be able to boot up more instances like this one at a later date. See [this page](http://www.techrepublic.com/blog/datacenter/how-to-create-a-new-ami-from-a-snapshot-and-launch-a-new-vm/5349" \t "_blank) for help dealing with Amazon AMIs.

current node.js version is v0.10.16

wget <http://nodejs.org/dist/v0.10.16/node-v0.10.16.tar.gz>

tar –zxf node-v0.10.16.tar.gz

cd node-v0.10.16

./configure && make && sudo make install

export PATH=$PATH:/usr/local/bin/node

install shiny package machine wide site library

sudo su - -c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""

git clone https://github.com/rstudio/shiny-server.git

(whereis npm -> /usr/local/bin/npm)

sudo /usr/local/bin/npm install -g shiny-server

sudo useradd -r shiny

sudo mkdir -p /var/shiny-server/www

sudo mkdir /var/shiny-server/log

sudo yum install xorg-x11-xauth xterm

sudo yum -y install make libX11-devel.\* libICE-devel.\* libSM-devel.\* libdmx-devel.\* libx\* xorg-x11\* libFS\* libX\*

sudo yum install gcc gcc-c++ gcc\_gfortran readline-devel make texinfo \ libX11-devel libX11-devel.\* libICE-devel.\* libSM-devel.\* libdmx-devel.\* \ libx\* xorg-x11\* libFS\* libX\* tetex mesa-libGLU-devel libjpeg-devel \ libpng-devel cairo-devel pango-devel –y

sudo yum install cairo-devel

sudo yum install libpng\*

wget http://cran.rstudio.com/src/base/R-3/R-3.0.1.tar.gz

tar -xvf R-3.0.1.tar.gz

cd R-3.0.1

sudo ./configure

sudo make

sudo make install

(this did not work, got error, bash: R: command not found)

sudo su --c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""

sudo –i

R -e "install.packages('shiny', repos='http://cran.rstudio.com/')"

Install shiny server

cd

git clone https://github.com/rstudio/shiny-server.git

sudo /usr/local/bin/npm install -g shiny-server

sudo -i shiny-server