How to set up geospatial software in AWS EC2 the right way

Manish Verma

This document describes the steps I followed (with help from my son, who is a Linux guru) to set up R, RStudioServer, Python, and Jupyter Notebook for geospatial analysis.

We describe a method below that allows you to set up your own password and username for logging in to the server.

We first created an Amazon Linux 2 instance. The package manager, `amazon-linux-extras`, only supported R 3.4 and we discovered that some of the most important packages that I wanted such as 'sf' and 'lwgeom' are not available for 3.4 anymore. So, we terminated the instance and created a new one with the latest version of Ubuntu (Ubuntu 20.04 at the time of this writing).

You can read the AWS documentation about how to set up an EC2 instance. When creating the instance, make sure that you configure the Security Group settings to allow SSH and also open the port 8787 for TCP connections so that you can connect to RStudio Server via web. RStudio Server Open Source does not support encrypted connections (HTTPS) so be careful about which IPs you want to allow. Ideally, you want to only allow the IP of the machine you are using. If it changes you can always edit the Security Group. Once you are logged into your VM, follow the steps below to set up the software. You can also open port 8888 if you want to use Jupyter Notebook, which *does* support HTTPS.

Update your packages

```
sudo apt update && sudo apt upgrade
```

You may have to install some C libraries that R packages depend on. The following are two libraries that are dependencies of rgdal and units:

```
sudo apt install libudunits2-dev libgdal-dev
```

Install and configure RStudio Server.

1. Install R and RStudio Server according to the instructions from the RStudio website.

   ```
   sudo apt install r-base # should get you R 3.6.3
   sudo apt install gdebi-core
   wget --no-verbose https://download2.rstudio.org/server/bionic/amd64/rstudio-server-1.2.5042-amd64.deb
   sudo gdebi rstudio-server-1.2.5042-amd64.deb
   ```

After the last command, RStudio Sever (the binary is called `rserver`) should be started automatically. If you want to make sure you can run `ps aux` to see a list of all processes that are currently running.

2. If you were to navigate to your EC2 instances public IP right now, there would be one main issue: you would not be able to log in because your default account (`ubuntu`) does not have a password. Set a password for your default account by running

   ```
   sudo passwd ubuntu
   ```

3. Now reload the server so it updates its configuration

   ```
   sudo systemctl restart rstudio-server.service
   ```

4. Navigate to `http://{EC2 instance's public ip}:8787` in your web browser and enter the username and password for your default account.

Set up a Jupyter Notebook

This is somewhat easier.

1. Install Anaconda Python (or whatever distribution you prefer)

   ```
   wget --no-verbose https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh
   bash Anaconda3-2020.02-Linux-x86_64.sh
   ```

2. Create a self-signed TLS certificate by running

   ```
   mkdir ~/ssl
   cd ~/ssl
   openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout mykey.key -out mycert.pem
   ```

   This allows you to log in to the RStudio Server from your web browser using HTTPS instead of HTTP.

3. Create a Jupyter Notebook password by running

   ```
   jupyter notebook password
   ```

4. Start the server and connect to it. First, run the following on your EC2 instance:

   ```
   jupyter notebook --certfile=~/ssl/mycert.pem --keyfile ~/ssl/mykey.key
   ```

   Then, run the following on your local machine to forward your local port 8888 to port 8888 on your EC2 instance:

   ```
   ssh -i {EC2 private key} -NfL 8888:localhost:8888 ubuntu@{EC2 instance's public ip}
   ```

   Finally, visit https://localhost:8888 and log in.