Jupyter Notebooks on AWS EC2 in 12 (mostly easy) steps [updated April 2019]

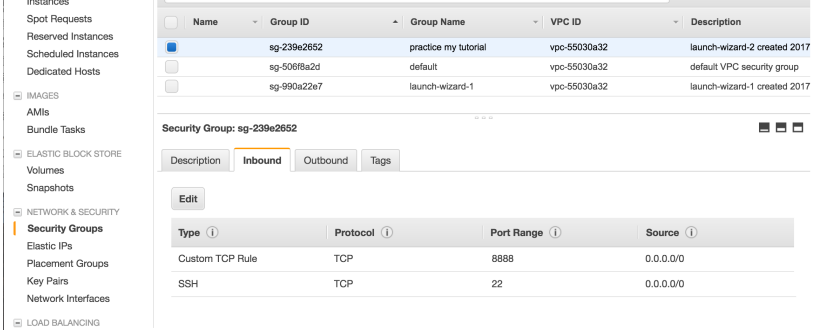
Alex Sanchez Follow
Jun 30, 2017 · 9 min read

. . .

Please note that I have updated this tutorial by simplifying the process from 15 steps to 12! Now you connect to your notebook via SSH tunneling instead of HTTPS. I think this method is better and it is MUCH easier to setup. I also fixed some minor errors and updated some of the commands so they work with the latest EC2 and Anaconda3 patterns. I hope you enjoy!

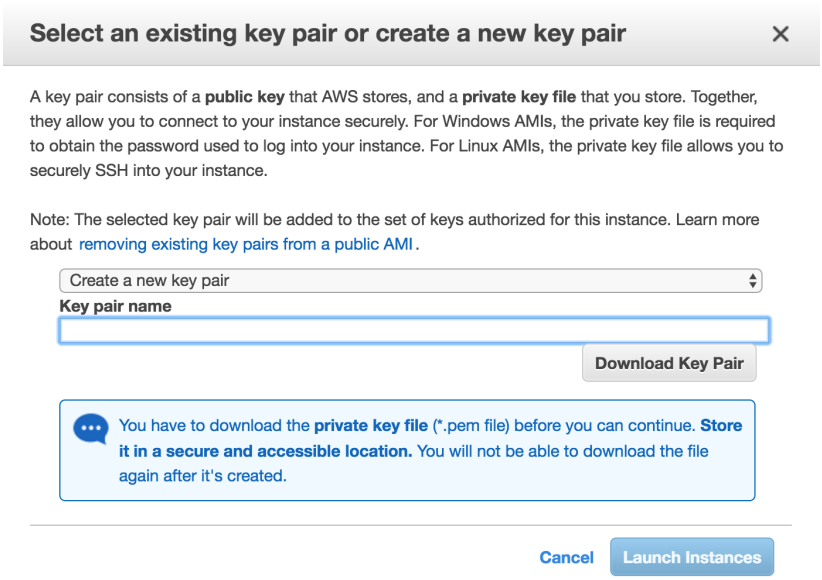
My need to run Jupyter Notebooks on EC2 came from needing more powerful resources for training a machine learning models for a Kaggle challenge. On my personal MacBook Pro it would take almost a month, but on 16 core cloud machine, it would take 5–7 hours. Fortunately, there were some resources available online to guide me and I'd like thank Chris Albon for his incredibly helpful guides on which this one is based. Hopefully, this guide will save people time so they can get coding faster!

1. Use your current Amazon user id and password or create an IAM user with your existing login.
2. Follow the default settings to create an EC2 instance and choose the Amazon Linux OS. For now, just select the free-tier t.2 micro as your instance type. You can change this in the future to something more powerful that fits your project's needs.
3. Instead of clicking "Review and Launch" button right away, click the "Next:" button until you get to Security Groups. Under Security Groups, select an exiting group or create a new one then open the inbound port 8888. Port 8888 is what we'll use for the Jupyter Notebook server towards the end of this tutorial. "SSH-ing" into port 22 should already be set to open by default.



4. Follow the rest of the prompts using whatever the default values are and then finally click "Launch." At this point, you should be prompted with some security key options. You can use an existing key or download a new one. Let's assume you don't have one yet. The PEM file is a key that AWS will check when you try to access (or SSH) into your EC2 instance from your local computer's terminal.

Select the option to create a new one and give it an easy to remember name (all lower case with no spaces for ease of typing too). Download the PEM file and put it in an easy to reach location like your home folder. Now you can finally launch your EC2 instance by selecting… you guessed it… Launch Instance.



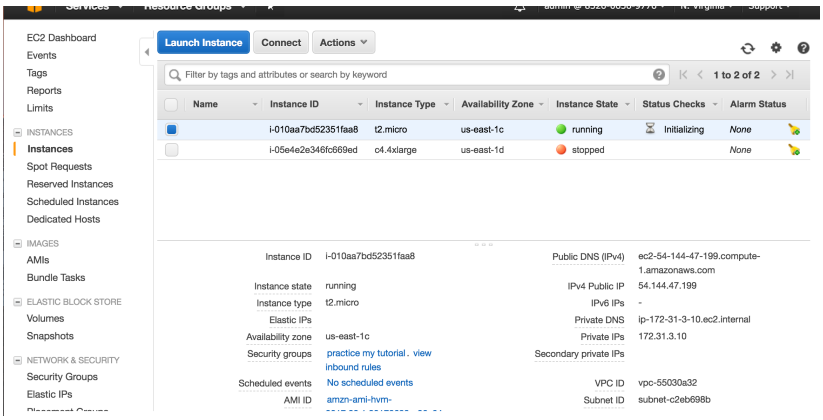
5. Once the EC2 instance is up (usually take a minute or less), SSH into your EC2 instance from your terminal by typing:

```
ssh -i "tutorialexample.pem" ec2-user@ec2-54-144-47-199.compute-1.amazonaws.com
```

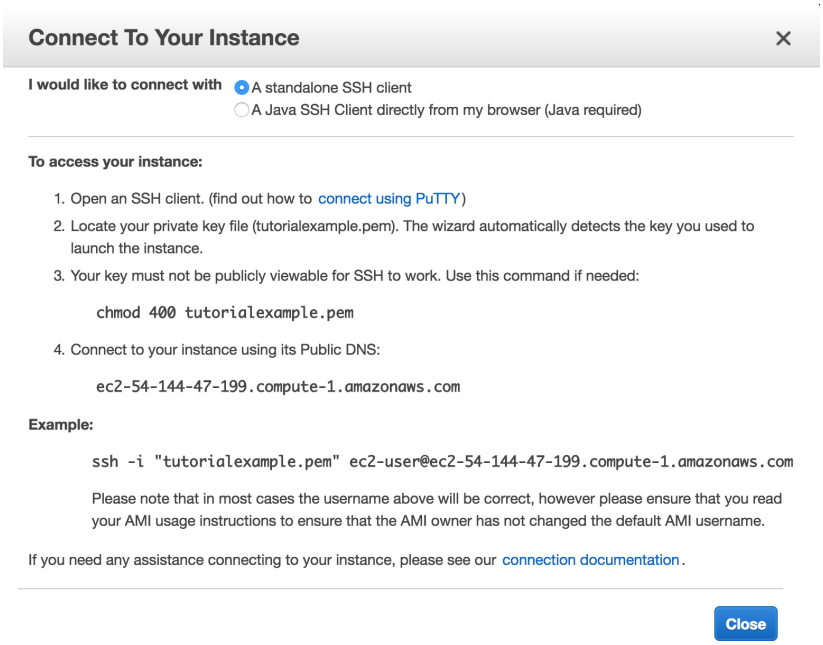
This assumes your PEM file is in the same directly as your present working directory (type "pwd" into your terminal if you're unsure what your present working directory is). Also you'll need to type in your own user@ec2-instance address. If you've been following my steps so far the user is "ec2-user" (this is the default) and the address after "@" is your instance's Public DNS (IPv4) address which you can view by selecting your instance (see

screenshot below). If you get an error that your PEM file is not publicly viewable, you made need to execute this command:

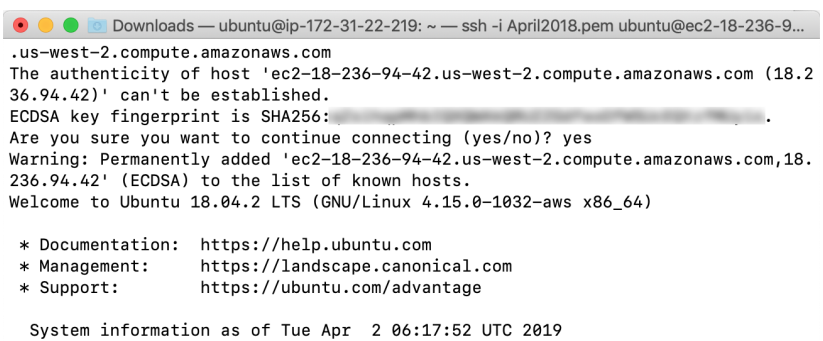
```
chmod 400 tutorialexample.pem
```



Alternatively, you can click the Connect button next to Launch Instance (see above) and AWS will give you your instance specific instructions for SSH-ing in (see below). Thanks AWS!



Once you execute the SSH command, you'll be prompted with a yes/no question. Type yes and you should be SSH-ed into your instance! (see screenshot below).



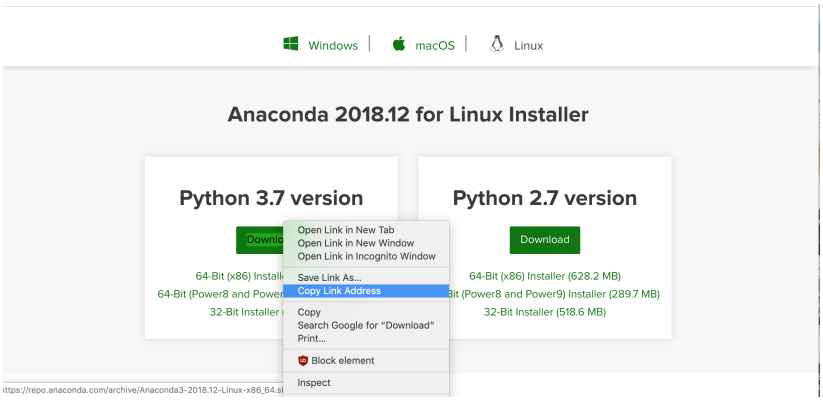
```
  System load:  0.0               Processes:           86
  Usage of /:   13.6% of 7.69GB   Users logged in:     0
  Memory usage: 14%               IP address for eth0: [redacted]
  Swap usage:   0%

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

6. Give yourself a big pat on the back. You're now running a virtual machine in the cloud!

7. Download Anaconda 3 installer for Linux by typing this command:

```
wget https://repo.anaconda.com/archive/Anaconda3-2018.12-Linux-x86_64.sh
```

Note: If you're reading this tutorial in the distant future, you can get the exact link to the latest version on Anaconda3 by going to https://www.anaconda.com/distribution/#download-section and copying the link to the latest command line installer (see below).



*** Make sure you choose the Linux installer

8. Install Anaconda3 by typing:

```
bash Anaconda3-2018.12-Linux-x86_64.sh
```

You'll have hit enter to get thru all the legalese, but eventually you need to type yes to agree and then just hit enter to install Anaconda3 into the default directory. Once it starts installing, you'll see all the packages included in Anaconda3 also being installed. Anaconda3 gives you everything you'll need to get your Jupyter Notebook running. At the end you'll be prompted to include Anaconda3 into your .bashrc PATH. Make sure to type "yes" (see below).

```
installing: zlib-1.2.8-3 ...
installing: anaconda-4.4.0-np112py36_0 ...
```

```
installing: conda-4.3.21-py36_0 ...
installing: conda-env-2.6.0-0 ...
Python 3.6.1 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/ec2-user/.bashrc ? [yes|no]
[no] >>> yes

Prepending PATH=/home/ec2-user/anaconda3/bin to PATH in /home/ec2-user/.bashrc
A backup will be made to: /home/ec2-user/.bashrc-anaconda3.bak


For this change to become active, you have to open a new terminal.

Thank you for installing Anaconda3!

Share your notebooks and packages on Anaconda Cloud!
Sign up for free: https://anaconda.org

[ec2-user@ip-172-31-3-10 ~]$
```

IF YOU TYPED YES TO ADD THE PATH (as instructed) THEN SKIP TO STEP 9

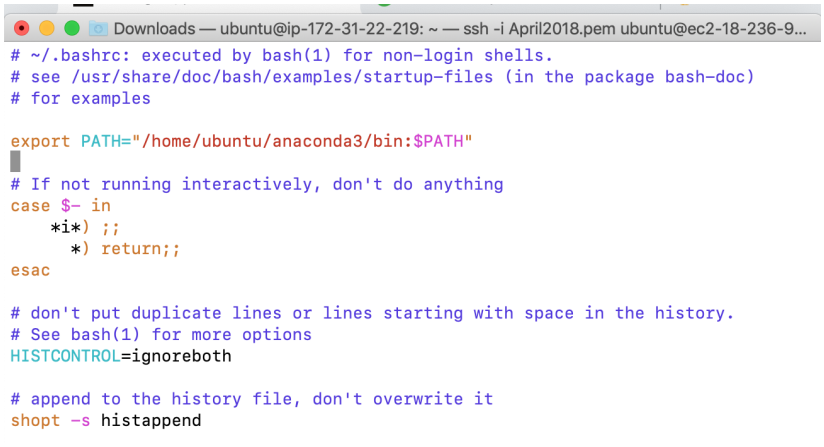
If you accidentally hit enter before typing “yes”, it will default to “no.” To correct this you’ll have to manually type the PATH into your .bashrc file. To do this type:

```
vim .bashrc
```

Your screen will now be in vim mode. If you’ve never used vim before you might feel like you’ve entered an alternate reality. Basically, yes. Vim is a text editor for your terminal and is one of the greatest tools to learn and master is you plan on doing a significant amount of hacking. But let’s focus on the task ahead of us! You need to add the text:

```
export PATH="/home/ubuntu/anaconda3/bin:$PATH"
```

First you’ll need to get to the bottom of the file. This can be achieved in a number of ways, but the most straight forward is to just hit the down arrow key until you ready the bottom (I encourage you to learn more efficient ways to getting to the bottom of a file outside of this tutorial). Now, just hit the “i” button and you’ll be in INSERT mode (think EDIT mode) which will allow you type and edit the file and you can also copy and paste with COMMAND-c and COMMAND-v. After typing in the path your file should look similar to this:



To save your edits and exit out of vim hit ESC (this take you out of INSERT mode) and then type ":wq" (stands for write-quit) and hit ENTER which will bring your back to your EC2 command line. If you fuck this up, no worries. You can exit vim anytime without saving by ESC-ing out of whatever mode you're in and typing ":q!" and then just vim back in. Good luck!

9. Set Anaconda3 as your default Python environment. You'll notice that your EC2 instance is configured to use the system's Python 2.7. This is fine if you want to use 2.7 for all your projects, but you really should be using Python 3 for future projects which is why we installed Anaconda3 so we can switch to using the latest stable version of Python 3 instead of the default 2.7. To switch your environment to use Python 3 type the command type:

```
source .bashrc
```

Now if you type "python" you'll see that you're using Python 3.7.1 (default, Dec 14 2018, 19:28:38) [GCC 7.3.0] :: Anaconda, Inc. on linux.

10. Create your ssh config file:
From your home directory on your local computer type:

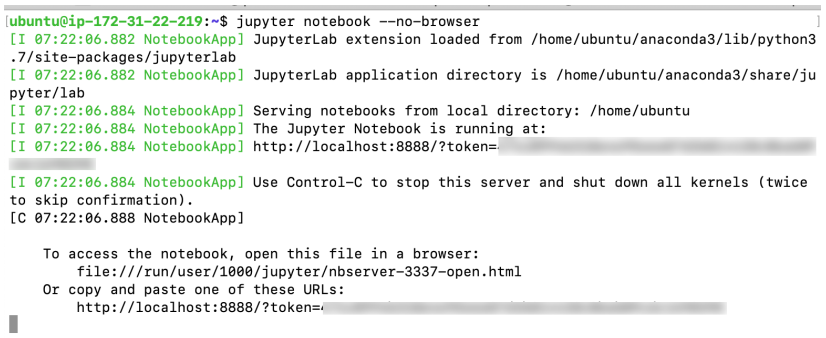
```
vim .ssh/config
```

That command will create an empty config file for you to edit. To add text to your config file, hit the "i" button and enter the following text:

```
Host ec2
    Hostname your-ec2's-public-ip-address here
    User ubuntu
    IdentityFile ~/tutorialexample.pem
```

Now hit ESC and type `:wq` to save your config file and exit the vim editor

11. Run jupyter notebook or lab on your ec2 instance:
Go to your ec2's command line and type: `jupyter notebook --no-browser` or `jupyter lab --no-browser` and you should see the notebook start up like below:



12. Connect your local port to your ec2's notebook port thru SSH
In your local computer's CLI type:

```
ssh -NfL 9999:localhost:8888 ec2
```

In essence this maps your port 9999 to the ec2's notebook port (8888 by default) via SSH. Now all you need to do is paste the URL from when you ran `jupyter notebook` on your ec2's CLI into your local computer's browser (I use Chrome and it seems to work with no issues) and change the 8888 to 9999 and hit enter to navigate to the notebook's web interface. I mapped the local port 9999 instead of 8888 just in case your local machine is already using 8888 for an existing notebook.

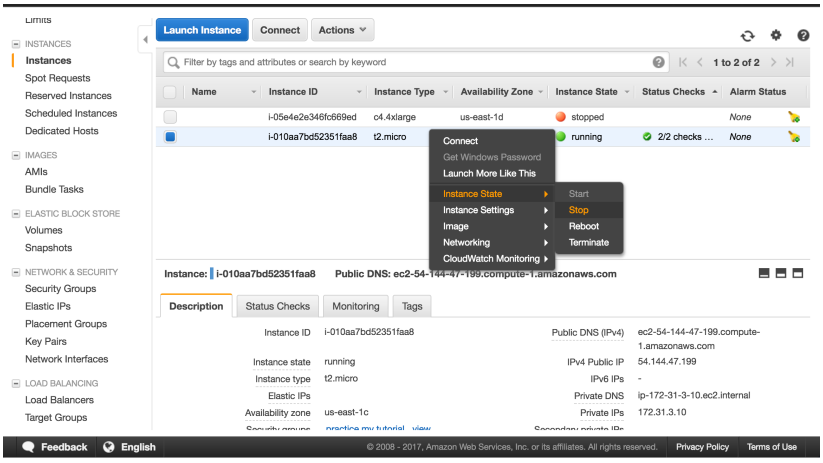
12. *HIGH FIVE* Congrats, you can now harness the power of AWS to run your Python 3 code!

13. A couple more things! If you want to easily transport files from your ec2 instance to your local computer, I highly recommend using Jupyter Lab which will allow you to download files from your ec2 instance to your local machine via the file navigator window.

Also, you may notice that if you lose your internet connection, it'll close your jupyter notebook or lab instance running on EC2. To prevent this, it is highly recommended that you use tmux with your EC2 CLI. It's a little out of scope for this quick tutorial, but there are a lot of good resources online if you just google "tmux tutorial."

Important housekeeping:

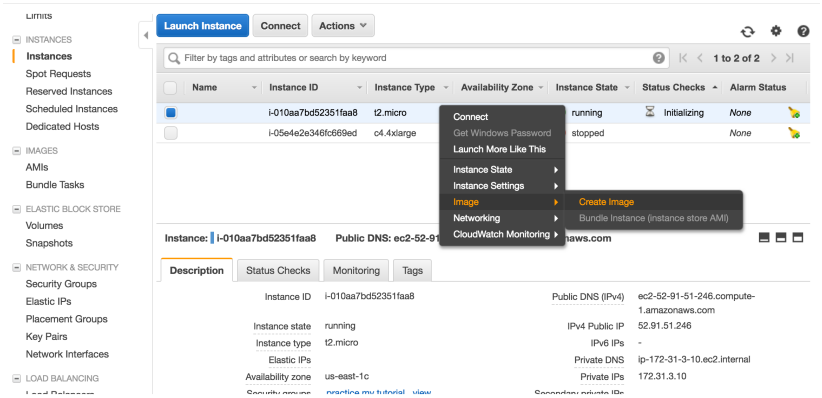
If you don't know all the INs and OUTs of AWS it'll be good to know the following info. When you're done working on your EC2 instance you should stop it to prevent being charged for time not using it. To do this just go to the AWS console then to EC2 and click instances on the sidebar:



Once you stop your instance you'll no longer be charged for using it with one caveat: all the installations you did and files created will be saved for you since you're using an instance with EBS storage. AWS charges you for keeping this data for the next time you start up an instance. The cost depends on how much data is saved. Most instances with EBS will store your data on a EBS General Purpose SSD (gp2) Volume which as of June 2017 has a monthly cost of $0.10 per GB. So not bad if you're only storing

code. Go here to check pricing for storing data: https://aws.amazon.com/ebs/pricing/

Let's say you want to run another instance using the same data/configuration. What you'll want to do is create an AMI from your instance. Then you can even delete your instances, but still have all the work and files you created saved for future use. This is incredibly easy:



Now under "AMIs" on the side bar, you'll have everything saved and can launch new EC2 instances directly from the AMI. AMIs can also be saved in S3 for much less money by following these steps.

When you want to start your instance again, just go to the Instance State and select Start. When you start an instance you get a new Public DNS address so make sure to use the new address when you SSH in.

And lastly, when running computationally expense code, you'll probably want to use a more powerful type of instance than the t.2 micro we started with. Once your instance is stopped you just need to right-click on it and under "Instance Settings" choose "Change Instance Type." Make sure you're aware of the costs of using your new instance type (https://aws.amazon.com/ec2/pricing/on-demand/) and that your code makes the most of multi-core processors (i.e. if your code isn't set to use multi-core functionality in Python, then having 64 cores will run the same as 1 core).

Hope this was useful! Let me know if I made any mistakes or if something something was particularly obscure.

AWS