labibi Documentation

Release 1.0

C. Titus Brown

October 21, 2016

Contents

This workshop was given on October 12th and 13th, 2016, by Harriet Alexander and C. Titus Brown, at the Scripps Institute of Oceanography.

For more information, please contact Titus directly.

The workshop was recorded (although the recording isn't always very good, sorry!) You can view it here:

- Day 1, morning
- day 1, afternoon
- day 2, morning
- day 2, afternoon

Tutorials:

CHAPTER 1

Welcome!

1.1 1. Learning goals

For you:

- get a first (or second) look at tools;
- gain some experience in the basic command line;
- get 80% of way to a complete analysis of some data;
- introduction to philosophy and perspective of data analysis in science;

1.2 2. Safe space and code of conduct

This is intended to be a safe and friendly place for learning!

Please see the Software Carpentry workshop Code of Conduct: http://software-carpentry.org/conduct.html

In particular, please ask questions, because I guarantee you that your question will help others!

1.3 3. Instructor introductions

Harriet Alexander - postdoc at UC Davis.

Titus Brown - prof at UC Davis in the School of Vet Med.

1.4 4. Amazon and cloud computing - why?!

- simplifies software installation;
- can be used for bigger analyses quite easily;
- good for "burst" capacity (just got a data set!)
- accessible everywhere;

1.5 5. Sticky notes and how they work... + Minute Cards

Basic rules:

- no sticky note - "working on it"
- green sticky note - "all is well"
- red sticky note - "need help!"

Place the sticky notes where we can see them from the back of the room – e.g. on the back of your laptop.

At the end of each session (coffee break, lunch, end of day) please write down on an index card **one thing you learned** and **one thing you're still confused about.**

—

Next: `n-overview`

CHAPTER 2

Starting up an Amazon Web Services machine

2.1 Start here: *Start an Amazon Web Services computer:*

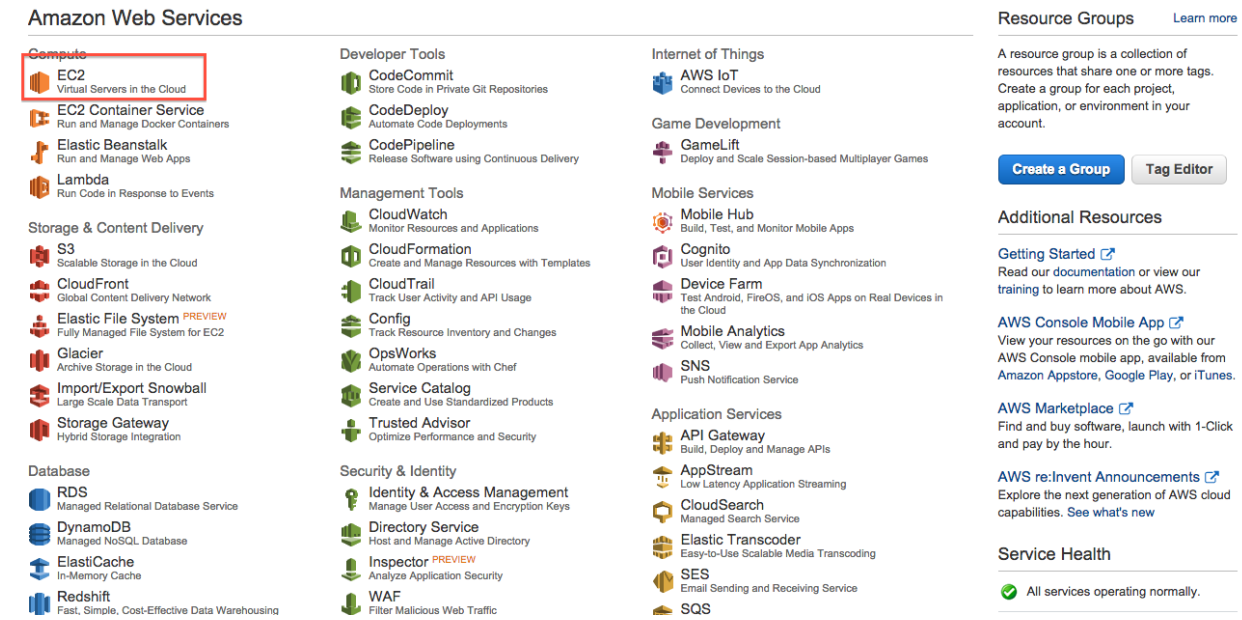
2.2 Full table of contents:

2.2.1 Start an Amazon Web Services computer:

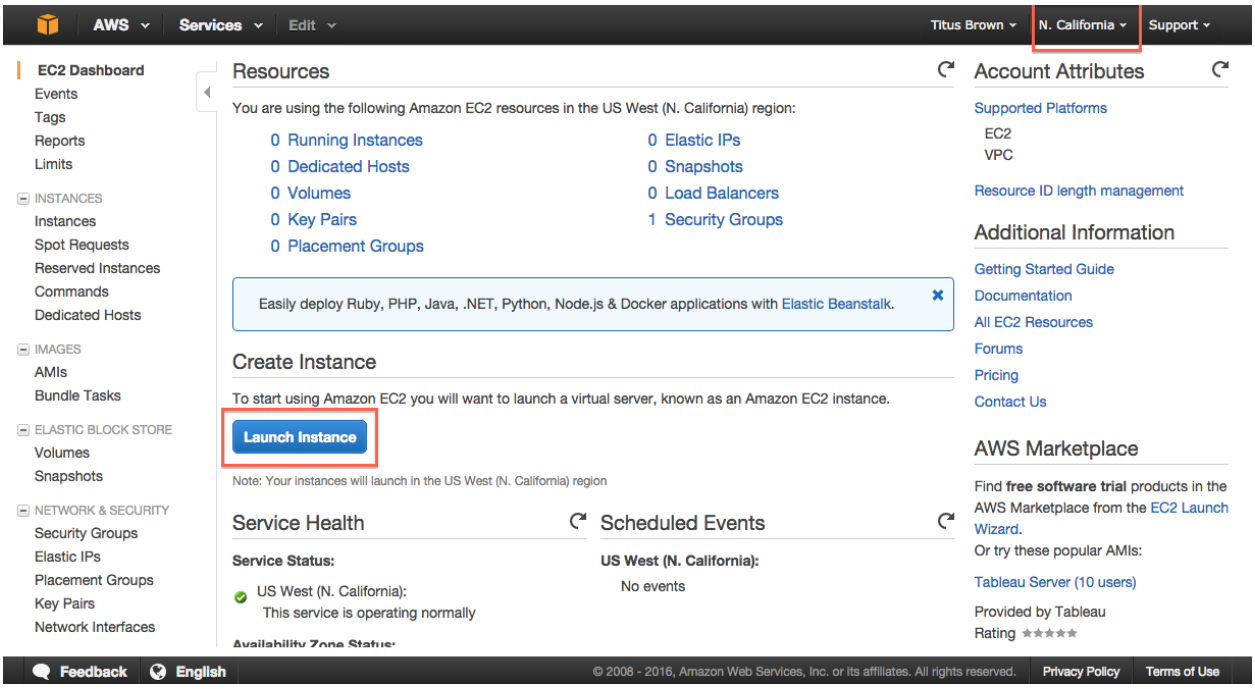
This page shows you how to create a new "AWS instance", or a running computer.

Start at the Amazon Web Services console (http://aws.amazon.com/ and sign in to the console).

0. Select "EC2 - virtual servers in the cloud"

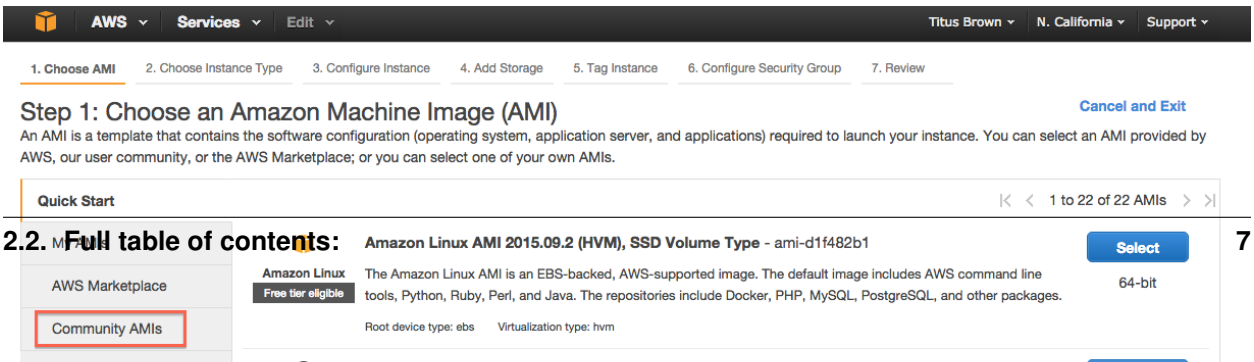


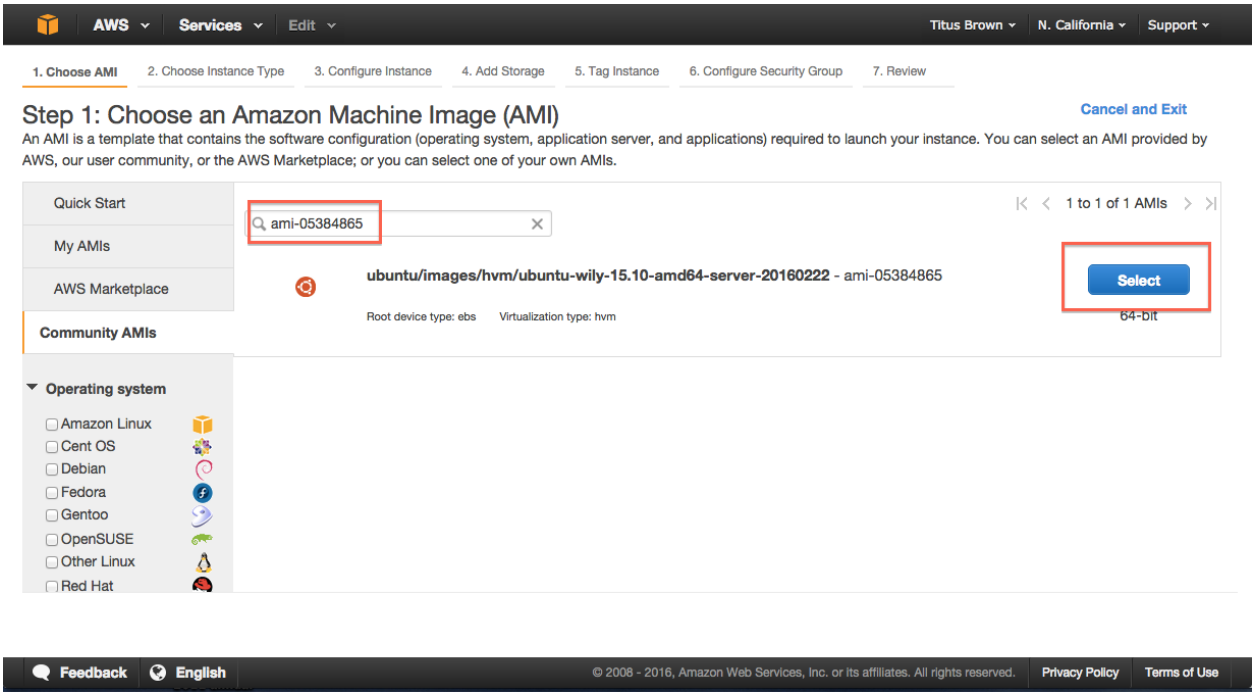
1. Switch to zone US West (N California)



2. Click on "Launch instance."

3. Select "Community AMIs."

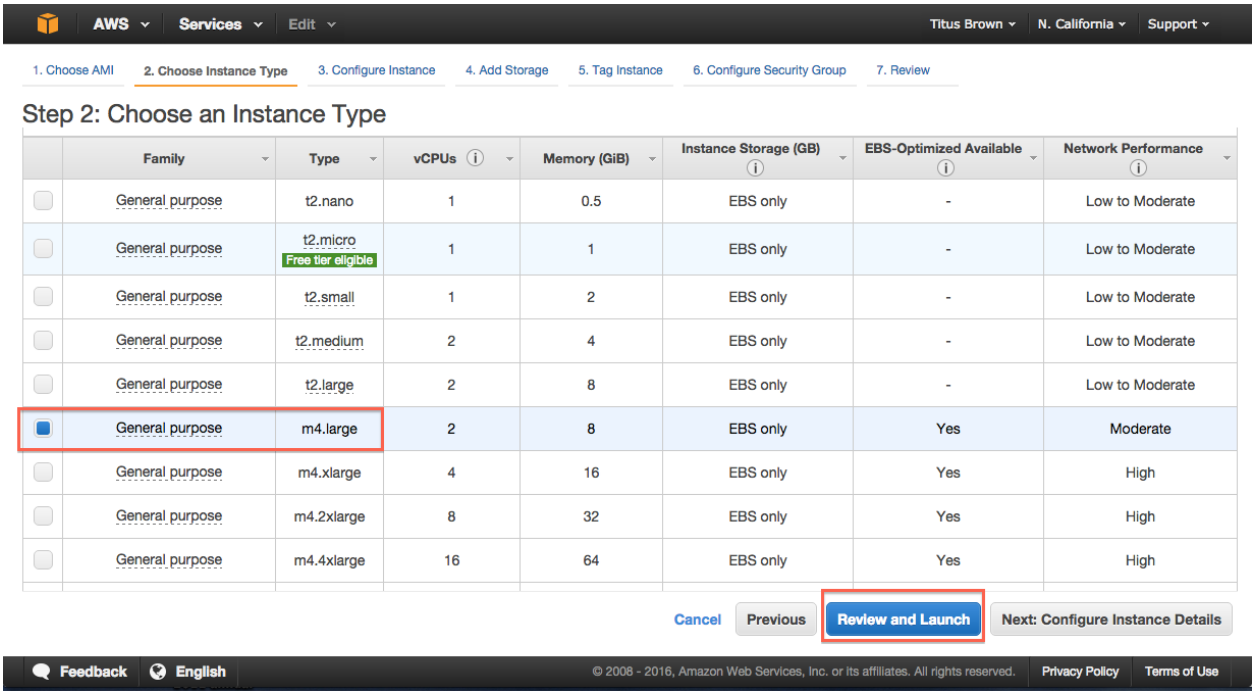



AWS
Services
Edit
Titus Brown
N. California
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
Quick Start
My AMIs
AWS Marketplace
Community AMIs
ami-05384865
1 to 1 of 1 AMIs
ubuntu/images/hvm/ubuntu-wily-15.10-amd64-server-20160222 - ami-05384865
Root device type: ebs
Virtualization type: hvm
Select
64-bit
Operating system
Amazon Linux
Cent OS
Debian
Fedora
Gentoo
OpenSUSE
Other Linux
Red Hat

Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

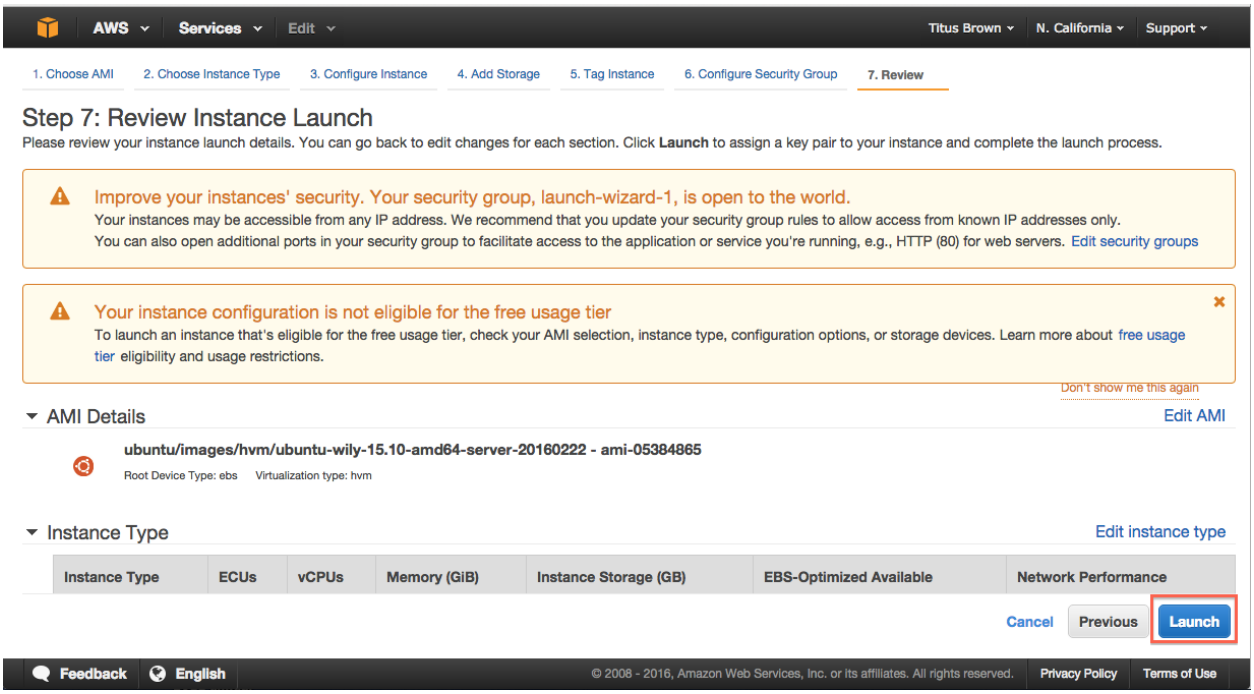
5. Click on "Select."

6. Choose m4.xlarge.



7. Click "Review and Launch."

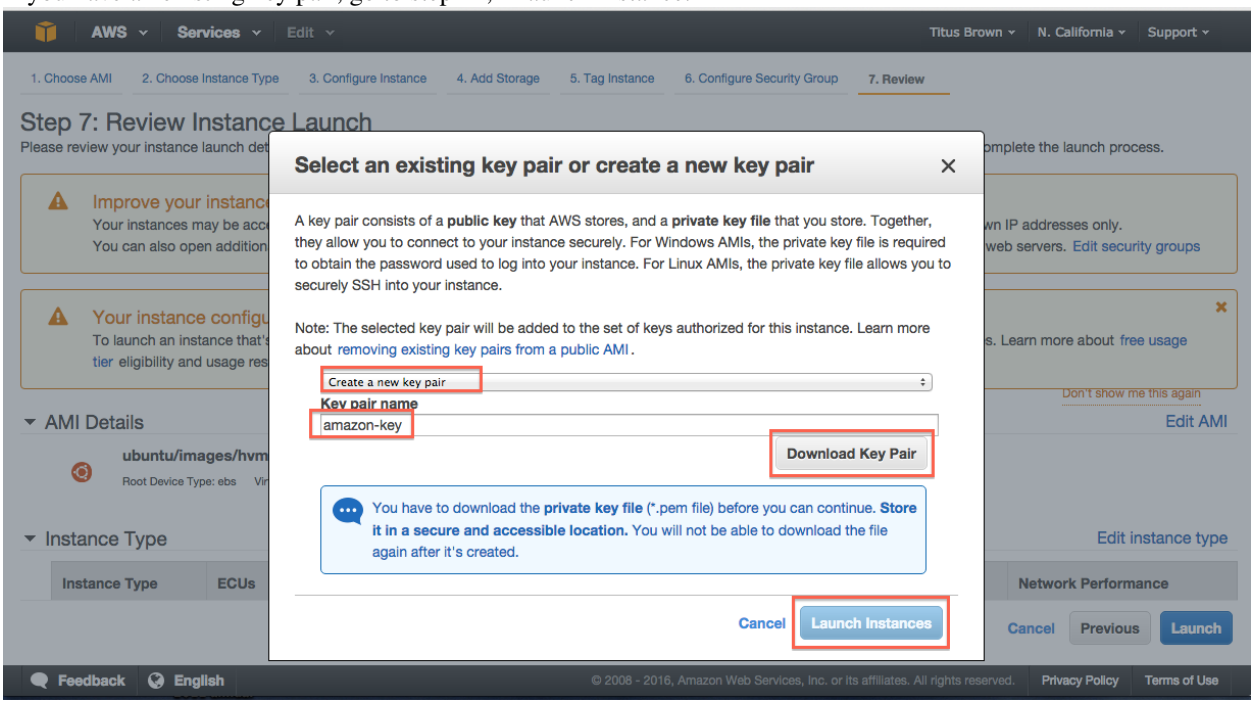
8. Click "Launch."



9. Select "Create a new key pair."

Note: you only need to do this the first time you create an instance. If you know where your amazon-key.pem file is, you can select 'Use an existing key pair' here. But you can always create a new key pair if you want, too.

If you have an existing key pair, go to step 12, “Launch instance.”

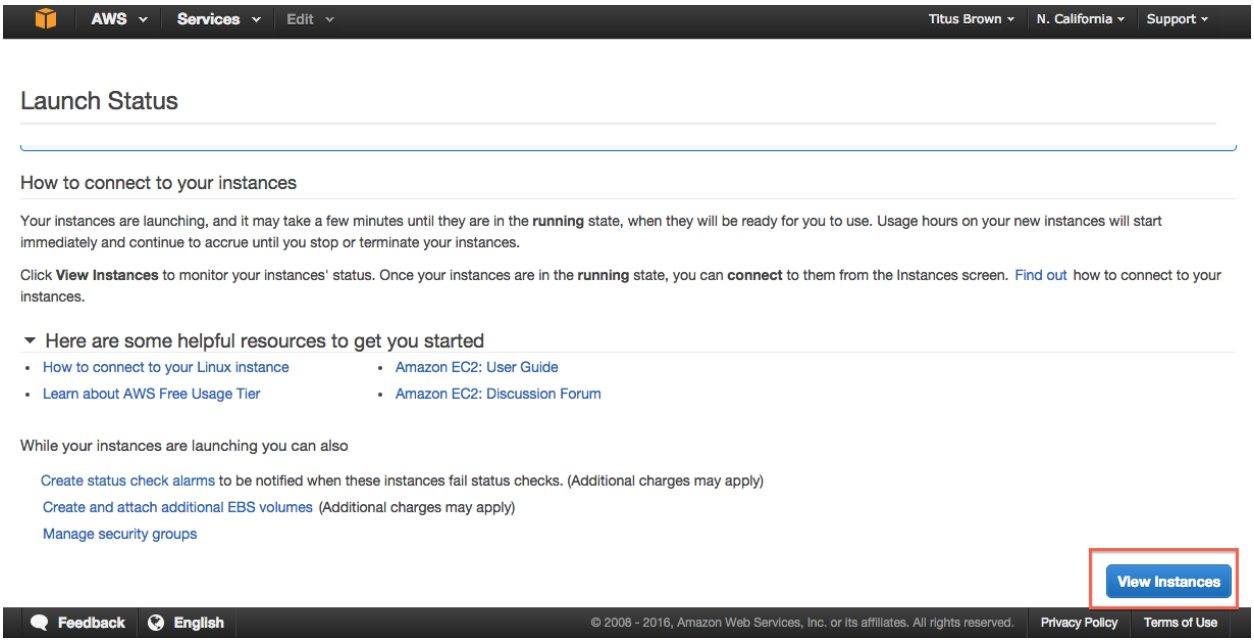


10. Enter name 'amazon-key'.

11. Click "Download key pair."

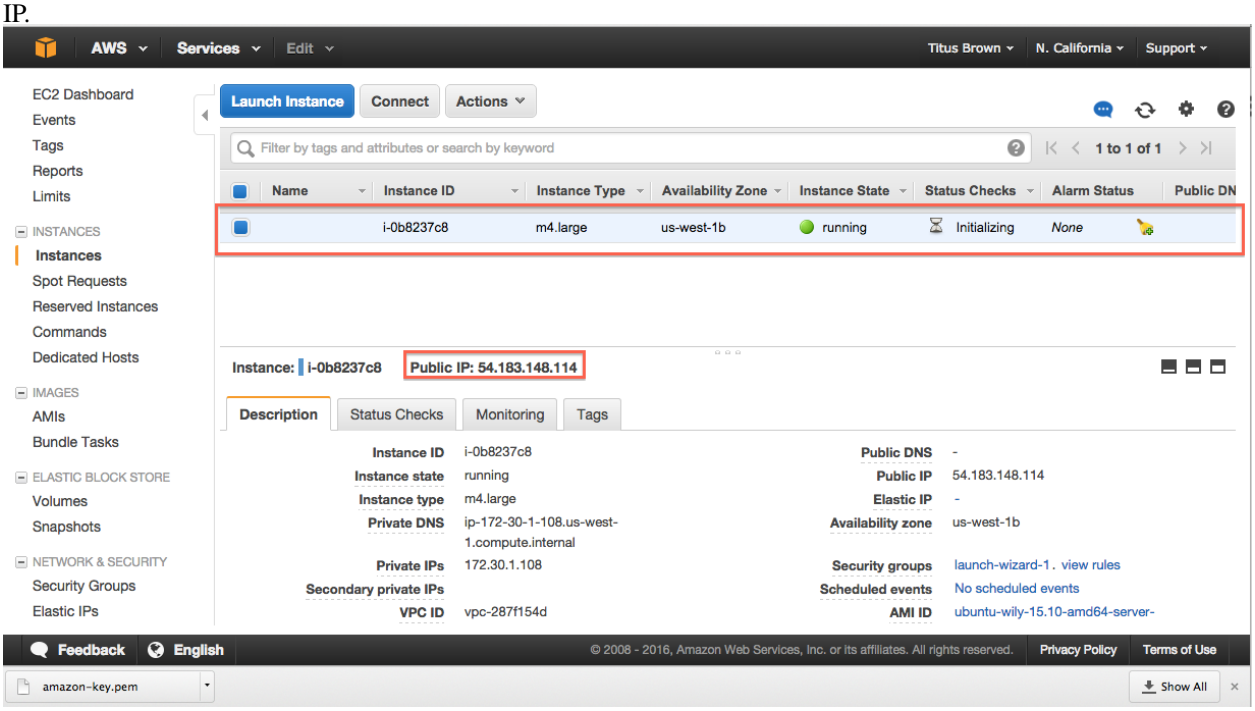
12. Click "Launch instance."

13. Select View instances (lower right)



14. Bask in the glory of your running instance

Note that for your instance name you can use either "Public IP" or "Public DNS". Here, the machine only has a public IP.



You can now Log into your instance with the UNIX shell or Configure your instance firewall.

2.2.2 Log into your instance with the UNIX shell

You will need the `amazon-key.pem` file that was downloaded in step 11 of booting up your new instance (see Start an Amazon Web Services computer:).

Then, you can either Log into your instance from a Mac or Linux machine or Log into your instance from a Windows machine.

Log into your instance via the UNIX shell (Mac/Linux)

See: Log into your instance from a Mac or Linux machine

Log into your instance via MobaXTerm (Windows)

See: Log into your instance from a Windows machine

Logging in is the starting point for most of the follow-on tutorials. For example, you can now install and run software on your EC2 instance.

Go back to the top page to continue: Starting up an Amazon Web Services machine

2.2.3 Log into your instance from a Mac or Linux machine

You'll need to do two things: first, set the permissions on `amazon-key.pem`:

```
chmod og-rwx ~/Downloads/amazon-key.pem
```

Then, ssh into your new machine using your key:

```
ssh -i ~/Downloads/amazon-key.pem -l ubuntu MACHINE_NAE
```

where you should replace MACHINE_NAME with the public IP or hostname of your EC2 instance, which is located at the top of the host information box (see screenshot below). It should be something like `54.183.148.114` or `ec2-XXX-YYY.amazonaws.com`.

Here are some screenshots!

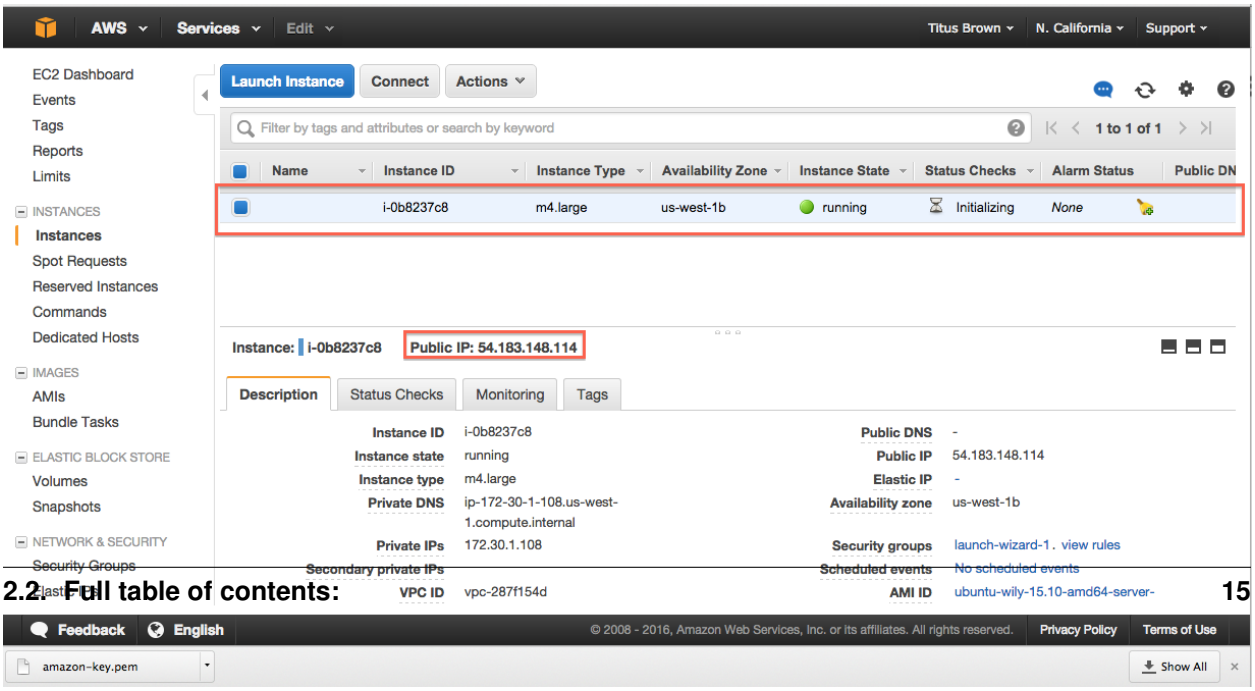
Change permissions and execute ssh

```
% chmod og-rwx ~/Downloads/amazon-key.pem
% ssh -i ~/Downloads/amazon-key.pem ubuntu@54.183.148.114
The authenticity of host '54.183.148.114 (54.183.148.114)' can't be established.
RSA key fingerprint is b6:de:2f:fb:e7:12:e5:1e:5d:66:37:ef:40:bb:b7:c8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.183.148.114' (RSA) to the list of known hosts.
```

Successful login

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.183.148.114' (RSA) to the list of known hosts.
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  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.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-30-1-108:~$
```

Host information box - MACHINE_NAME location



Logging in is the starting point for most of the follow-on tutorials. For example, you can now install and run software on your EC2 instance.

Go back to the top page to continue: Starting up an Amazon Web Services machine

2.2.4 Log into your instance from a Windows machine

Go follow the instructions this URL:

https://angus.readthedocs.org/en/2015/amazon/log-in-with-mobaxterm-win.html

Logging in is the starting point for most of the follow-on tutorials. For example, you can now install and run software on your EC2 instance.

Go back to the top page to continue: Starting up an Amazon Web Services machine

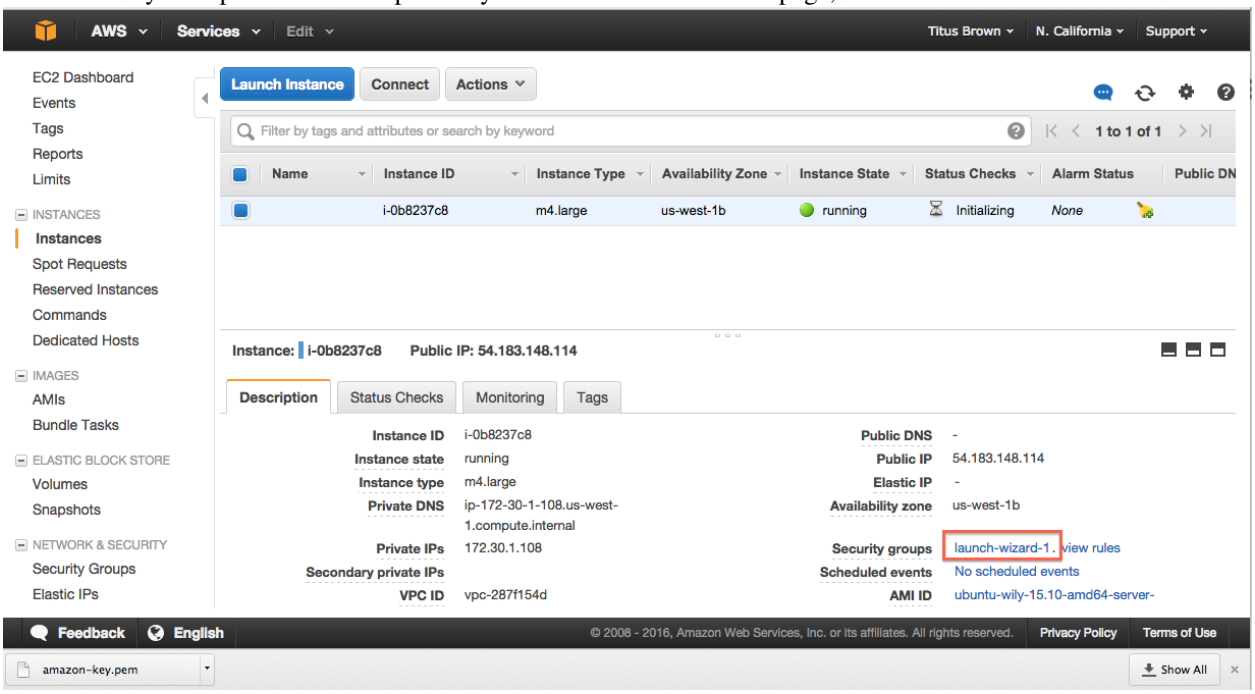
2.2.5 Configure your instance firewall

Normally, Amazon computers only allow shell logins via ssh (port 22 access). If we want to run a Web service or something else, we need to give the outside world access to other network locations on the computer.

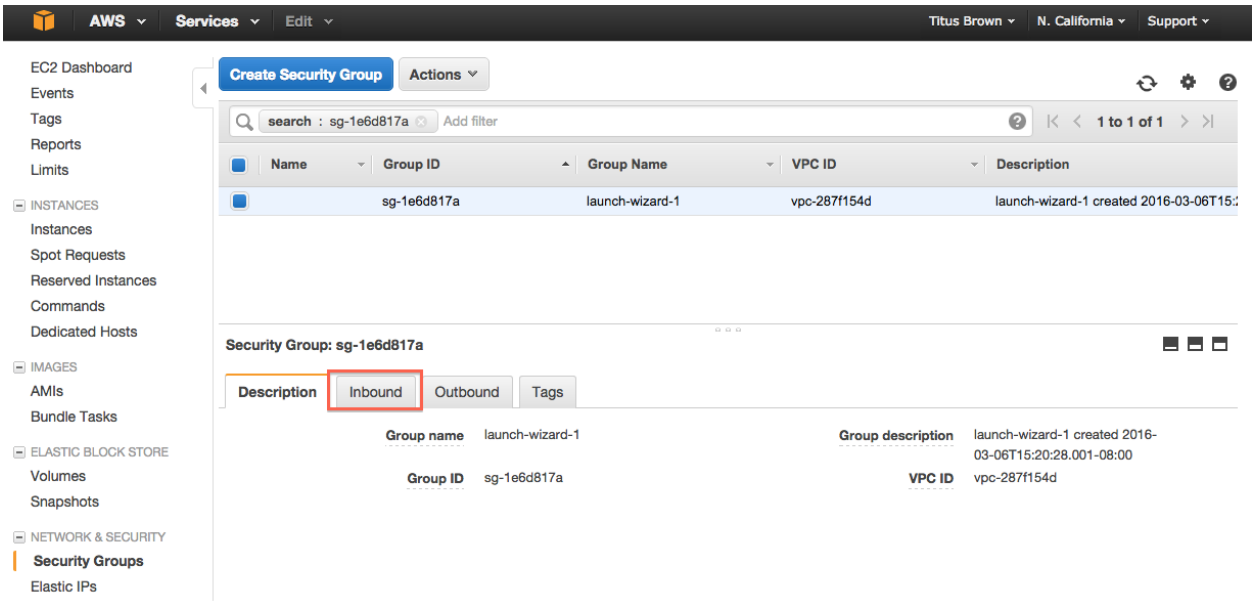
Below, we will open ports 8000-9000, which will let us run things like RStudio Server. If you want to run other things, like a Web server, you'll need to find the port(s) associated with those services and open those instead of 8000-9000. (Tip: Web servers run on port 80.)

1. Select 'Security Groups'

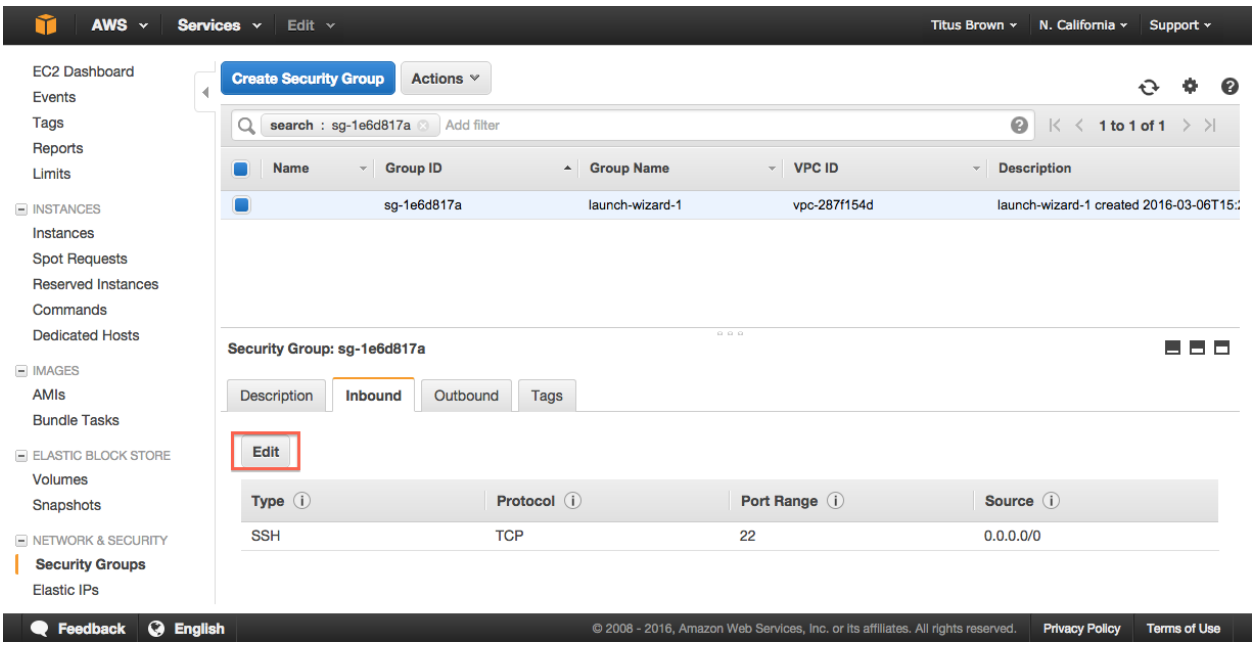
Find "Security Groups" in the lower pane of your instance's information page, and click on "launch-wizard-N".



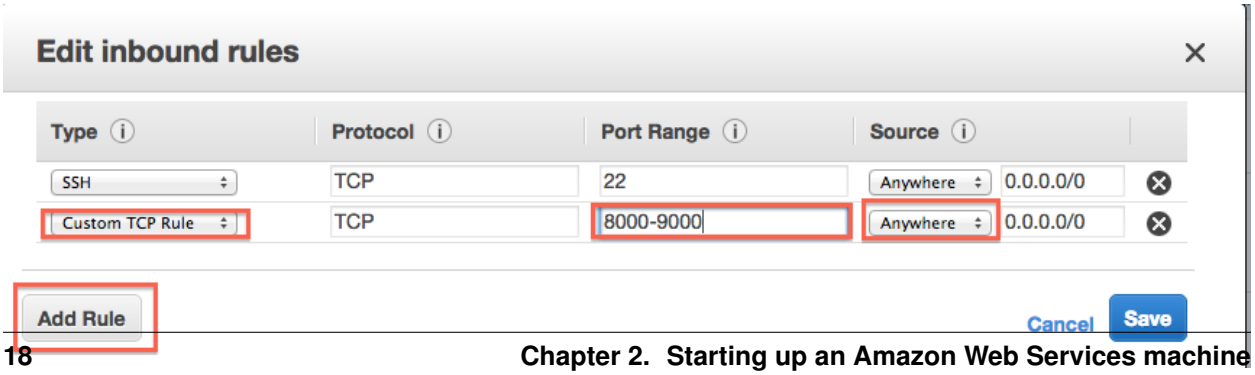
2. Select 'Inbound'



3. Select 'Edit'



4. Select 'Add Rule'

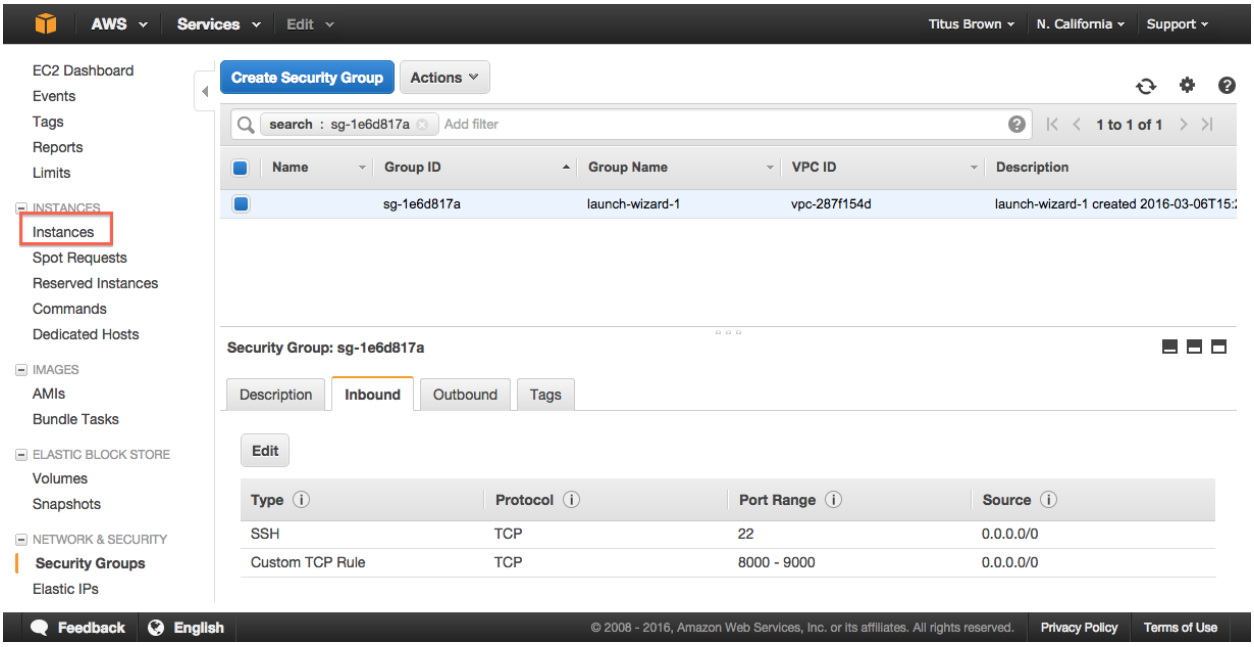


5. Enter rule information

Add a new rule: Custom TCP, 8000-9000, Source Anywhere

6. Select 'Save'.

7. Return to the Instances page.

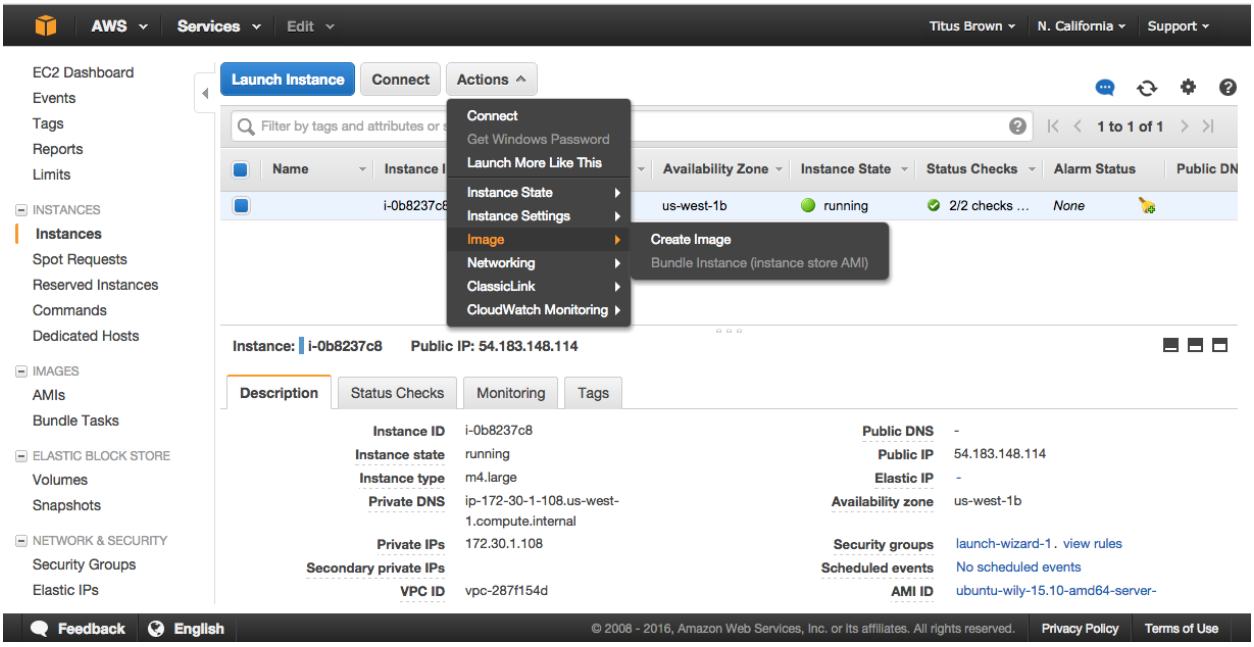


You're done!

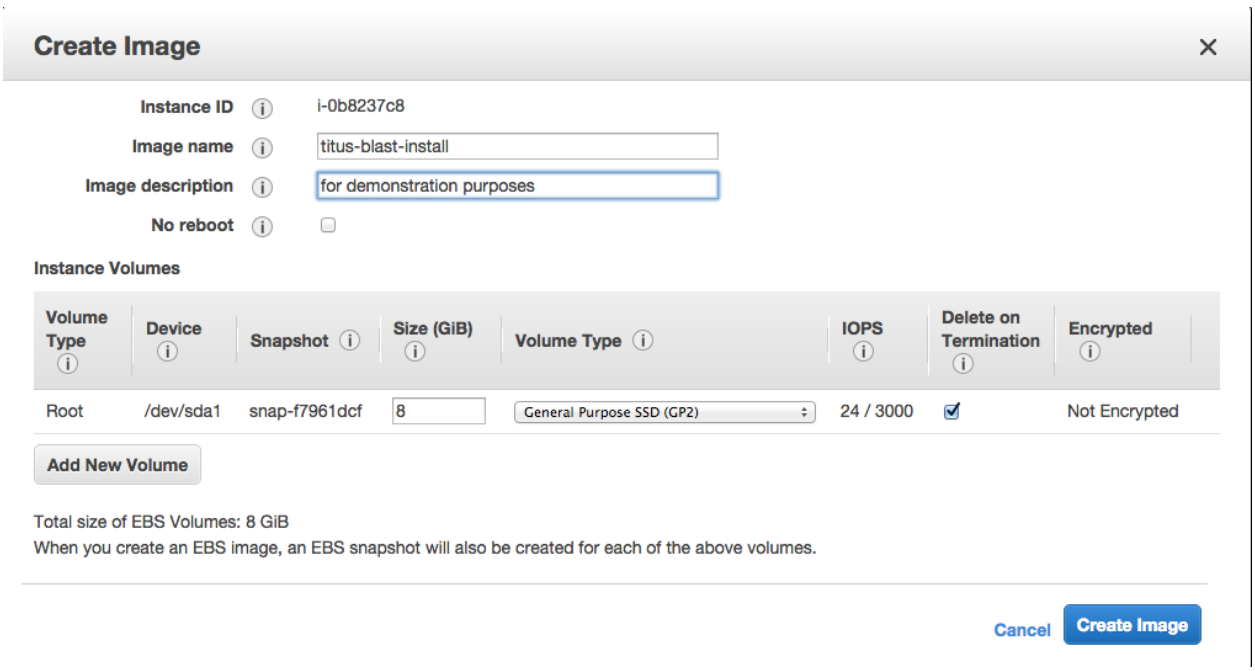
Go back to the index: Starting up an Amazon Web Services machine

2.2.6 Creating your own Amazon Machine Image

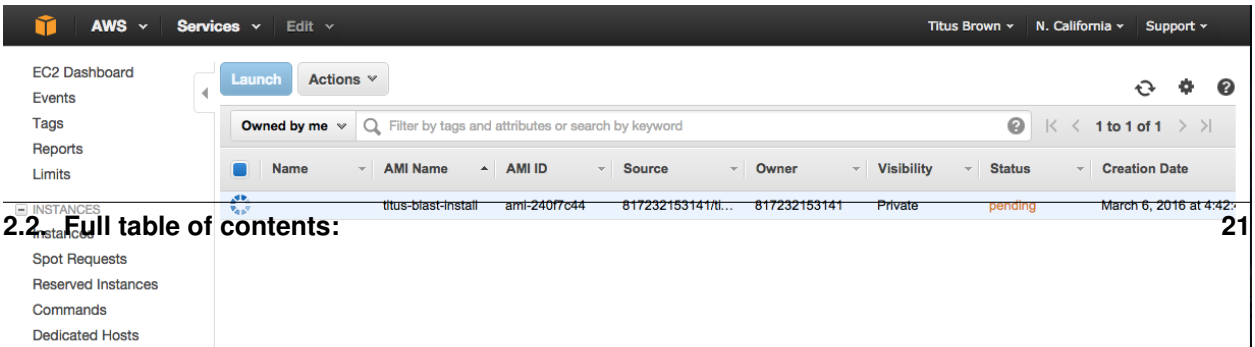
1. Actions, Create image



2. Fill out name and description



3. Wait for it to become available



Go back to the index: Starting up an Amazon Web Services machine

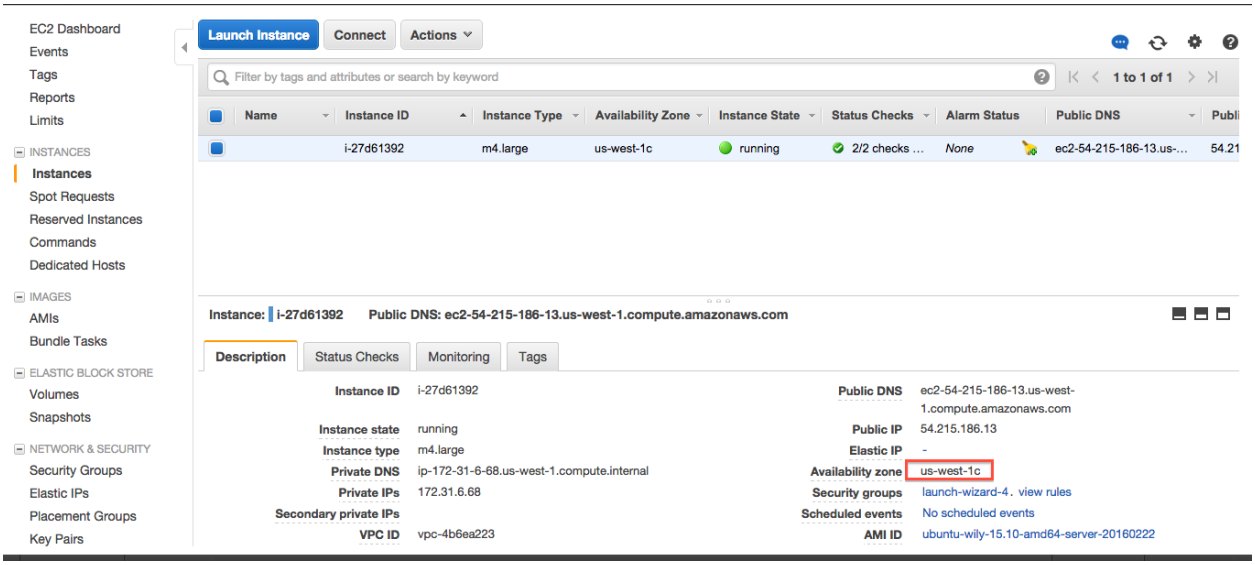
2.2.7 Working with persistent storage: volumes and snapshots

Volumes are basically UNIX disks ("block devices") that will persist after you terminate your instance. They are tied to a zone within a region and can only be mounted on instances within that zone.

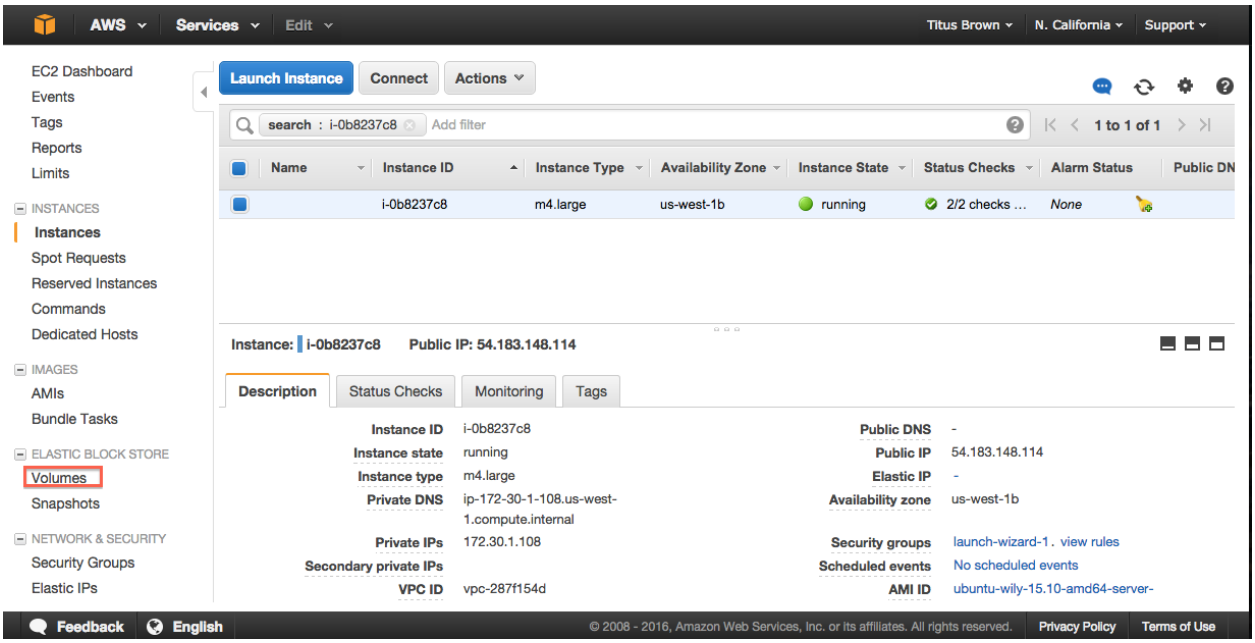
Snapshots are an Amazon-specific thing that let you communicate data on volumes between accounts. They are "read-only" backups that are created from volumes; they can be used to create new volumes in turn, and can also be shared with specific people (or made public). Snapshots are tied to a region but not a zone.

Creating persistent volumes to store data

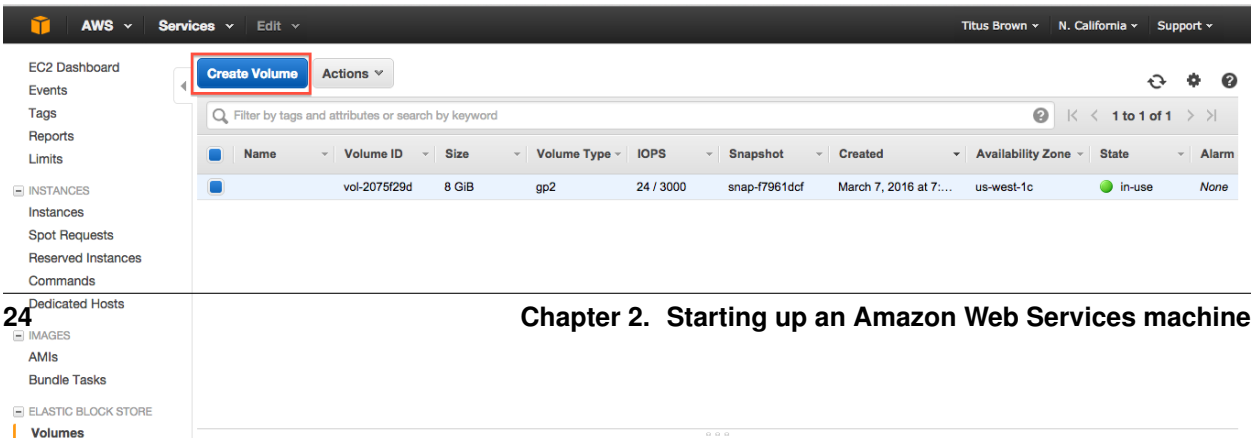
0. Locate your instance *zone*

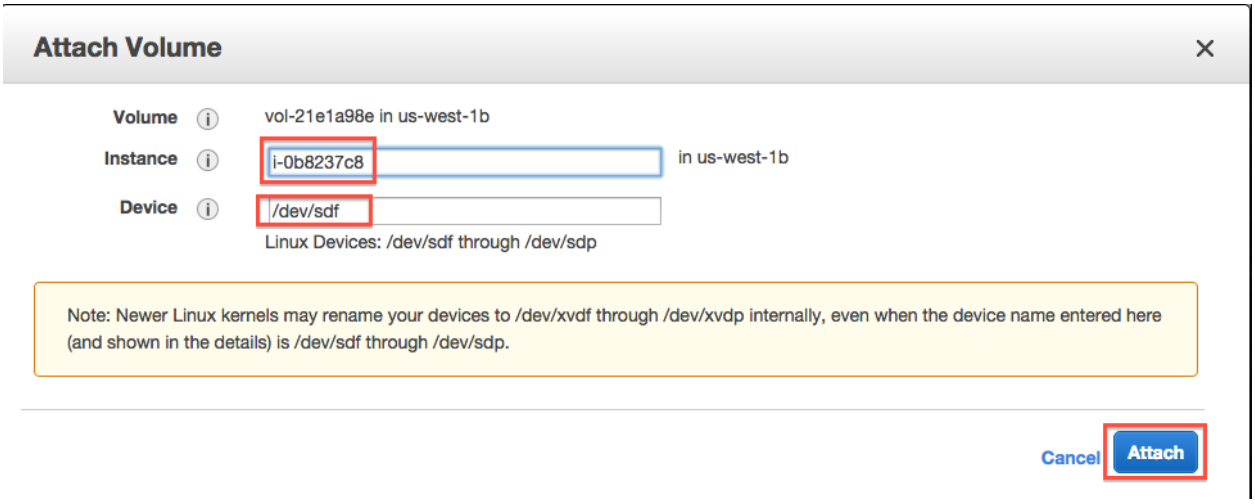


1. Click on the volumes tab



2. 'Create Volume'





7. On your instance, list block devices

Type:

```
lsblk
```

You should see something like this:

```
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda    202:0    0    8G  0 disk
`-xvda1 202:1    0    8G  0 part /
xvdf    202:80   0  100G  0 disk
```

Now format the disk (ONLY ON EMPTY DISKS - THIS WILL ERASE ANY DATA ON THE DISK):

```
sudo mkfs -t ext4 /dev/xvdf
```

and mount the disk:

```
sudo mkdir /disk
sudo mount /dev/xvdf /disk
sudo chmod a+rwxt /disk
```

and voila, anything you put on /disk will be on the volume that you allocated!

The command 'df -h' will show you what disks are actually mounted & where.

Detaching volumes

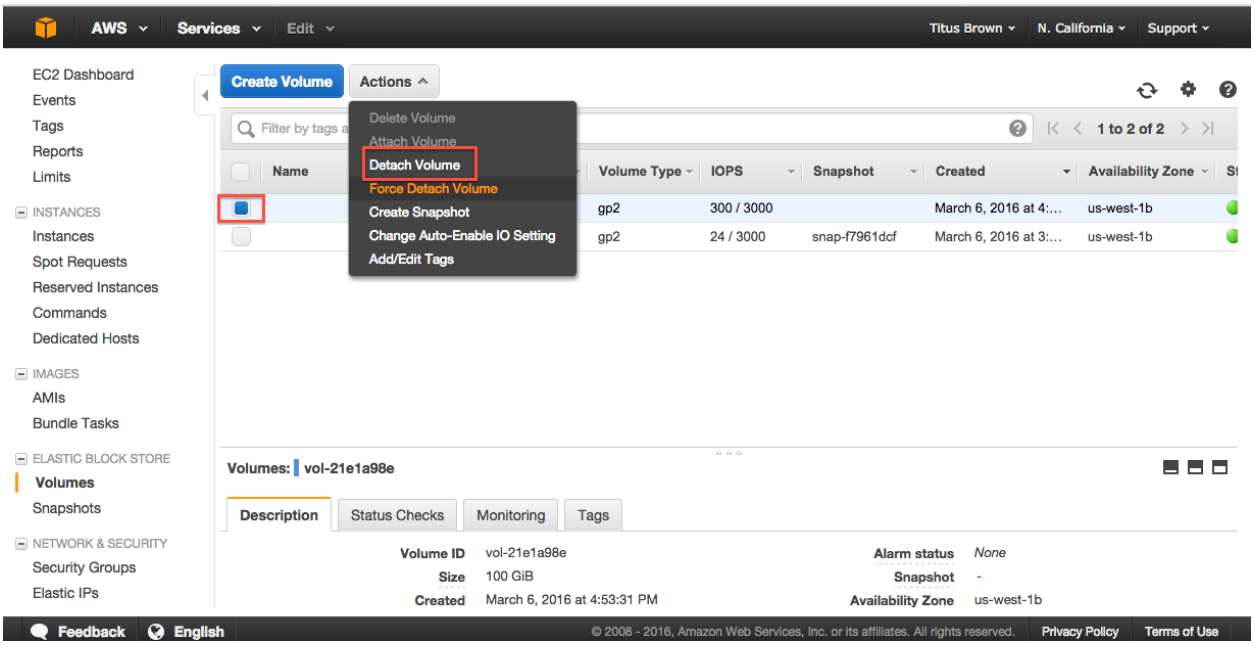
1. Unmount it from the instance

Change out of the directory, stop any running programs using it, and then:

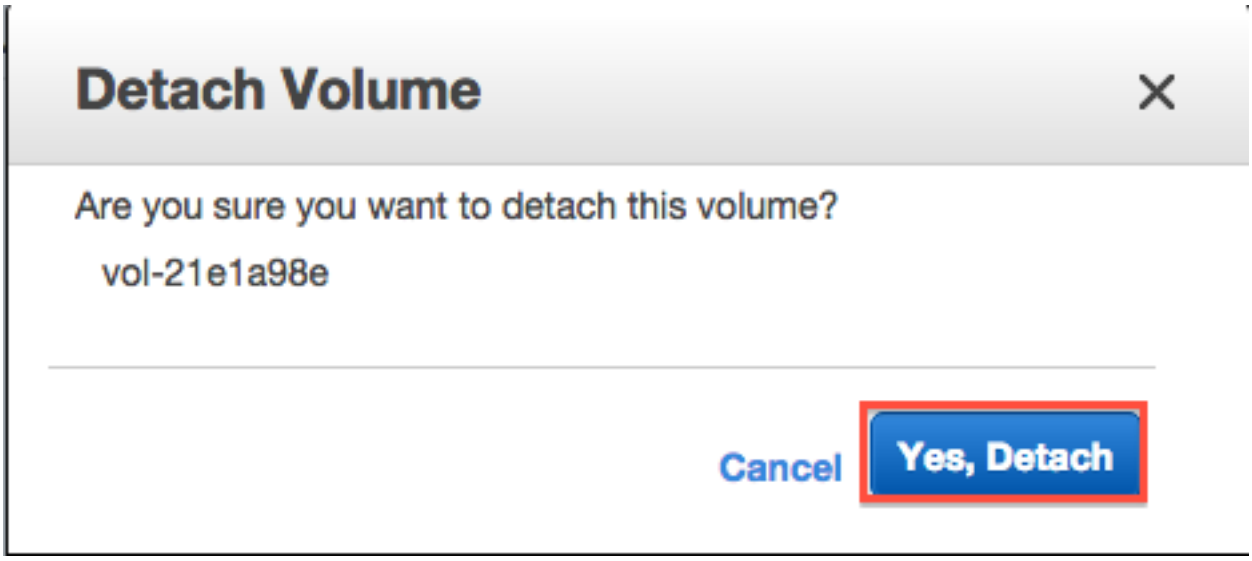
```
sudo umount /disk
```

2. Detach

On the 'volumes' tab in your EC2 console, go to Actions, Detach.



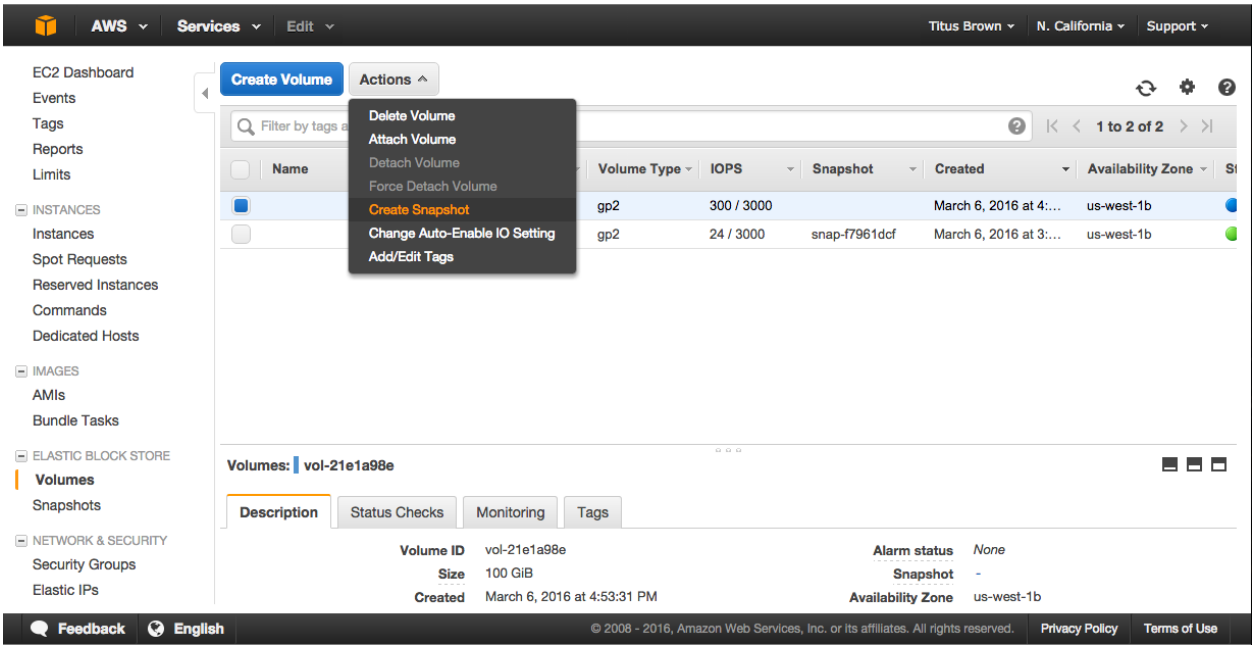
3. Yes, detach.



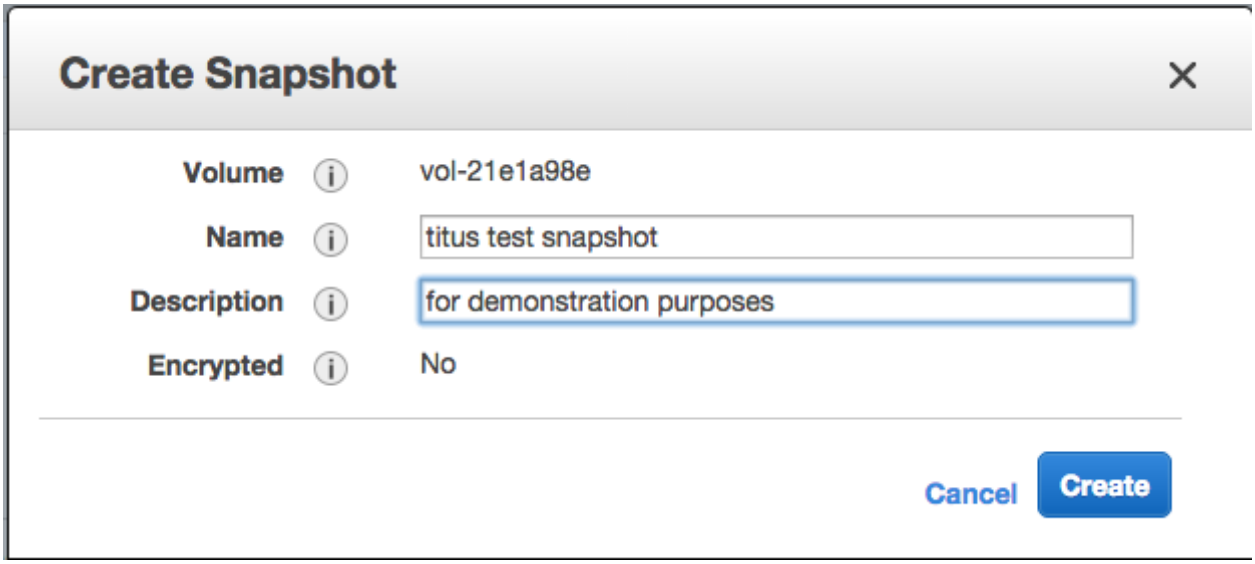
Note, volumes remain attached when you reboot or stop an instance, but are (of course) detached when you terminate an instance.

Creating snapshots of volumes

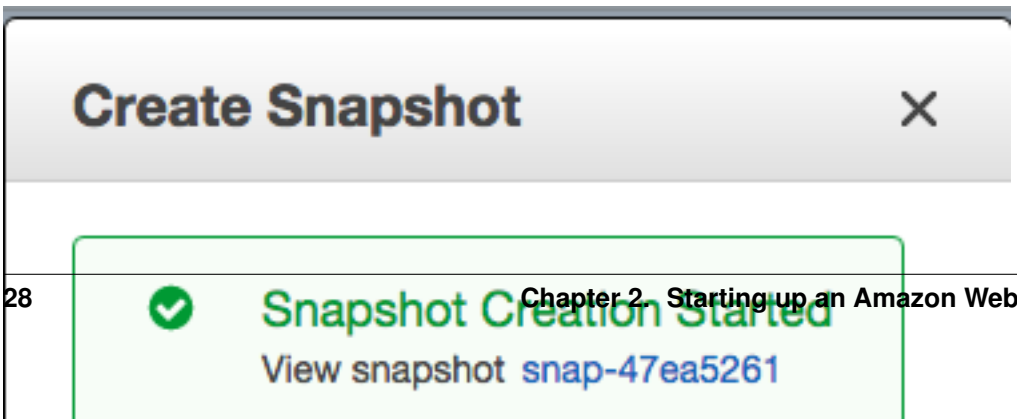
1. Actions, Create snapshot



2. Fill out name and description



3. Click 'Close' & wait.



2.2.8 Terminating your instance

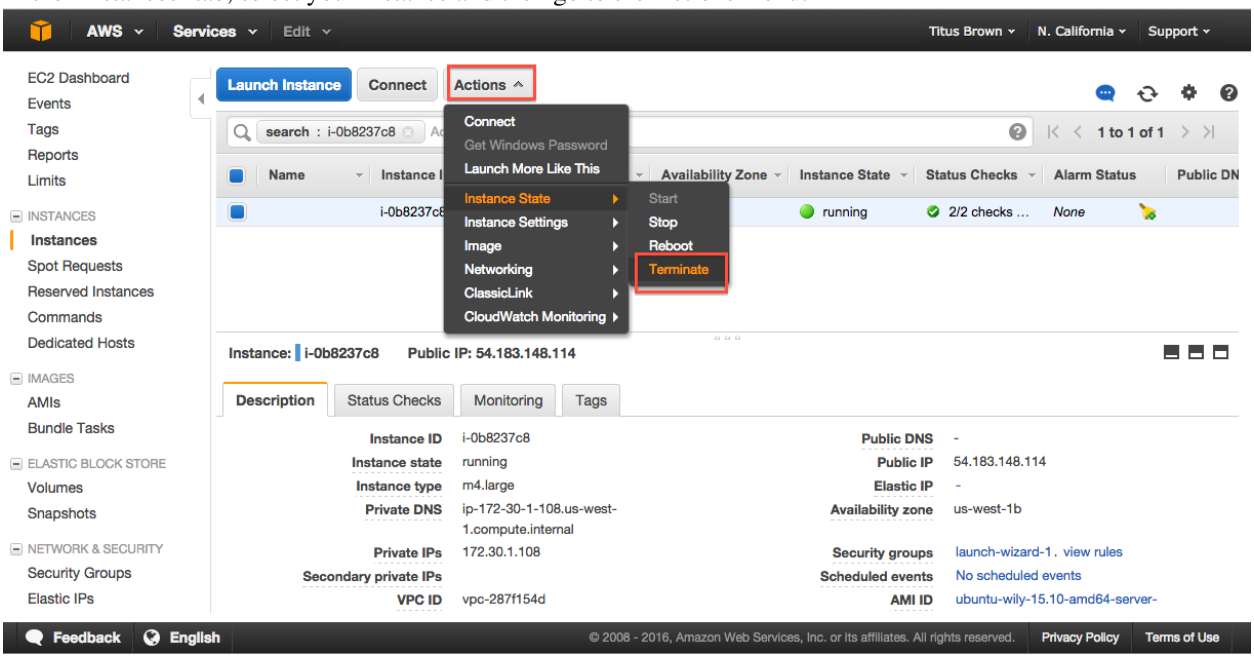
Amazon will happily charge you for running instances and/or associated ephemeral storage until the cows come home - it's your responsibility to turn things off. The Right Way to do this for running instances is to terminate.

The caveat here is that *everything ephemeral* will be deleted (excluding volumes that you created/attached). So you want to make sure you transfer off anything you care about.

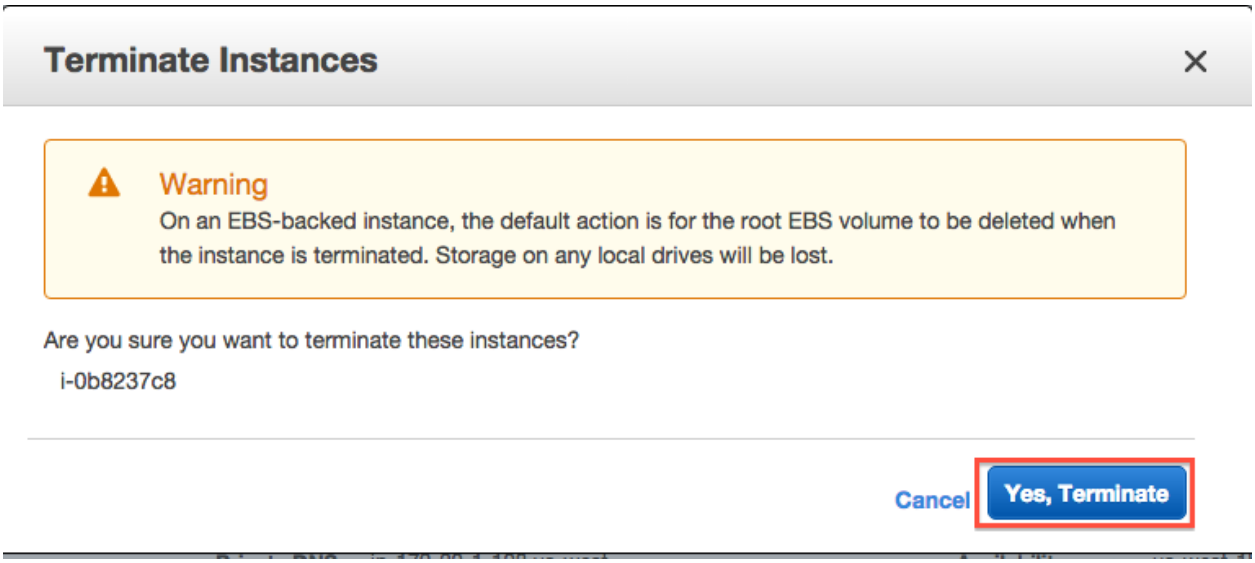
To terminate:

1. Select Actions, Instance State, Terminate

In the 'Instances' tab, select your instance and then go to the Actions menu.

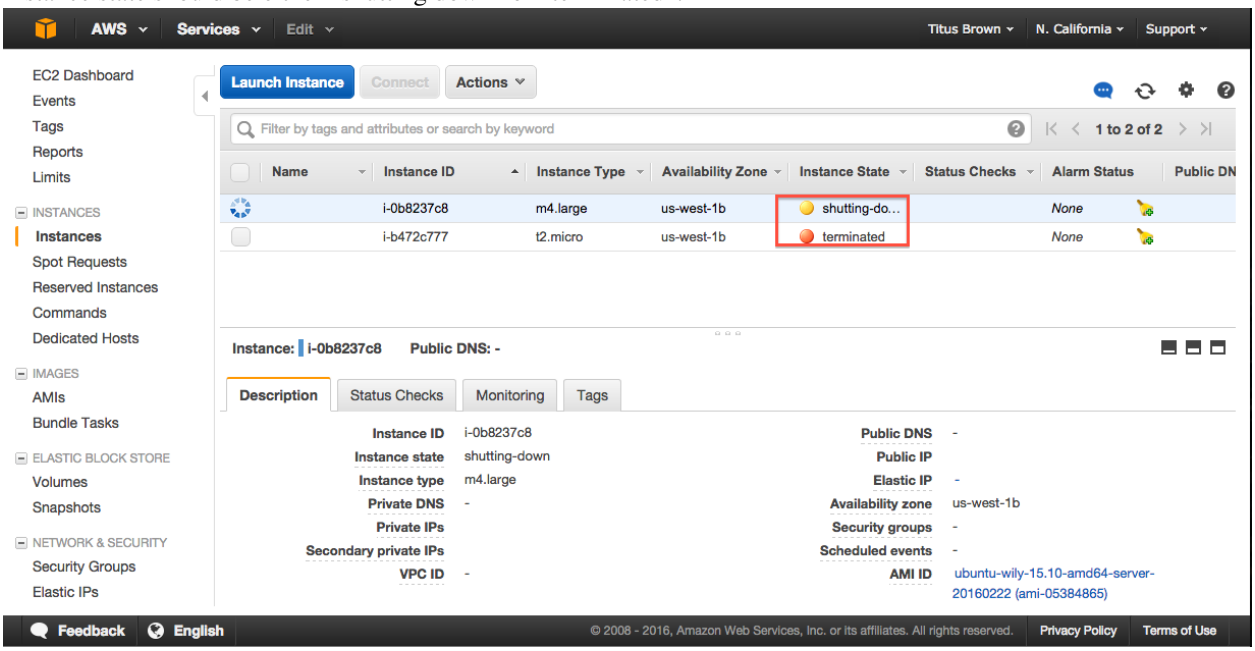


2. Agree to terminate.



3. Verify status on your instance page.

Instance state should be either "shutting down" or "terminated".



Return to index: Starting up an Amazon Web Services machine

2.2.9 Things to mention and discuss

When do disks go away?

- never on reboot;
- ephemeral disks go away on stop;

- AMI-attached volumes go away on terminate;
- attached volumes never go away on terminate and have to be explicitly deleted;
- snapshots only go away when you explicitly delete them.

What are you charged for?

- you are charged for a running instance at the @@instance price rates;
- ephemeral storage/instance-specific storage is included within that.
- when you stop an instance, you are charged at disk-space rates for the stopped disk;
- when you create a volume, you are charged for that volume until you delete it;
- when you create a snapshot, you are charged for that snapshot until you delete it.

To make sure you're not getting charged, go to your Instance view and clear all search filters; anything that is "running" or "stopped" is costing you. Also check your volumes and your snapshots - they should be empty.

Regions vs zones:

- AMIs and Snapshots (and keys and security groups) are per region;
- Volumes and instances are per zone;

2.2.10 Running RStudio Server in the cloud

In this section, we will run RStudio Server on a remote Amazon machine. This will require starting up an instance, configuring its network firewall, and installing and running some software.

Reference documentation for running RStudio Server on Ubuntu:

https://www.rstudio.com/products/rstudio/download-server/

---

1. Start up an Amazon instance

Start an ami-05384865 on an m4.xlarge machine, as per the instructions here:

Start an Amazon Web Services computer:.

2. Configure your network firewall

Normally, Amazon computers only allow shell logins via ssh. Since we want to run a Web service, we need to give the outside world access to other network locations on the computer.

Follow these instructions:

Configure your instance firewall

(You can do this while the computer is booting.)

3. Log in via the shell

Follow these instructions to log in via the shell:

Log into your instance with the UNIX shell.

4. Set a password for the 'ubuntu' account

Amazon Web Services computers normally require a key (the .pem file) instead of a login password, but RStudio Server will need us to log in with a password. So we need to configure a password for the account we're going to use (which is 'ubuntu')

Create a password like so:

```
sudo passwd ubuntu
```

and set it to something you'll remember.

5. Install R and the gdebi tool

Update the software catalog and install a few things:

```
sudo apt-get update && sudo apt-get -y install gdebi-core r-base
```

This will take a few minutes.

6. Download & install RStudio Server

```
wget https://download2.rstudio.org/rstudio-server-0.99.891-amd64.deb
sudo gdebi -n rstudio-server-0.99.891-amd64.deb
```

Upon success, you should see:

```
Mar 07 15:20:18 ip-172-31-6-68 systemd[1]: Starting RStudio Server...
Mar 07 15:20:18 ip-172-31-6-68 systemd[1]: Started RStudio Server.
```

7. Open your RStudio Server instance

Finally, go to 'http://' + your hostname + ':8787' in a browser, eg.

```
http://ec2-XX-YY-33-165.us-west-1.compute.amazonaws.com:8787/
```

and log into RStudio with username 'ubuntu' and the password you set it to above.

Voila!

You can now just go ahead and use this, or you can "stop" it, or you can freeze into an AMI for later use.

Note that on reboot, RStudio Server will start up again and all your files will be there.

Go back to the index: Starting up an Amazon Web Services machine.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

CHAPTER 4

Short read quality and trimming

Start up an instance with ami-05384865 and 500 GB of local storage (Start an Amazon Web Services computer:). You should also configure your firewall (Configure your instance firewall) to pass through TCP ports 8000-8888.

Then, Log into your computer.

—

You should now be logged into your Amazon computer! You should see something like this:

```
ubuntu@ip-172-30-1-252:~$
```

this is the command prompt.

4.1 Prepping the computer

Before we do anything else, we need to set up a place to work and install a few things.

First, let's set up a place to work. Here, we'll make /mnt writeable:

```
sudo chmod a+rwxt /mnt
```

Note: /mnt is the location we're going to use on Amazon computers, but if you're working on a local cluster, it will have a different location. Talk to your local sysadmin and ask them where they recommend putting lots of short-term working files, i.e. the "scratch" space.

4.2 Installing some software

Run:

```
sudo apt-get -y update && \
sudo apt-get -y install trimmomatic fastqc python-pip \
   samtools zlib1g-dev ncurses-dev python-dev
```

Install anaconda:

```
curl -O https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh
bash Anaconda3-4.2.0-Linux-x86_64.sh
```

Then update your environment and install khmer:

```
source ~/.bashrc

cd
git clone https://github.com/dib-lab/khmer.git
cd khmer
sudo python2 setup.py install
```

4.3 Running Jupyter Notebook

Let's also run a Jupyter Notebook in /mnt. First, configure it a teensy bit more securely, and also have it run in the background:

```
jupyter notebook --generate-config

cat >>/home/ubuntu/.jupyter/jupyter_notebook_config.py <<EOF
c = get_config()
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.password = u'sha1:5d813e5d59a7:b4e430cf6dbd1aad04838c6e9cf684f4d76e245c'
c.NotebookApp.port = 8000

EOF
```

Now, run!

```
cd /mnt
jupyter notebook &
```

You should be able to visit port 8000 on your AWS computer and see the Jupyter console. (The password is 'davis'.)

4.4 Data source

We're going to be using a subset of data from Hu et al., 2016. This paper from the Banfield lab samples some relatively low diversity environments and finds a bunch of nearly complete genomes.

(See DATA.md for a list of the data sets we're using in this tutorial.)

4.5 1. Copying in some data to work with.

We've loaded subsets of the data onto an Amazon location for you, to make everything faster for today's work. We're going to put the files on your computer locally under the directory /mnt/data:

```
mkdir /mnt/data
```

Next, let's grab part of the data set:

```
cd /mnt/data
curl -O -L https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-1
curl -O -L https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-1
```

Now if you type:

```
ls -l
```

you should see something like:

```
total 346936
-rw-rw-r-- 1 ubuntu ubuntu 169620631 Oct 11 23:37 SRR1976948_1.fastq.gz
-rw-rw-r-- 1 ubuntu ubuntu 185636992 Oct 11 23:38 SRR1976948_2.fastq.gz
```

These are 1m read subsets of the original data, taken from the beginning of the file.

One problem with these files is that they are writeable - by default, UNIX makes things writeable by the file owner. Let's fix that before we go on any further:

```
chmod u-w *
```

We'll talk about what these files are below.

4.6 1. Copying data into a working location

First, make a working directory; this will be a place where you can futz around with a copy of the data without messing up your primary data:

```
mkdir /mnt/work
cd /mnt/work
```

Now, make a "virtual copy" of the data in your working directory by linking it in –

```
ln -fs /mnt/data/* .
```

These are FASTQ files – let's take a look at them:

```
less SRR1976948_1.fastq.gz
```

(use the spacebar to scroll down, and type 'q' to exit 'less')

Question:

- where does the filename come from?
- why are there 1 and 2 in the file names?

Links:

- FASTQ Format

4.7 2. FastQC

We're going to use FastQC to summarize the data. We already installed 'fastqc' on our computer for you.

Now, run FastQC on two files:

```
fastqc SRR1976948_1.fastq.gz
fastqc SRR1976948_2.fastq.gz
```

Now type 'ls':

```
ls -d *fastqc*
```

to list the files, and you should see:

```
SRR1976948_1_fastqc.html
SRR1976948_1_fastqc.zip
SRR1976948_2_fastqc.html
SRR1976948_2_fastqc.zip
```

You can download these files using your Jupyter Notebook console, if you like; or you can look at these copies of them:

- SRR1976948_1_fastqc/fastqc_report.html
- SRR1976948_2_fastqc/fastqc_report.html

Questions:

- What should you pay attention to in the FastQC report?
- Which is "better", file 1 or file 2? And why?

Links:

- FastQC
- FastQC tutorial video

4.8 3. Trimmomatic

Now we're going to do some trimming! We'll be using Trimmomatic, which (as with fastqc) we've already installed via apt-get.

The first thing we'll need are the adapters to trim off:

```
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-semi-2015-03-04/TruSeq2-PE.fa
```

Now, to run Trimmomatic:

```
TrimmomaticPE SRR1976948_1.fastq.gz \
              SRR1976948_2.fastq.gz \
     SRR1976948_1.qc.fq.gz s1_se \
     SRR1976948_2.qc.fq.gz s2_se \
     ILLUMINACLIP:TruSeq2-PE.fa:2:40:15 \
     LEADING:2 TRAILING:2 \
     SLIDINGWINDOW:4:2 \
     MINLEN:25
```

You should see output that looks like this:

```
...
Input Read Pairs: 1000000 Both Surviving: 885734 (88.57%) Forward Only Surviving: 114262 (11.43%) Rev
TrimmomaticPE: Completed successfully
```

Questions:

- How do you figure out what the parameters mean?
- How do you figure out what parameters to use?
- What adapters do you use?
- What version of Trimmomatic are we using here? (And FastQC?)
- Do you think parameters are different for RNAseq and genomic data sets?

- What's with these annoyingly long and complicated filenames?
- why are we running R1 and R2 together?

For a discussion of optimal trimming strategies, see MacManes, 2014 – it's about RNAseq but similar arguments should apply to metagenome assembly.

Links:

- Trimmomatic

4.9 4. FastQC again

Run FastQC again on the trimmed files:

```
fastqc SRR1976948_1.qc.fq.gz
fastqc SRR1976948_2.qc.fq.gz
```

And now view my copies of these files:

- SRR1976948_1.qc_fastqc/fastqc_report.html
- SRR1976948_2.qc_fastqc/fastqc_report.html

Let's take a look at the output files:

```
less SRR1976948_1.qc.fq.gz
```

(again, use spacebar to scroll, 'q' to exit less).

Questions:

- is the quality trimmed data "better" than before?
- Does it matter that you still have adapters!?

Optional: K-mer Spectral Error Trimming

Next: Run the MEGAHIT assembler

CHAPTER 5

K-mer Spectral Error Trimming

(Optional)

khmer documentation: http://khmer.readthedocs.io/en/latest

If you plot a k-mer abundance histogram of the samples, you'll notice something: there's an awful lot of unique (abundance=1) k-mers. These are erroneous k-mers caused by sequencing errors.

In a new Python3 Jupyter Notebook, run:

```
cd /mnt/work
```

and then

```
!abundance-dist-single.py -M 1e9 -k 21 SRR1976948_1.fastq.gz SRR1976948_1.fastq.gz.dist
```

and in another cell:

```
%matplotlib inline
import numpy
from pylab import *
dist1 = numpy.loadtxt('SRR1976948_1.fastq.gz.dist', skiprows=1, delimiter=',')
plot(dist1[:,0], dist1[:,1])
axis(xmax=50)
```

Many of these errors remain even after you do the Trimmomatic run; you can see this with:

```
!abundance-dist-single.py -M 1e9 -k 21 SRR1976948_1.qc.fq.gz SRR1976948_1.qc.fq.gz.dist
```

and then plot:

```
dist2 = numpy.loadtxt('SRR1976948_1.qc.fq.gz.dist', skiprows=1, delimiter=',')
plot(dist1[:,0], dist1[:,1], label='untrimmed')
plot(dist2[:,0], dist2[:,1], label='trimmed')
legend(loc='upper right')
axis(xmax=50)
```

This is for two reasons:

First, Trimmomatic trims based solely on the quality score, which is a statistical statement about the correctness of a base - a Q score of 30 means that, of 1000 bases with that Q score, 1 of those bases will be wrong. So, a base can have a high Q score and still be wrong! (and **many** bases will have a low Q score and still be correct)

Second, we trimmed **very** lightly - only bases that had a very low quality were removed. This was intentional because with assembly, you want to retain as much coverage as possible, and the assembler will generally figure out what the "correct" base is from the coverage.

An alternative to trimming based on the quality scores is to trim based on k-mer abundance - this is known as k-mer spectral error trimming. K-mer spectral error trimming *always* beats quality score trimming in terms of eliminating errors; e.g. look at this table from Zhang et al., 2014:

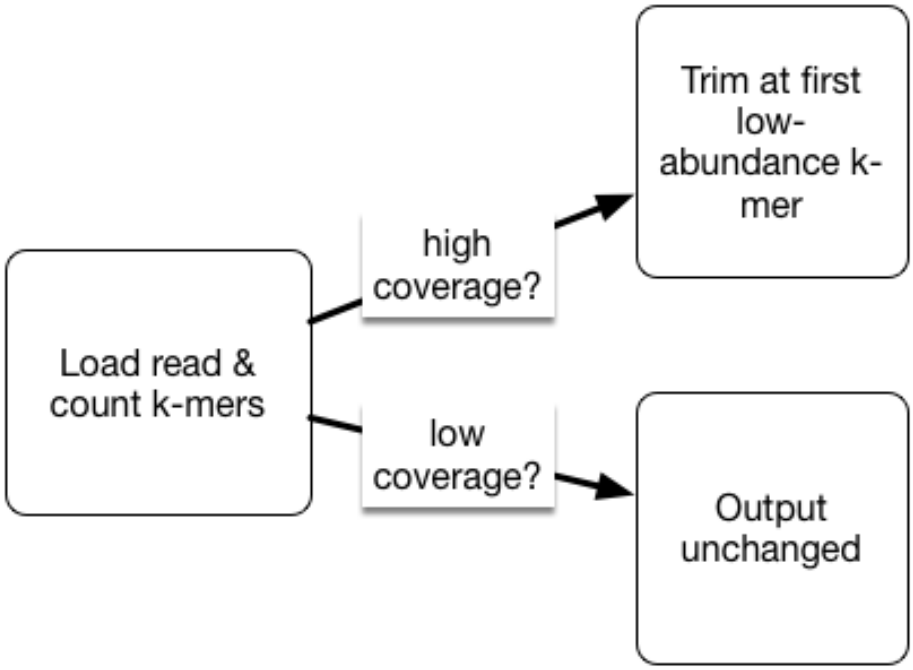
	FP rate	bases trimmed	distinct k-mers	unique k-mers
untrimmed	-	-	41.6 m	34.1 m
khmer iteration 1	80.0%	13.5%	13.3 m	6.5 m
khmer iteration 2	40.2%	1.7%	7.6 m	909.9k
khmer iteration 3	25.4%	0.3%	6.8 m	168.1k
khmer iteration 4	23.2%	0.1%	6.7 m	35.8k
khmer iteration 5	22.8%	0.0%	6.6 m	7.9k
khmer iteration 6	22.7%	0.0%	6.6 m	1.9k
filter by FASTX	-	9.1%	26.6 m	20.3 m
filter by seqtk(default)	-	8.9%	17.7 m	12.1 m
filter by seqtk(-q 0.01)	-	15.4%	9.9 m	5.1 m
filter by seqtk(-b 3 -e 5)	-	8.0%	34.5 m	27.7 m

The results of trimming reads at unique (erroneous) k-mers from a 5 m read *E. coli* data set (1.4 GB) in under 30

The basic logic is this: if you see low abundance k-mers in a high coverage data set, those k-mers are almost certainly the result of errors. (Caveat: strain variation could also create them.)

In metagenomic data sets we do have the problem that we may have very low and very high coverage data. So we don't necessarily want to get rid of all low-abundance k-mers, because they may represent truly low abundance (but useful) data.

As part of the khmer project in my lab, we have developed an approach that sorts reads into high abundance and low abundance reads, and only error trims the high abundance reads.



This does mean that many errors may get left in the data set, because we have no way of figuring out if they are errors or simply low coverage, but that's OK (and you can always trim them off if you really care).

To run such error trimming, use the command `trim-low-abund.py` (at the command line, or prefix with a '!' in the notebook):

```
interleave-reads.py SRR1976948_1.qc.fq.gz SRR1976948_2.qc.fq.gz |
   trim-low-abund.py -V -M 8e9 -C 3 -Z 10 - -o SRR1976948.trim.fq
```

5.1 Why (or why not) do k-mer trimming?

If you can assemble your data set without k-mer trimming, there's no reason to do it. The reason we're error trimming here is to speed up the assembler (by removing data) and to decrease the memory requirements of the assembler (by removing a number of k-mers).

To see how many k-mers we removed, you can examine the distribution as above, or use the `unique-kmers.py` script:

```
unique-kmers.py SRR1976948_1.qc.fq.gz SRR1976948_2.qc.fq.gz
unique-kmers.py SRR1976948.trim.fq
```

Next: Run the MEGAHIT assembler

CHAPTER 6

Run the MEGAHIT assembler

MEGAHIT is a very fast, quite good assembler designed for metagenomes.

First, install it:

```
cd
git clone https://github.com/voutcn/megahit.git
cd megahit
make
```

Now, download some data:

```
cd /mnt/data
curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/S
curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/S
```

These are data that have been run through k-mer abundance trimming (see K-mer Spectral Error Trimming) and subsampled so that we can run an assembly in a fairly short time period :).

Now, finally, run the assembler!

```
mkdir /mnt/assembly
cd /mnt/assembly
ln -fs ../data/*.subset.pe.fq.gz .

~/megahit/megahit --12 SRR1976948.abundtrim.subset.pe.fq.gz,SRR1977249.abundtrim.subset.pe.fq.gz \
    -o combined
```

This will take about 25 minutes; at the end you should see output like this:

```
... 12787984 bp, min 200 bp, max 61353 bp, avg 1377 bp, N50 3367 bp
... ALL DONE. Time elapsed: 1592.503825 seconds
```

The output assembly will be in `combined/final.contigs.fa`.

6.1 While the assembly runs...

How assembly works - whiteboarding the De Bruijn graph approach.

Interpreting the MEGAHIT working output :)

What does, and doesn't, assemble?

How good is assembly anyway?

Discussion:

Why would we assemble, vs looking at raw reads? What are the advantages and disadvantages?

What are the technology tradeoffs between Illumina HiSeq, Illumina MiSeq, and PacBio? (Also see this paper.)

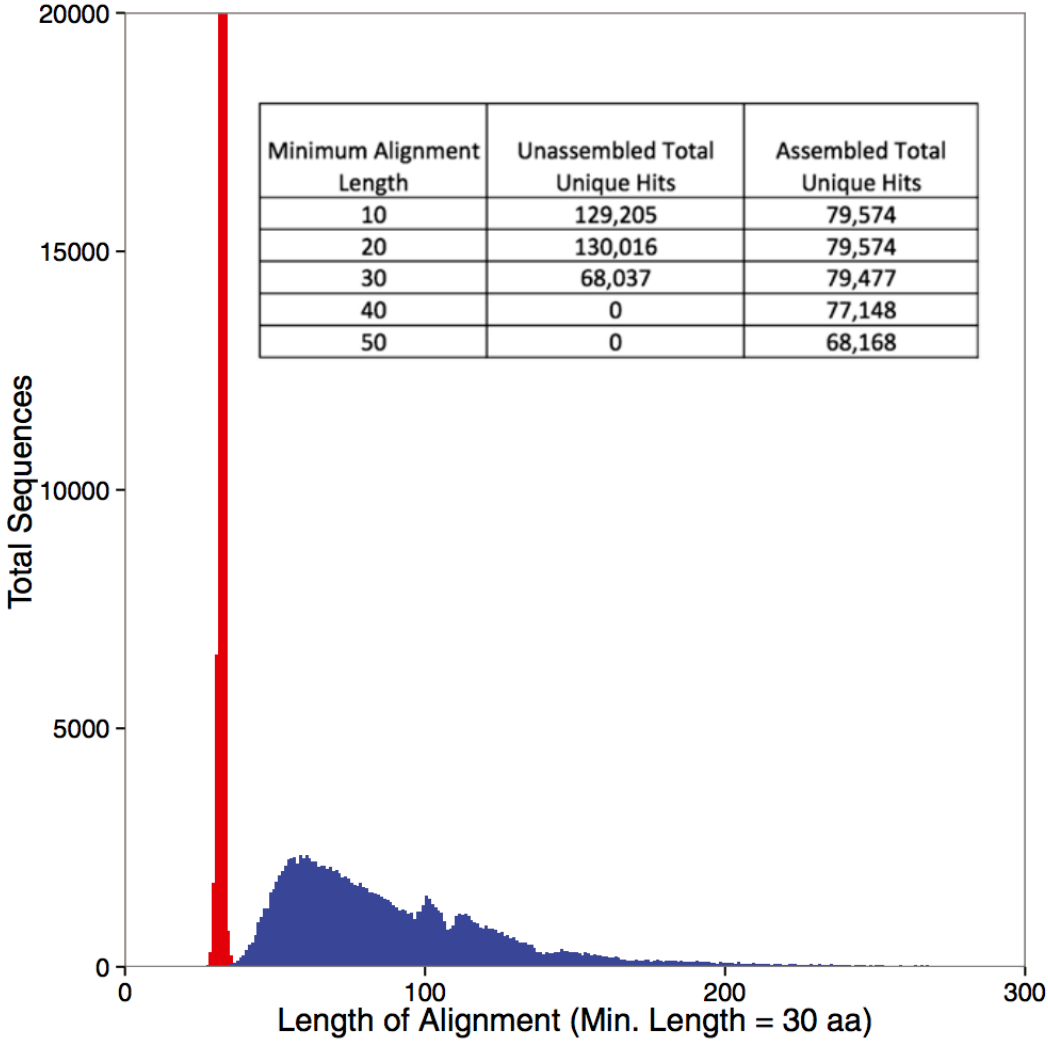
What kind of experimental design considerations should you have if you plan to assemble?

Some figures: the first two come from work by Dr. Sherine Awad on analyzing the data from Shakya et al (2014). The third comes from an analysis of read search vs contig search of a protein database.

Table 1. Running Time and Memory Utilization

(1) IDBA-UD	
Running Time	33h 54m
Memory Utilization (GB)	123.84
(2) SPAdes	
Running Time	67h 02m
Memory Utilization (GB)	381.79
(3) MEGAHIT	
Running Time	1h 53m
Memory Utilization (GB)	33.41

(2) Ambiguous Approach		
(1) IDBA-UD		
99.0	Genome Coverage Duplication Ratio	88.84% 0.92%
95.0	Genome Coverage Duplication Ratio	94.98% 1.84%
(2) SPAdes		
99.0	Genome Coverage Duplication Ratio	88.34% 0.83%
95.0	Genome Coverage Duplication Ratio	94.96% 1.11%
(3) MEGAHIT		
99.0	Genome Coverage Duplication Ratio	88.70% 0.10%
95.0	Genome Coverage Duplication Ratio	94.47% 1.73%



Minimum Alignment Length	Unassembled Total Unique Hits	Assembled Total Unique Hits
10	129,205	79,574
20	130,016	79,574
30	68,037	79,477
40	0	77,148
50	0	68,168

6.2 After the assembly is finished

At this point we can do a bunch of things:

- annotate the assembly (Annotation with Prokka);
- evaluate the assembly's inclusion of k-mers and reads;
- set up a BLAST database so that we can search it for genes of interest;
- quantify the abundance of the contigs or genes in the assembly, using the original read data set (Gene Abundance Estimation with Salmon);
- bin the contigs in the assembly into species bins;

CHAPTER 7

Annotation with Prokka

Prokka is a tool that facilitates the fast annotation of prokaryotic genomes.

The goals of this tutorial are to:

- Install Prokka
- Use Prokka to annotate our genomes

7.1 Installing Prokka

Download and extract the latest version of prokka:

```
cd ~/
wget http://www.vicbioinformatics.com/prokka-1.11.tar.gz
tar -xvzf prokka-1.11.tar.gz
```

We also will need some dependencies such as bioperl:

```
sudo apt-get install bioperl libdatetime-perl libxml-simple-perl libdigest-md5-perl
sudo perl -MCPAN -e shell
sudo perl -MCPAN -e 'install "XML::Simple"'
```

Now, you should be able to add Prokka to your `$PATH` and set up the index for the sequence database:

```
export PATH=$PATH:$HOME/prokka-1.11/bin
prokka --setupdb
```

Prokka should be good to go now– you can check to make sure that all is well by typing `prokka`. This should print the help screen with all available options.

7.2 Running Prokka

Make a new directory for the annotation:

```
cd /mnt
mkdir annotation
cd annotation
```

Link the metagenome assembly file into this directory:

```
ln -fs /mnt/assembly/combined/final.contigs.fa
```

Now it is time to run Prokka! There are tons of different ways to specialize the running of Prokka. We are going to keep it simple for now, though. It will take a little bit to run.

```
prokka subset_assembly.fa --outdir prokka_annotation --prefix metagG
```

This will generate a new folder called `prokka_annotation` in which will be a series of files, which are detailed here.

In particular, we will be using the `*.ffn` file to assess the relative read coverage within our metagenomes across the predicted genomic regions.

7.3 References

- http://www.vicbioinformatics.com/software.prokka.shtml
- https://www.ncbi.nlm.nih.gov/pubmed/24642063
- https://github.com/tseemann/prokka/blob/master/README.md

CHAPTER 8

Day 2 - installation instructions

(Instructions mostly copied from Short read quality and trimming!)

Use ami-05384865, with a 500 GB local disk (see: Start an Amazon Web Services computer:)

Make `/mnt/` read/write:

```
sudo chmod a+rwxt /mnt
```

Run:

```
sudo apt-get -y update && \
sudo apt-get -y install trimmomatic fastqc python-pip \
   samtools zlib1g-dev ncurses-dev python-dev
```

Install anaconda:

```
curl -O https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86_64.sh
bash Anaconda3-4.2.0-Linux-x86_64.sh
```

Then update your environment and install khmer:

```
source ~/.bashrc

cd
git clone https://github.com/dib-lab/khmer.git
cd khmer
sudo python2 setup.py install
```

8.1 Running Jupyter Notebook

Let's also run a Jupyter Notebook in /mnt. First, configure it a teensy bit more securely, and also have it run in the background.

Generate a config:

```
jupyter notebook --generate-config
```

Add a password, have it not run a browser, and put it on port 8000 by default:

```
cat >>/home/ubuntu/.jupyter/jupyter_notebook_config.py <<EOF
c = get_config()
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
```

```
c.NotebookApp.password = u'sha1:5d813e5d59a7:b4e430cf6dbd1aad04838c6e9cf684f4d76e245c'
c.NotebookApp.port = 8000

EOF
```

Now, run!

```
cd /mnt
jupyter notebook &
```

You should be able to visit port 8000 on your AWS computer and see the Jupyter console.

CHAPTER 9

Gene Abundance Estimation with Salmon

Salmon is one of a breed of new, very fast RNAseq counting packages. Like Kallisto and Sailfish, Salmon counts fragments without doing up-front read mapping. Salmon can be used with edgeR and others to do differential expression analysis (if you are quantifying RNAseq data).

Today we will use it to get a handle on the relative distribution of genomic reads across the predicted protein regions.

The goals of this tutorial are to:

- Install salmon
- Use salmon to estimate gene coverage in our metagenome dataset

Extra resources:

- see the finished plotting notebook.
- see the extract-sequences.py script.

9.1 Installing Salmon

Download and extract the latest version of Salmon and add it to your PATH:

```
cd
wget https://github.com/COMBINE-lab/salmon/releases/download/v0.7.2/Salmon-0.7.2_linux_x86_64.tar.gz
tar -xvzf Salmon-0.7.2_linux_x86_64.tar.gz
cd Salmon-0.7.2_linux_x86_64
export PATH=$PATH:$HOME/Salmon-0.7.2_linux_x86_64/bin
```

9.2 Running Salmon

Go to the data directory and download the prokka annotated sequences, assembled metagenome, and fastq files:

```
mkdir -p /mnt/data
cd /mnt/data
curl -L -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-1
curl -L -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-1
curl -L -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-1
tar -xvzf prokka_annotation_assembly.tar.gz
```

Make a new directory for the quantification of data with Salmon:

```
mkdir /mnt/quant
cd /mnt/quant
```

Grab the nucleotide (`*ffn`) predicted protein regions from Prokka and link them here. Also grab the trimmed sequence data (`*fq`)

```
ln -fs /mnt/data/prokka_annotation/*ffn .
ln -fs /mnt/data/*.abundtrim.subset.pe.fq.gz .
```

Create the salmon index:

```
salmon index -t metagG.ffn -i transcript_index --type quasi -k 31
```

Salmon requires that paired reads be separated into two files. We can split the reads using the `split-paired-reads.py` from the khmer package:

```
for file in *.abundtrim.subset.pe.fq.gz
do
  tail=.fq.gz
  BASE=${file/$tail/}
  split-paired-reads.py $BASE$tail -1 ${file/$tail/}.1.fq -2 ${file/$tail/}.2.fq
done
```

Now, we can quantify our reads against this reference:

```
for file in *.pe.1.fq
do
tail1=.abundtrim.subset.pe.1.fq
tail2=.abundtrim.subset.pe.2.fq
BASE=${file/$tail1/}
salmon quant -i transcript_index --libType IU \
      -1 $BASE$tail1 -2 $BASE$tail2 -o $BASE.quant;
 done
```

(Note that –libType must come before the read files!)

This will create a bunch of directories named after the fastq files that we just pushed through. Take a look at what files there are within one of these directories:

```
find SRR1976948.quant -type f
```

9.3 Working with count data

Now, the `quant.sf` files actually contain the relevant information about expression – take a look:

```
head -10 SRR1976948.quant/quant.sf
```

The first column contains the transcript names, and the fourth column is what we will want down the road - the normalized counts (TPM). However, they're not in a convenient location / format for use; let's fix that.

Download the gather-counts.py script:

```
curl -L -O https://raw.githubusercontent.com/ngs-docs/2016-metagenomics-sio/master/gather-counts.py
```

and run it:

```
python2 ./gather-counts.py
```

This will give you a bunch of .counts files, which are processed from the quant.sf files and named for the directory from which they emanate.

9.4 Plotting the results

In Jupyter Notebook, open a new Python3 notebook and enter:

```
%matplotlib inline
import numpy
from pylab import *
```

In another cell:

```
cd /mnt/quant
```

In another cell:

```
counts1 = [ x.split()[1] for x in open('SRR1976948.quant.counts')]
counts1 = [ float(x) for x in counts1[1:] ]
counts1 = numpy.array(counts1)

counts2 = [ x.split()[1] for x in open('SRR1977249.quant.counts')]
counts2 = [ float(x) for x in counts2[1:] ]
counts2 = numpy.array(counts2)

plot(counts1, counts2, '*')
```

9.5 References

- http://salmon.readthedocs.io/en/latest/salmon.html
- http://biorxiv.org/content/early/2016/08/30/021592

CHAPTER 10

Mapping

Download bwa:

```
cd
curl -L https://sourceforge.net/projects/bio-bwa/files/bwa-0.7.15.tar.bz2/download > bwa-0.7.15.tar.b
```

Unpack and build it:

```
tar xjvf bwa-0.7.15.tar.bz2
cd bwa-0.7.15
make
```

Install it:

```
sudo cp bwa /usr/local/bin
```

10.1 Downloading data

Now, go to a new directory and grab the data:

```
mkdir /mnt/mapping
cd /mnt/mapping

curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/S
curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/S
```

We will also need the assembly; rather than rebuilding it, you can download a copy that we saved for you:

```
curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/s
gunzip subset_assembly.fa
```

Next, you'll need to index the assembly:

```
bwa index subset_assembly.fa
```

10.2 Splitting the reads

The reads are in paired-end/interleaved format, so you'll need to split them -:

```
for i in *.pe.fq.gz
do
   gunzip -c $i | head -800000 | split-paired-reads.py -1 $i.1 -2 $i.2 -
done
```

This will take the interleaved reads and produce .1 and .2 files from them.

10.3 Mapping the reads

Map the left reads:

```
for i in *.1
do
   bwa aln subset_assembly.fa $i > $(echo $i | cut -d. -f1)_1.sai
done
```

Map the right reads:

```
for i in *.2
do
   bwa aln subset_assembly.fa $i > $(echo $i | cut -d. -f1)_2.sai
done
```

Combine the paired ends with bwa sampe:

```
bwa sampe subset_assembly.fa SRR1976948_1.sai SRR1976948_2.sai SRR1976948.*.1 SRR1976948.*.2 > SRR197
bwa sampe subset_assembly.fa SRR1977249_1.sai SRR1977249_2.sai SRR1977249.*.1 SRR1977249.*.2 > SRR197
```

10.4 Converting to BAM to visualize

First, index the assembly for samtools:

```
samtools faidx subset_assembly.fa
```

Then, convert both SAM files to BAM files:

```
for i in *.sam
do
   samtools import subset_assembly.fa $i $i.bam
   samtools sort $i.bam $i.bam.sorted
   samtools index $i.bam.sorted.bam
done
```

10.5 Visualizing the read mapping

Find a contig name to visualize:

```
grep -v ^@ SRR1976948.sam | \
    cut -f 3 | sort | uniq -c | sort -n
```

Pick one e.g. k99_13588.

Now execute:

```
samtools tview SRR1976948.sam.bam.sorted.bam subset_assembly.fa -p k99_13588:400
```

(use arrow keys to scroll, 'q' to quit)

Look at it in both mappings:

```
samtools tview SRR1977249.sam.bam.sorted.bam subset_assembly.fa -p k99_13588:400
```

Why is the mapping so good??

Note: no strain variation :).

Grab some untrimmed data:

```
curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/S
curl -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-12/S
```

Now align this untrimmed data:

```
gunzip -c SRR1976948_1.fastq.gz | head -800000 > SRR1976948.1
gunzip -c SRR1976948_2.fastq.gz | head -800000 > SRR1976948.2

bwa aln subset_assembly.fa SRR1976948.1 > SRR1976948_1.untrimmed.sai
bwa aln subset_assembly.fa SRR1976948.2 > SRR1976948_2.untrimmed.sai

bwa sampe subset_assembly.fa SRR1976948_1.untrimmed.sai SRR1976948_2.untrimmed.sai SRR1976948.1 SRR19

i=SRR1976948.untrimmed.sam
samtools import subset_assembly.fa $i $i.bam
samtools sort $i.bam $i.bam.sorted
samtools index $i.bam.sorted.bam
```

And now look:

```
samtools tview SRR1976948.untrimmed.sam.bam.sorted.bam subset_assembly.fa -p k99_13588:500
```

You can also use 'Tablet' to view the downloaded BAM file - see the Tablet paper.

CHAPTER 11

Slicing and dicing with k-mers

(Note, this won't work with amplified data.)

Extra resources:

- plotting notebook

—

At the command line, create a new directory and extract some data:

```
cd /mnt
mkdir slice
cd slice
```

We're going to work with half the read data set for speed reasons –

```
gunzip -c ../mapping/SRR1976948.abundtrim.subset.pe.fq.gz | \
   head -6000000 > SRR1976948.half.fq
```

In a Jupyter Notebook (go to 'http://' + machine name + ':8000'), password 'davis', create new Python notebook "conda root", run:

```
cd /mnt/slice
```

and then in another cell:

```
!load-into-counting.py -M 4e9 -k 31 SRR1976948.kh SRR1976948.half.fq
```

and in another cell:

```
!abundance-dist.py SRR1976948.kh SRR1976948.half.fq SRR1976948.dist
```

and in yet another cell:

```
%matplotlib inline
import numpy
from pylab import *
dist1 = numpy.loadtxt('SRR1976948.dist', skiprows=1, delimiter=',')
plot(dist1[:,0], dist1[:,1])
axis(ymax=10000, xmax=1000)
```

Then:

```
python2 ~/khmer/sandbox/calc-median-distribution.py SRR1976948.kh \
   SRR1976948.half.fq SRR1976948.readdist
```

And:

```
python2 ~/khmer/sandbox/slice-reads-by-coverage.py SRR1976948.kh SRR1976948.half.fq slice.fq -m 0 -M
```

11.1 Assemble the slice

(Re)install megahit:

```
cd
git clone https://github.com/voutcn/megahit.git
cd megahit
make
```

Go back to the slice directory and extract paired ends:

```
cd /mnt/slice
extract-paired-ends.py slice.fq
```

Assemble!

```
~/megahit/megahit --12 slice.fq.pe -o slice
```

The contigs will be in `slice/final.contigs.fa`.

CHAPTER 12

Using and Installing Circos

Circos is a powerful visualization tool that allows for the creation of circular graphics to display complex genomic data (e.g. genome comparisons). On top of the circular ideogram generated can be layered any number of graphical information (heatmaps, scatter plots, etc.).

The goals of this tutorial are to:

- Install circos on your Ubuntu AWS system
- Use Circos to visualize our metagenomic data

Note: Beyond this brief crash course , circos is very well-documented and has a great series of tutorials and course materials that are useful.

12.1 Installing Circos

You'll need to install one additional ubuntu package, libgd:

```
sudo apt-get -y install libgd-perl
```

Within your Amazon Instance make a directory called circos and navigate into it. There, we will download and extract the latest version of circos:

```
cd /mnt
mkdir circos
cd circos
curl -O http://dib-training.ucdavis.edu.s3.amazonaws.com/metagenomics-scripps-2016-10-12/circos-0.69-
tar -xvzf circos-0.69-3.tar.gz
```

Circos runs within Perl and as such does not need to be compiled to run. So, we can just add the location of circos to our path variable. (Alternatively, you can append this statement to the end of your `.bashrc` file.)

```
export PATH=/mnt/circos/circos-0.69-3/bin:$PATH
```

Circos does, however, require quite a few additional perl modules to operate correctly. To see what modules are missing and need to be downloaded type the following:

```
circos -modules > modules
```

Now, to download all of these we will be using CPAN, a package manager for perl. We are going to pick out all the missing modules and then loop over those modules and download them using cpan.

```
grep missing modules |cut -f13 -d " " > missing_modules
for mod in $(cat missing_modules);
  do
  sudo cpan install $mod;
  done
```

This will take a while to run. When it is done check that you now have all modules downloaded by typing:

```
circos -modules
```

If you got all 'ok' then you are good to go!

And with that, circos should be up and ready to go. Run the example by navigating to the examples folder within the circos folder.

```
cd /mnt/circos/circos-0.69-3/example
bash run
```

This will take a little bit to run but should generate a file called `circos.png`. Open it and you can get an idea of the huge variety of things that are possible with circos and a lot of patience. We will not be attempting anything that complex today, however.

12.2 Visualizing Gene Coverage and Orientation

First, let's make a directory where we will be doing all of our work for plotting:

```
mkdir /mnt/circos/plotting
cd /mnt/circos/plotting
```

Now, link in the `*gff` file output from prokka (which we will use to define the location of genes in each of our genomes), the genome assembly file `final.contigs.fa`, and the `SRR*counts` files that we generated with salmon:

```
ln -fs /mnt/data/prokka_annotation/*gff .
ln -fs /mnt/data/final.contigs.fa .
ln -fs /mnt/quant/*counts .
```

We also need to grab a set of useful scripts and config files for this plotting exercise:

```
curl -L -O https://github.com/ngs-docs/2016-metagenomics-sio/raw/master/circos-build.tar.gz
tar -xvzf circos-build.tar.gz
curl -L -O https://s3-us-west-1.amazonaws.com/dib-training.ucdavis.edu/metagenomics-scripps-2016-10-1
gunzip subset_assembly.fa.gz
mv subset_assembly.fa final.contigs.fa
```

We are going to limit the data we are trying to visualize and get longest contigs from our assembly. We can do this using a script from the khmer package:

```
extract-long-sequences.py  final.contigs.fa -l 24000 -o final.contigs.long.fa
```

Next, we will run a script that processes the data from the the files that we just moved to create circos-acceptable files. This is really the crux of using circos: figuring out how to get your data into the correct format.

```
python parse_data_for_circos.py
```

If you are interested– take a look at the script and the input files to see how these data were manipulated.

Circos operates off of three main types of files: 1) a config files that dictate the style and inputs to your circos plot, 2) a karyotype file that defines the size and layout of your "chromosomes", and 3) any data files that you call in your config file that detail attributes you want to plot.

The above script generated our karyotype file and four different data files. What are they? How are they oriented?

Now, we all that is left is actually running circos. Navigate into the circos-build directory and type `circos`:

```
cd circos-build
circos
```

This command should generate an `circos.svg` and `circos.png`. Check out the `circos.png`!

Now, let's take a look at the file that controls this crazy figure– `circos.config`.

Try changing a few parameters– colors, radius, size, to see what you can do. Again, if you are into this type of visualization, do check out the extensive tutorial.

12.3 References

- http://genome.cshlp.org/content/early/2009/06/15/gr.092759.109.abstract
- http://circos.ca/documentation/tutorials
- http://circos.ca/documentation/course/

CHAPTER 13

Workflow and repeatability discussion

https://2016-oslo-repeatability.readthedocs.io/en/latest/

CHAPTER 14

Technical information

The github repository for this workshop is public at https://github.com/ngs-docs/2016-metagenomics-sio