labibi Documentation

Release 1.0

C. Titus Brown

November 14, 2015

1 Welcome! 3
1.1 1. Learning goals . . . 3
1.2 2. Safe space and code of conduct . . . 3
1.3 3. Instructor introductions . . . 3
1.4 4. Amazon and cloud computing - why?! . . . 3
1.5 5. Sticky notes and how they work... + Minute Cards . . . 4
1.6 6. Warning: . . . 4

2 Non-model organisms and RNAseq 5
2.1 The overall process . . . 5

3 Getting started with Amazon EC2 7
3.1 More details: . . . 7

4 Short read quality and trimming 17
4.1 Prepping the computer . . . 17
4.2 Data source . . . 17
4.3 1. Copying in some data to work with. . . . 17
4.4 1. Copying data into a working location . . . 18
4.5 2. FastQC . . . 18
4.6 3. Trimmomatic . . . 19
4.7 4. FastQC again . . . 20
4.8 5. Trim the rest of the sequences . . . 20
4.9 6. Interleave the sequences . . . 22

5 Running digital normalization 23

6 Running the actual assembly 25

7 BLASTing your assembled data 27

8 Annotation of denovo transcriptome 29
8.1 Identify the Gene/Transcript relationships: . . . 29
8.2 Generate the longest-ORF peptide candidates from the Trinity Assembly: . . . 29
8.3 Capturing BLAST Homologies . . . 30
8.4 Characterization of functional annotation features . . . 30
8.5 Integration of all annotations into one database . . . 31
8.6 Output an Annotation Report . . . 32

9 Quantification and Differential Expression 33
9.1 Download Express 33
9.2 Align Reads with Bowtie 33
9.3 Quantify Expression using eXpress 34
9.4 Differential Expression 34

10 Remapping your reads to your assembled transcriptome 37

11 Miscellaneous advice 39
11.1 Sequencing depth and number of samples 39
11.2 Downloading your data 39
11.3 Developing your own pipeline 40

12 More resources 41
12.1 Informational resources 41
12.2 Places to share data, scripts, and results files 41

13 Miscellaneous questions 43

14 Tips and Tricks for working with Remote Computers 45
14.1 Use screen to run things that take a long time. 45
14.2 Use CyberDuck to transfer files 45
14.3 Subsetting data 45
14.4 Running full analyses on Amazon Web Services 46

15 Technical information 47

This workshop was actually cancelled :). Now I'm using it as a demo.

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;

For us:

- what are the on campus needs? who are the on-campus people?

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

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

Tamer Mansour - MD/PhD, postdoc with Titus.

Camille Scott - CS graduate student here at Davis.

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;

- they give us $100 gift certificates...

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.**

1.6 6. Warning:

The bit that isn’t going to work is at the beginning. This is unavoidable. To reward you, we have a coffee break at 10:30am...

—

Next: Non-model organisms and RNAseq

CHAPTER 2

Non-model organisms and RNAseq

With non-model systems, where there is neither a good genome nor a lot of mRNAseq data, you have to build your own transcriptome from scratch – so-called “de novo transcriptome assembly”. There are a few programs to do this - most notably Trinity and Oases - and we have found little difference.

The main problem you’ll run into with non-model mRNAseq is that the output is fairly noisy with respect to splice variants. Our experience has been that many of these splice variants are probably “real” – in the sense of actually present – but may be biological “noise”, in the sense that they are not actually functional (See this excellent paper by Pickrell and Pritchard making the case). Regardless, there’s little that you can do about this, although we will talk about it a bit on the second day.

2.1 The overall process

`_static/nonmodel-rnaseq-pipeline.png`

- Copy over your RNAseq data (from two or more samples);
- Trim primers and junk from sequence (Short read quality and trimming)
- Do abundance normalization (Running digital normalization)
- Assemble everything together (Running the actual assembly)

This gives you an assembled transcriptome, consisting of many transcripts and transcript families.

At this point you can do one or more of the following:

- Annotate your transcripts (Annotation of denovo transcriptome)
- Quantify your transcripts and examine differential expression (Quantification and Differential Expression)
- BLAST your transcripts individually (BLASTing your assembled data)

Next: Getting started with Amazon EC2

CHAPTER 3

Getting started with Amazon EC2

Short version:

- go to http://aws.amazon.com/, log in, then “EC2” (upper left);
- select “Launch instance”;
- select “Ubuntu 14.04” from the list;
- select “m3.xlarge” from the list (towards bottom of “General purpose”);
- click “Review and launch”
- select “Launch”;
- if your first time through, create a key pair; otherwise select existing;
- click “launch instance”

3.1 More details:

3.1.1 Start up an EC2 instance

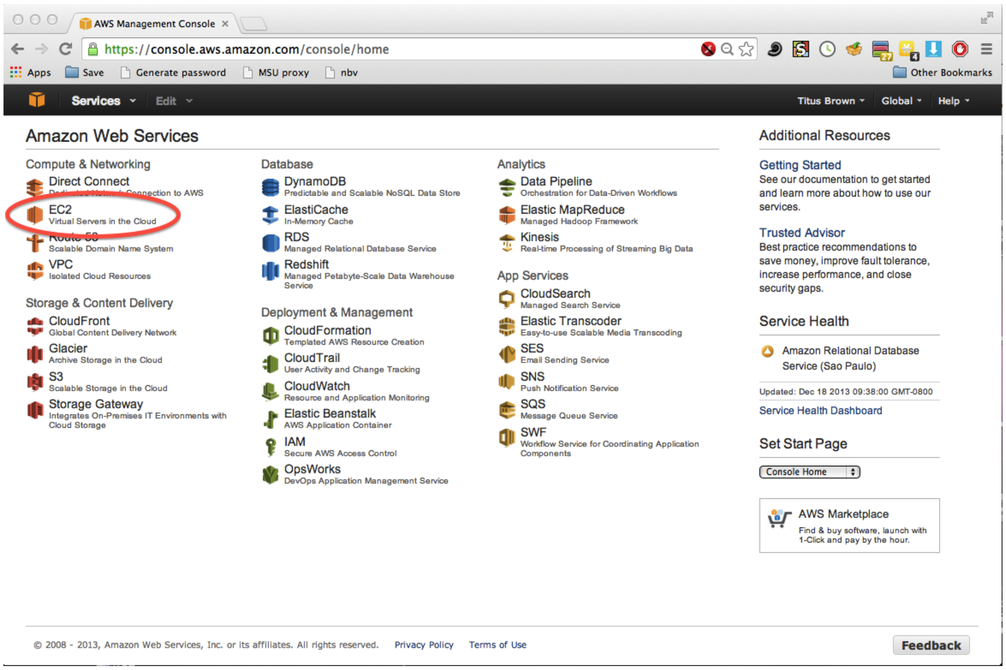
Log in

Go to ‘https://aws.amazon.com‘ in a Web browser.

Select ‘My Account/Console’ menu option ‘AWS Management Console.”

Log in with your username & password.

Click on EC2 (upper left).


AWS Management Console
https://console.aws.amazon.com/console/home
Apps
Save
Generate password
MSU proxy
nbv
Other Bookmarks
Services
Edit
Titus Brown
Global
Help
Amazon Web Services
Compute & Networking
Direct Connect
EC2
Virtual Servers in the Cloud
Route 53
Scalable Domain Name System
VPC
Isolated Cloud Resources
Storage & Content Delivery
CloudFront
Global Content Delivery Network
Glacier
Archive Storage in the Cloud
S3
Scalable Storage in the Cloud
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage
Database
DynamoDB
Predictable and Scalable NoSQL Data Store
ElastiCache
In-Memory Cache
RDS
Managed Relational Database Service
Redshift
Managed Petabyte-Scale Data Warehouse Service
Deployment & Management
CloudFormation
Templated AWS Resource Creation
CloudTrail
User Activity and Change Tracking
CloudWatch
Resource and Application Monitoring
Elastic Beanstalk
AWS Application Container
IAM
Secure AWS Access Control
OpsWorks
DevOps Application Management Service
Analytics
Data Pipeline
Orchestration for Data-Driven Workflows
Elastic MapReduce
Managed Hadoop Framework
Kinesis
Real-time Processing of Streaming Big Data
App Services
CloudSearch
Managed Search Service
Elastic Transcoder
Easy-to-use Scalable Media Transcoding
SES
Email Sending Service
SNS
Push Notification Service
SQS
Message Queue Service
SWF
Workflow Service for Coordinating Application Components
Additional Resources
Getting Started
Trusted Advisor
Service Health
Amazon Relational Database Service (Sao Paulo)
Service Health Dashboard
Set Start Page
Console Home
AWS Marketplace
Privacy Policy
Terms of Use
Feedback

See our documentation to get started and learn more about how to use our services.

Best practice recommendations to save money, improve fault tolerance, increase performance, and close security gaps.

Updated: Dec 18 2013 09:38:00 GMT-0800

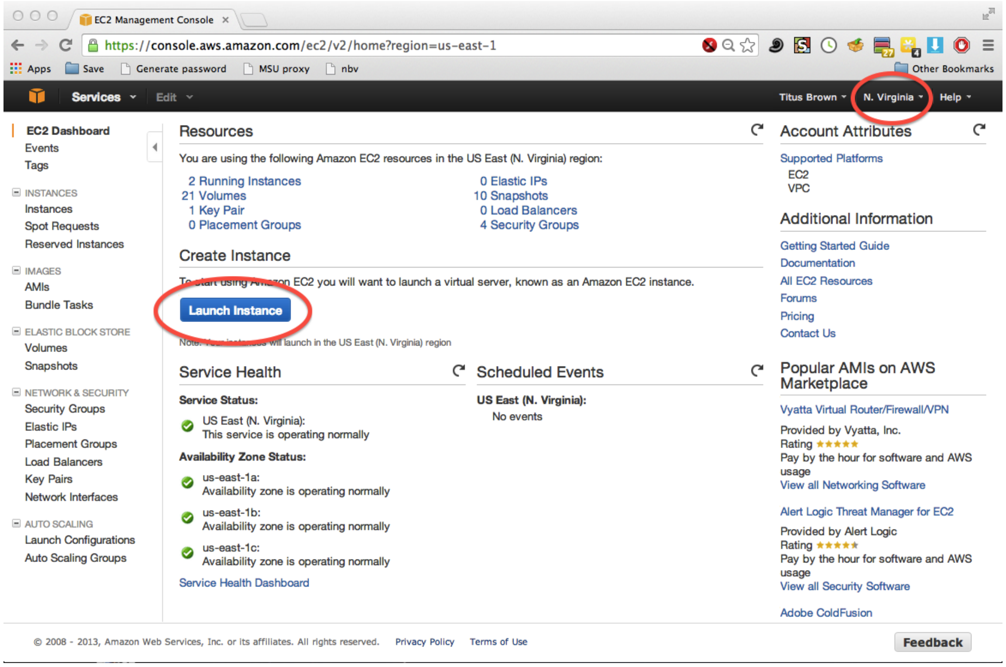
Find & buy software, launch with 1-Click and pay by the hour.

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Select your zone

Many of the resources that we use are hosted by Amazon on the East coast. Make sure that your dashboard has 'N. Virginia' on the upper right.

Then click on Launch Instance.


EC2 Management Console
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1
Apps
Save
Generate password
MSU proxy
nbv
Other Bookmarks
Services
Edit
Titus Brown
N. Virginia
Help
EC2 Dashboard
Events
Tags
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Network Interfaces
AUTO SCALING
Launch Configurations
Auto Scaling Groups
Resources
2 Running Instances
21 Volumes
1 Key Pair
0 Placement Groups
0 Elastic IPs
10 Snapshots
0 Load Balancers
4 Security Groups
Create Instance
Launch Instance
Service Health
Scheduled Events
Account Attributes
Supported Platforms
EC2
VPC
Additional Information
Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us
Popular AMIs on AWS Marketplace
Vyatta Virtual Router/Firewall/VPN
View all Networking Software
Alert Logic Threat Manager for EC2
View all Security Software
Adobe ColdFusion
Service Health Dashboard
Privacy Policy
Terms of Use
Feedback

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (N. Virginia) region

Service Status:

US East (N. Virginia):
This service is operating normally

Availability Zone Status:

us-east-1a:
Availability zone is operating normally

us-east-1b:
Availability zone is operating normally

us-east-1c:
Availability zone is operating normally

US East (N. Virginia):
No events

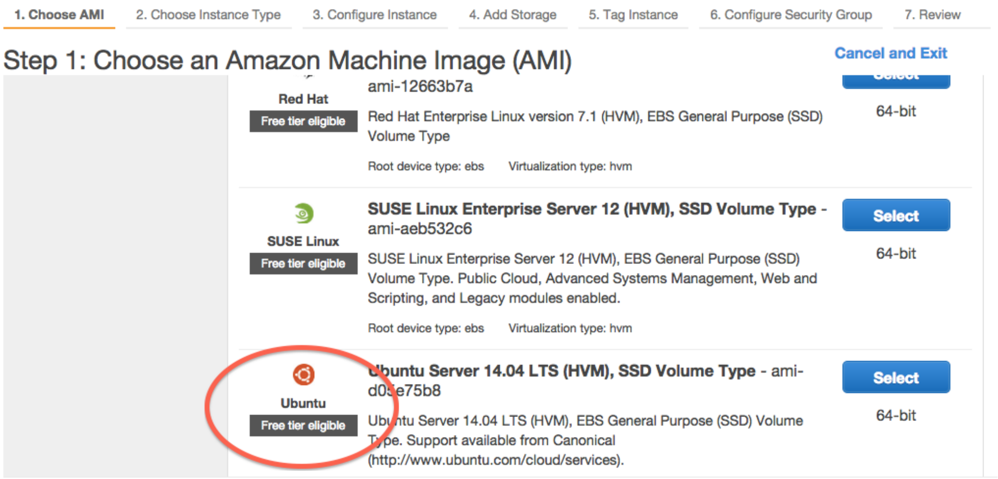
Provided by Vyatta, Inc.
Rating ★★★★★
Pay by the hour for software and AWS usage

Provided by Alert Logic
Rating ★★★★★
Pay by the hour for software and AWS usage

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Select the machine operating system to boot

Find the "Ubuntu Server 14.04" image in the first list to show up.


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Cancel and Exit
Select
Select
Select

Step 1: Choose an Amazon Machine Image (AMI)

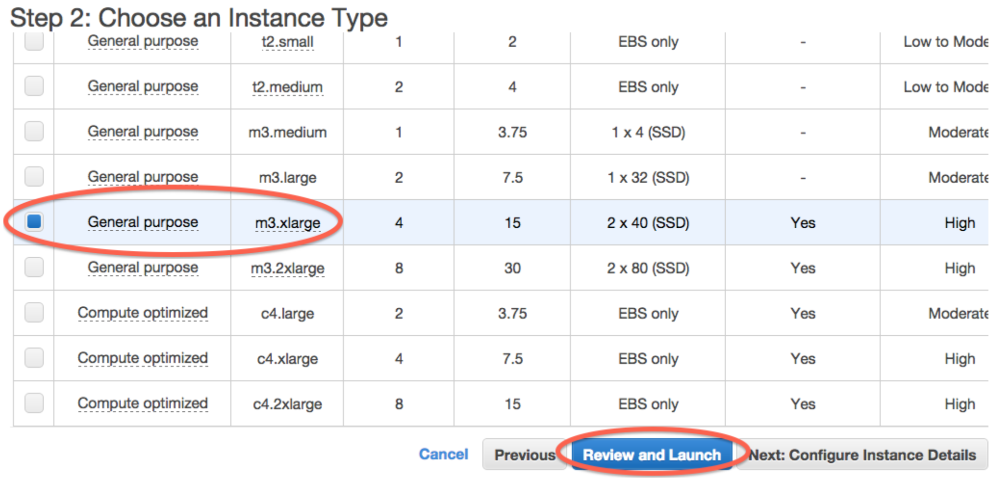
Red Hat Free tier eligible
ami-12663b7a
Red Hat Enterprise Linux version 7.1 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm
64-bit

SUSE Linux Free tier eligible
SUSE Linux Enterprise Server 12 (HVM), SSD Volume Type - ami-aeb532c6
SUSE Linux Enterprise Server 12 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm
64-bit

Ubuntu Free tier eligible
Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8
Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
64-bit

Choose the machine size

Select 'General purpose', 'm3.xlarge', and then 'Review and Launch'.

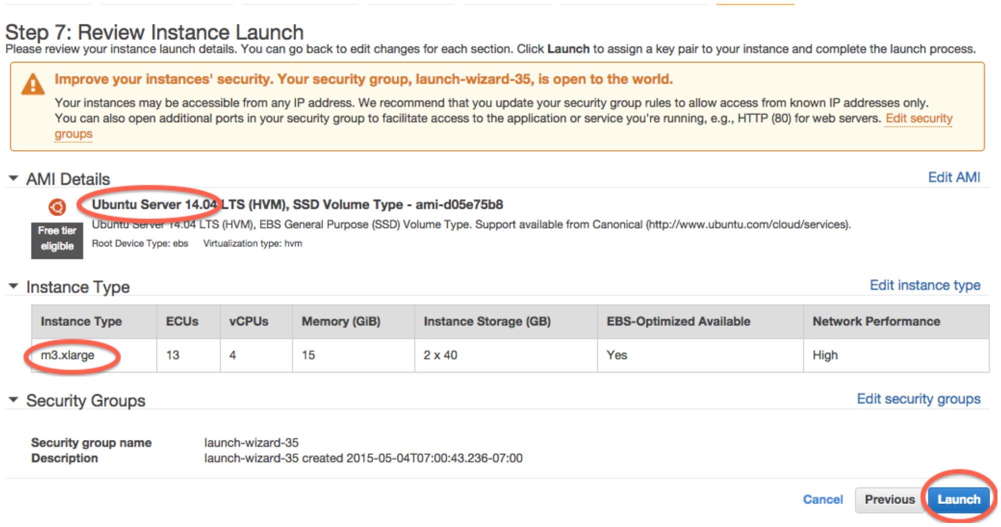

Cancel
Previous
Review and Launch
Next: Configure Instance Details

Step 2: Choose an Instance Type

☐	General purpose	t2.small	1	2	EBS only	-	Low to Mode
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Mode
☐	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
☐	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
☑	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
☐	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High
☐	Compute optimized	c4.large	2	3.75	EBS only	Yes	Moderate
☐	Compute optimized	c4.xlarge	4	7.5	EBS only	Yes	High
☐	Compute optimized	c4.2xlarge	8	15	EBS only	Yes	High

Confirm and launch

Review the details (ignore the warnings!) and click on Launch.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-35, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMI Details

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Root Device Type: ebs Virtualization type: hvm

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
m3.xlarge	13	4	15	2 x 40	Yes	High

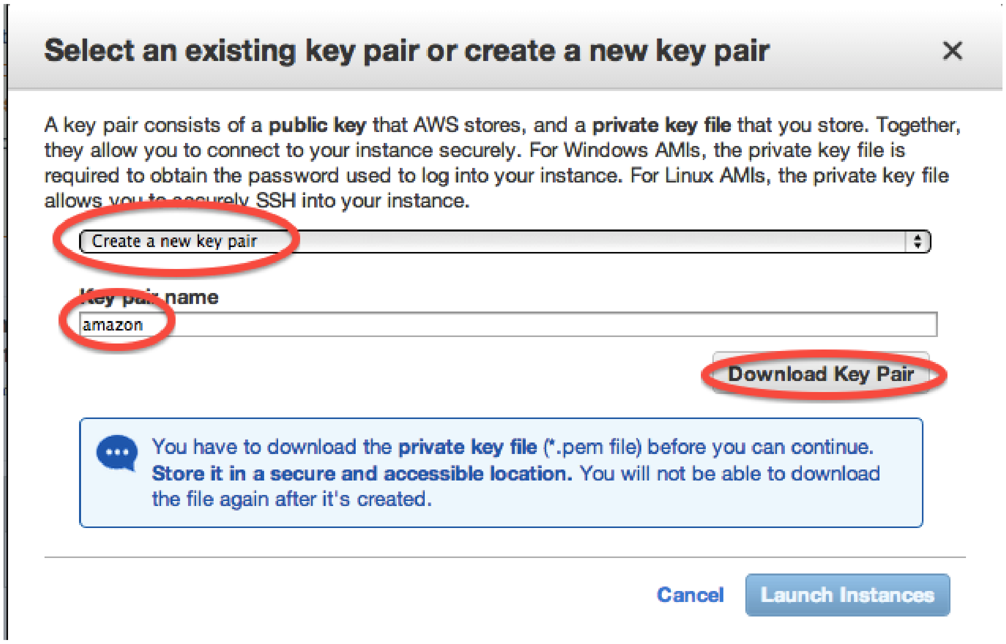
Security Groups

Security group name launch-wizard-35

Description launch-wizard-35 created 2015-05-04T07:00:43.236-07:00

(First time through) generate a new key pair

If you don’t have any key pairs, enter a key pair name and then download a key pair. Then click Launch Instance.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Create a new key pair

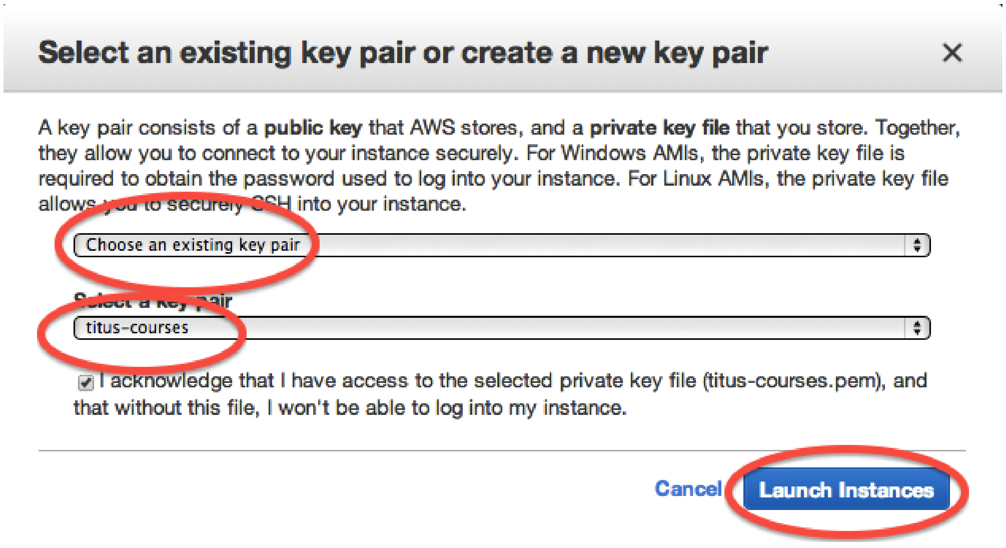
Key pair name

amazon

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

(Next times through) select an existing key pair

Select a key pair and click ‘Launch’.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Choose an existing key pair

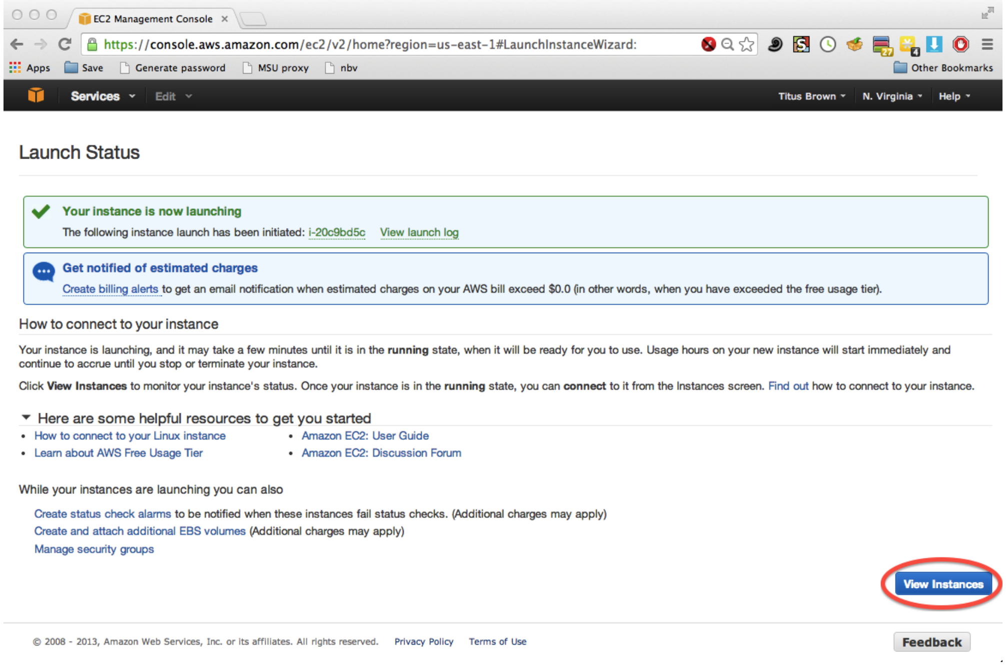
Select a key pair

titus-courses

☑ I acknowledge that I have access to the selected private key file (titus-courses.pem), and that without this file, I won't be able to log into my instance.

Cancel
Launch Instances

Click on View Instances



EC2 Management Console
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:
Apps
Save
Generate password
MSU proxy
nbv
Other Bookmarks
Services
Edit
Titus Brown
N. Virginia
Help

Launch Status

Your instance is now launching

The following instance launch has been initiated: i-20c9bd5c View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed $0.0 (in other words, when you have exceeded the free usage tier).

How to connect to your instance

Your instance is launching, and it may take a few minutes until it is in the **running** state, when it will be ready for you to use. Usage hours on your new instance will start immediately and continue to accrue until you stop or terminate your instance.

Click **View Instances** to monitor your instance's status. Once your instance is in the **running** state, you can **connect** to it from the Instances screen. Find out how to connect to your instance.

Here are some helpful resources to get you started

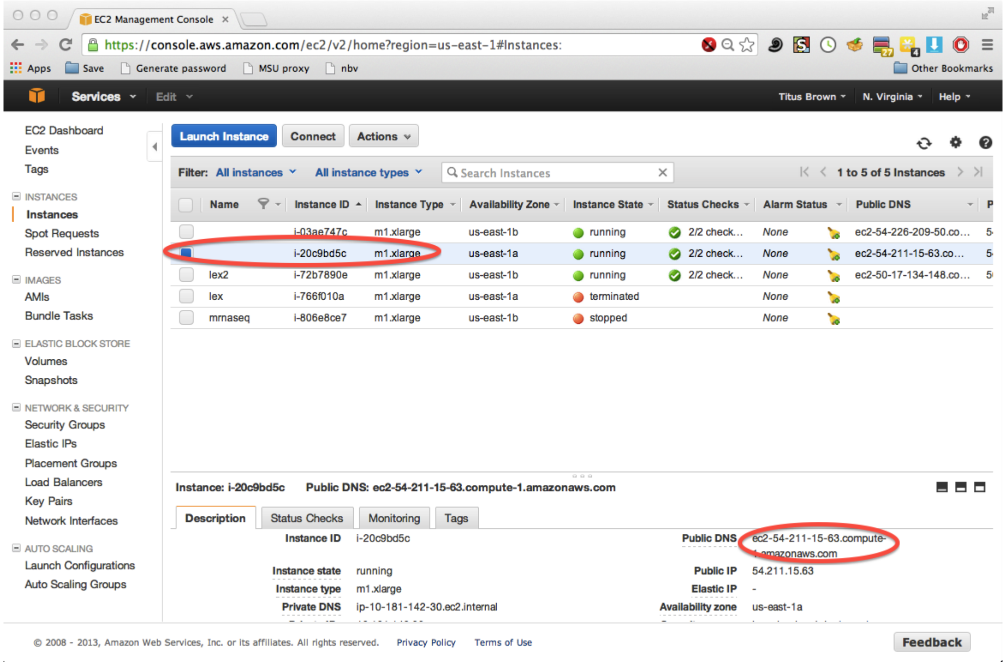
- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
Create and attach additional EBS volumes (Additional charges may apply)
Manage security groups

View Instances
© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Feedback

Select the public DNS name for later use



EC2 Management Console
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:
Services
Edit
Titus Brown
N. Virginia
Help
EC2 Dashboard
Events
Tags
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Network Interfaces
AUTO SCALING
Launch Configurations
Auto Scaling Groups
Launch Instance
Connect
Actions
Filter: All instances
All instance types
Search Instances
1 to 5 of 5 Instances
Description
Status Checks
Monitoring
Tags
Privacy Policy
Terms of Use
Feedback

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-03ae747c	m1.xlarge	us-east-1b	running	2/2 check…	None	ec2-54-226-209-50.co…
	i-20c9bd5c	m1.xlarge	us-east-1a	running	2/2 check…	None	ec2-54-211-15-63.co…
lex2	i-72b7890e	m1.xlarge	us-east-1b	running	2/2 check…	None	ec2-50-17-134-148.co…
lex	i-766f010a	m1.xlarge	us-east-1a	terminated		None	
mrnaseq	i-806e8ce7	m1.xlarge	us-east-1b	stopped		None	

Instance: i-20c9bd5c Public DNS: ec2-54-211-15-63.compute-1.amazonaws.com

Instance ID	i-20c9bd5c	Public DNS	ec2-54-211-15-63.compute-1.amazonaws.com
Instance state	running	Public IP	54.211.15.63
Instance type	m1.xlarge	Elastic IP	-
Private DNS	ip-10-181-142-30.ec2.internal	Availability zone	us-east-1a

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.

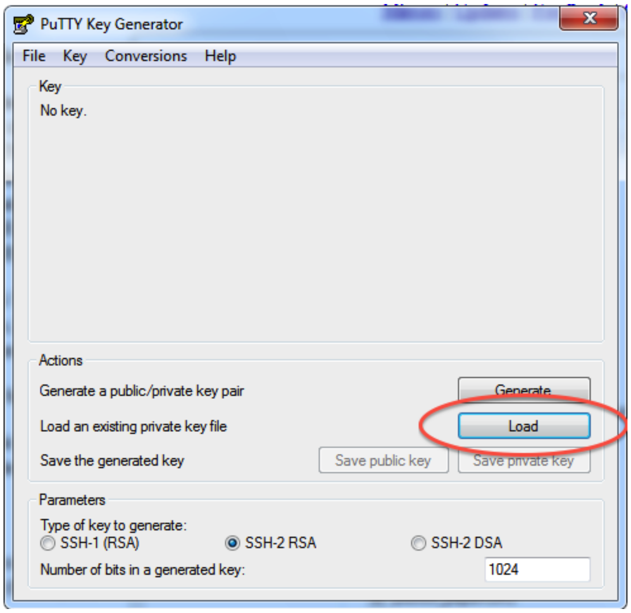
3.1.2 Logging into your new instance "in the cloud" (Windows version)

First, Download Putty and Puttygen from here: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Generate a ppk file from your pem file

(You only need to do this once for each key!)

Open puttygen; select "Load".



PuTTY Key Generator
File
Key
Conversions
Help
Key
Actions
Generate
Load
Save public key
Save private key
Parameters

No key.

Generate a public/private key pair

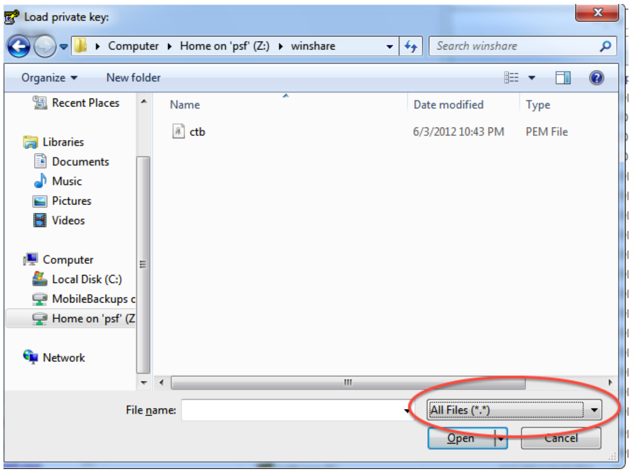
Load an existing private key file

Save the generated key

Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA

Number of bits in a generated key: 1024

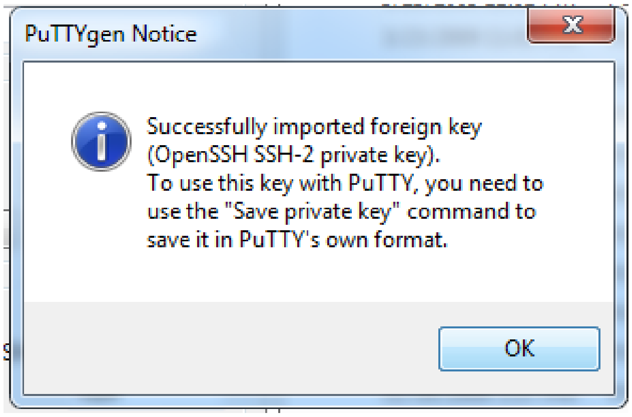
Find and load your '.pem' file; it's probably in your Downloads folder. Note, you have to select 'All files' on the bottom.


Load private key:
Computer ▸ Home on 'psf' (Z:) ▸ winshare
Search winshare
Organize
New folder
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
MobileBackups c
Home on 'psf' (Z
Network
File name:
Open
Cancel

Name	Date modified	Type
ctb	6/3/2012 10:43 PM	PEM File

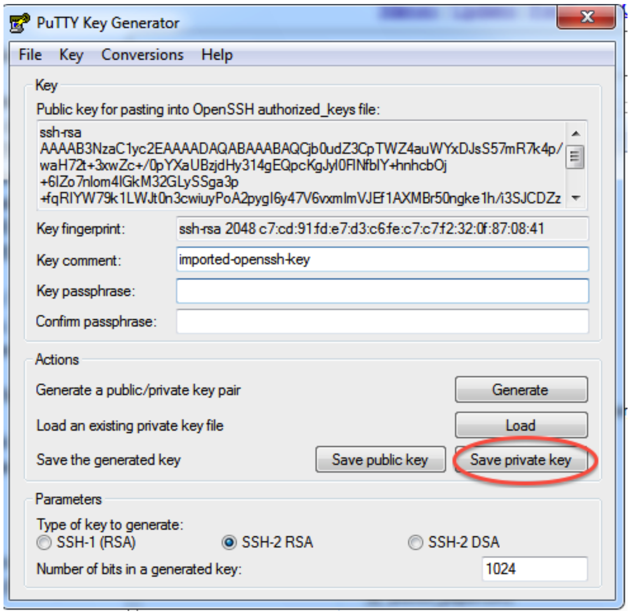
All Files (*.*)

Load it.


PuTTYgen Notice
OK

Successfully imported foreign key (OpenSSH SSH-2 private key).
To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

Now, "save private key". Put it somewhere easy to find.


PuTTY Key Generator
File
Key
Conversions
Help
Key
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCjb0udZ3CpTWZ4auWYxDJsS57mR7k4p/
waH72t+3xwZc+/0pYXaUBzjdHy314gEQpcKgJyl0FINfbIY+hnhcbOj
+6IZo7nlom4IGkM32GLySSga3p
+fqRIYW79k1LWJt0n3cwiuyPoA2pygI6y47V6vxmImVJEf1AXMBr50ngke1h/i3SJCDZz
```

Key fingerprint: ssh-rsa 2048 c7:cd:91:fd:e7:d3:c6:fe:c7:c7:f2:32:0f:87:08:41

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

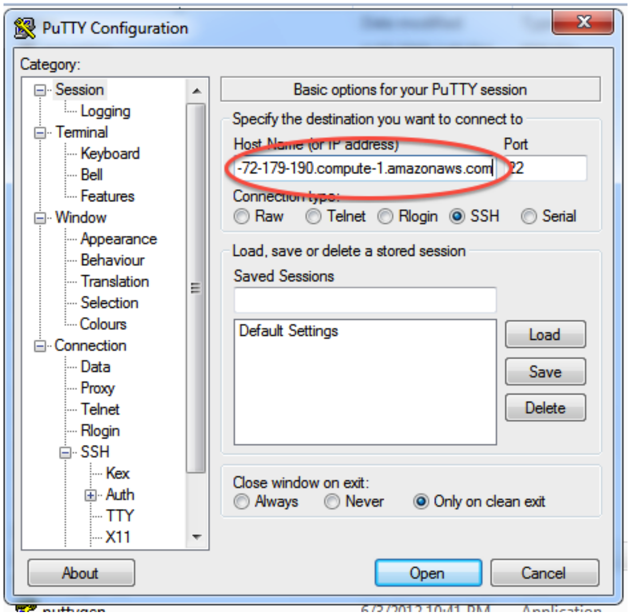
Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA

Number of bits in a generated key: 1024

Now that you've generated your PPK file from your PEM file, you can log in. To do that...

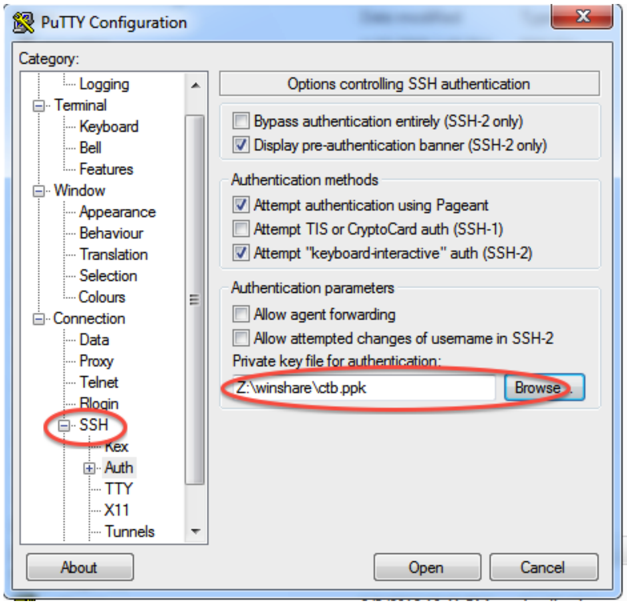
Logging into your EC2 instance with Putty

Open up putty, and enter your hostname into the Host Name box.


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Basic options for your PuTTY session
Specify the destination you want to connect to
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
About
Open
Cancel

Host Name (or IP address): -72-179-190.compute-1.amazonaws.com
Port: 22
Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial
Close window on exit: ○ Always ○ Never ● Only on clean exit

Now, go find the 'SSH' section and enter your ppk file (generated above by puttygen). Then select 'Open'.

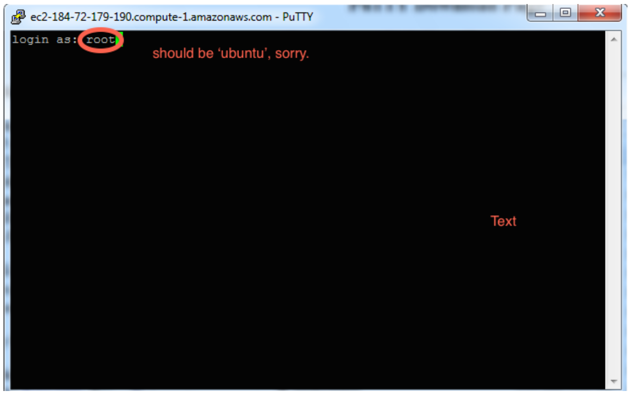

PuTTY Configuration
Category:
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Tunnels
Options controlling SSH authentication
Browse...
About
Open
Cancel

☐ Bypass authentication entirely (SSH-2 only)
☑ Display pre-authentication banner (SSH-2 only)

Authentication methods
☑ Attempt authentication using Pageant
☐ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

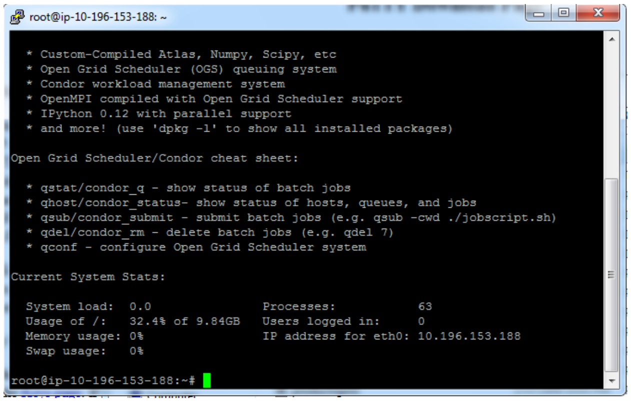
Authentication parameters
☐ Allow agent forwarding
☐ Allow attempted changes of username in SSH-2
Private key file for authentication: Z:\winshare\ctb.ppk

Log in as "ubuntu".


ec2-184-72-179-190.compute-1.amazonaws.com - PuTTY
should be 'ubuntu', sorry.
Text

```
login as: root
```

Declare victory!



```
  * Custom-Compiled Atlas, Numpy, Scipy, etc
  * Open Grid Scheduler (OGS) queuing system
  * Condor workload management system
  * OpenMPI compiled with Open Grid Scheduler support
  * IPython 0.12 with parallel support
  * and more! (use 'dpkg -l' to show all installed packages)

Open Grid Scheduler/Condor cheat sheet:

  * qstat/condor_q - show status of batch jobs
  * qhost/condor_status- show status of hosts, queues, and jobs
  * qsub/condor_submit - submit batch jobs (e.g. qsub -cwd ./jobscript.sh)
  * qdel/condor_rm - delete batch jobs (e.g. qdel 7)
  * qconf - configure Open Grid Scheduler system

Current System Stats:

  System load:  0.0               Processes:           63
  Usage of /:   32.4% of 9.84GB   Users logged in:     0
  Memory usage: 0%                IP address for eth0: 10.196.153.188
  Swap usage:   0%

root@ip-10-196-153-188:~#
```

3.1.3 Logging into your new instance "in the cloud" (Mac version)

OK, so you've created a running computer. How do you get to it?

The main thing you'll need is the network name of your new computer. To retrieve this, go to the instance view and click on the instance, and find the "Public DNS". This is the public name of your computer on the Internet.

Copy this name, and connect to that computer with ssh under the username 'ubuntu', as follows.

First, find your private key file; it's the .pem file you downloaded when starting up your EC2 instance. It should be in your Downloads folder. Move it onto your desktop and rename it to 'amazon.pem'.

Next, start Terminal (in Applications... Utilities...) and type:

```
chmod og-rwx ~/Desktop/amazon.pem
```

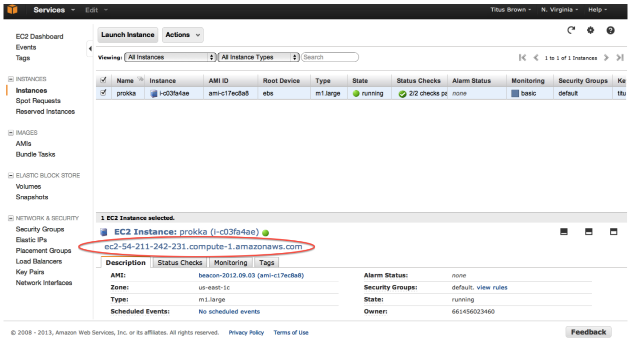
to set the permissions on the private key file to "closed to all evildoers".

Then type:

```
ssh -i ~/Desktop/amazon.pem ubuntu@ec2-???-???-???-???.compute-1.amazonaws.com
```

Here, you're logging in as user 'ubuntu' to the machine 'ec2-174-129-122-189.compute-1.amazonaws.com' using the authentication key located in 'amazon.pem' on your Desktop.

Note, you have to replace the stuff after the '@' sign with the name of the host; see the red circle in:


Services
Edit
Titus Brown
N. Virginia
Help
EC2 Dashboard
Events
Tags
INSTANCES
Instances
Spot Requests
Reserved Instances
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Load Balancers
Key Pairs
Network Interfaces
Launch Instance
Actions
Viewing: All Instances
All Instance Types
Search
1 to 1 of 1 Instances
Description
Status Checks
Monitoring
Tags
© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Feedback

☑	Name	Instance	AMI ID	Root Device	Type	State	Status Checks	Alarm Status	Monitoring	Security Groups	Ke
☑	prokka	i-c03fa4ae	ami-c17ec8a8	ebs	m1.large	running	2/2 checks pa	none	basic	default	titu

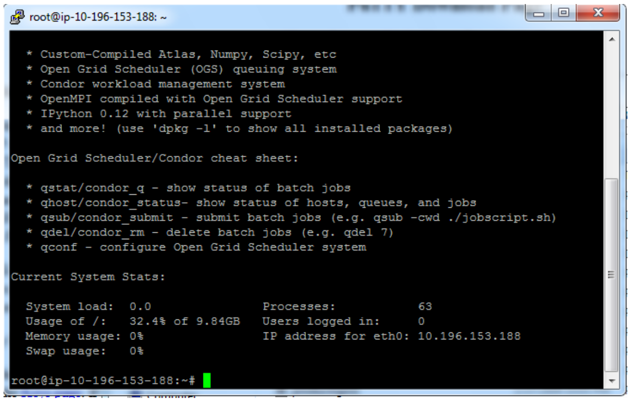
1 EC2 Instance selected.

EC2 Instance: prokka (i-c03fa4ae)

ec2-54-211-242-231.compute-1.amazonaws.com

AMI:	beacon-2012.09.03 (ami-c17ec8a8)	**Alarm Status:**	none
Zone:	us-east-1c	**Security Groups:**	default. view rules
Type:	m1.large	**State:**	running
Scheduled Events:	No scheduled events	**Owner:**	661456023460

At the end you should see text and a prompt that look like this:


root@ip-10-196-153-188: ~

```
  * Custom-Compiled Atlas, Numpy, Scipy, etc
  * Open Grid Scheduler (OGS) queuing system
  * Condor workload management system
  * OpenMPI compiled with Open Grid Scheduler support
  * IPython 0.12 with parallel support
  * and more! (use 'dpkg -l' to show all installed packages)

Open Grid Scheduler/Condor cheat sheet:

  * qstat/condor_q - show status of batch jobs
  * qhost/condor_status- show status of hosts, queues, and jobs
  * qsub/condor_submit - submit batch jobs (e.g. qsub -cwd ./jobscript.sh)
  * qdel/condor_rm - delete batch jobs (e.g. qdel 7)
  * qconf - configure Open Grid Scheduler system

Current System Stats:

  System load:  0.0              Processes:           63
  Usage of /:   32.4% of 9.84GB   Users logged in:     0
  Memory usage: 0%               IP address for eth0: 10.196.153.188
  Swap usage:   0%

root@ip-10-196-153-188:~#
```

A final checklist:

- EC2 instance is running;
- used ubuntu 14.04
- NOT a micro instance (m3.xlarge, or bigger);

3.1.4 Amazon Web Services reference material

Instance types

Instance costs

Short read quality and trimming

Note: Reminder: if you're on Windows, you should install mobaxterm.

OK, you should now be logged into your Amazon computer! How exciting!

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:

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

This makes '/mnt' a place where we can put data and working files.

Next, let's install a few things:

```
sudo apt-get update
sudo apt-get install -y trimmomatic fastqc python-pip python-dev
```

These are the Trimmomatic and FastQC programs, which we'll use below, along with some software prerequisites that we'll need for other things below.

4.2 Data source

We're going to be using a subset of data from Tulin et al., 2013, a paper looking at early transcription in the organism *Nematostella vectensis*, the sea anemone.

4.3 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 http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
```

Now if you type:

```
ls -l
```

you should see something like:

```
-r--r--r-- 1 ubuntu ubuntu   7874107 Dec 14  2013 0Hour_ATCACG_L002_R1_001.extract.fastq.gz
-r--r--r-- 1 ubuntu ubuntu   7972058 Dec 14  2013 0Hour_ATCACG_L002_R1_002.extract.fastq.gz
...
```

These are subsets of the original data, where we selected for reads that belong to a few particular transcripts.

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.

4.4 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 0Hour_ATCACG_L002_R1_001.extract.fastq.gz
```

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

Question:

- why do the files have DNA in the name?
- why are there R1 and R2 in the file names?
- why don't we combine all the files?

Links:

- FASTQ Format

4.5 2. FastQC

We're going to use FastQC to summarize the data. We already installed 'fastqc' on our computer - that's what the 'apt-get install' did, above.

Now, run FastQC on two files:

```
fastqc 0Hour_ATCACG_L002_R1_001.extract.fastq.gz
fastqc 0Hour_ATCACG_L002_R2_001.extract.fastq.gz
```

Now type ‘ls’:

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

to list the files, and you should see:

```
0Hour_ATCACG_L002_R1_001.extract_fastqc
0Hour_ATCACG_L002_R1_001.extract_fastqc.zip
0Hour_ATCACG_L002_R2_001.extract_fastqc
0Hour_ATCACG_L002_R2_001.extract_fastqc.zip
```

We are *not* going to show you how to look at these files right now - you need to copy them to your local computer to do that. We’ll show you that tomorrow. But! we can show you what they look like, because I’ve made copiesd of them for you:

- 0Hour_ATCACG_L002_R1_001.extract_fastqc/fastqc_report.html
- 0Hour_ATCACG_L002_R2_001.extract_fastqc/fastqc_report.html

Questions:

- What should you pay attention to in the FastQC report?
- Which is “better”, R1 or R2? And why?

Links:

- FastQC
- FastQC tutorial video

4.6 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 0Hour_ATCACG_L002_R1_001.extract.fastq.gz \
              0Hour_ATCACG_L002_R2_001.extract.fastq.gz \
     0Hour_ATCACG_L002_R1_001.qc.fq.gz s1_se \
     0Hour_ATCACG_L002_R2_001.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:

```
...
Quality encoding detected as phred33
Input Read Pairs: 140557 Both Surviving: 138775 (98.73%) Forward Only Surviving: 1776 (1.26%) Reverse
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 RNAseq trimming strategies, see MacManes, 2014.

Links:

- Trimmomatic

4.7 4. FastQC again

Run FastQC again on the trimmed files:

```
fastqc 0Hour_ATCACG_L002_R1_001.qc.fq.gz
fastqc 0Hour_ATCACG_L002_R2_001.qc.fq.gz
```

And now view my copies of these files:

- 0Hour_ATCACG_L002_R1_001.qc.fq_fastqc/fastqc_report.html
- 0Hour_ATCACG_L002_R2_001.qc.fq_fastqc/fastqc_report.html

Let's take a look at the output files:

```
less 0Hour_ATCACG_L002_R1_001.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!?

4.8 5. Trim the rest of the sequences

First download the rest of the data:

```
cd /mnt/data
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
```

And link it in:

```
cd /mnt/work
ln -fs /mnt/data/*.fastq.gz .
```

Now we have a lot of files – and we really don't want to trim each and every one of them by typing in a command for each pair! Here we'll make use of a great feature of the UNIX command line – the ability to automate such tasks.

Here's a for loop that you can run - we'll walk through what it does while it's running:

```
rm -f orphans.fq

for filename in *_R1_*.extract.fastq.gz
do
      # first, make the base by removing .extract.fastq.gz
      base=$(basename $filename .extract.fastq.gz)
      echo $base

      # now, construct the R2 filename by replacing R1 with R2
      baseR2=${base/_R1_/_R2_}
      echo $baseR2

      # finally, run Trimmomatic
      TrimmomaticPE ${base}.extract.fastq.gz ${baseR2}.extract.fastq.gz \
         ${base}.qc.fq.gz s1_se \
         ${baseR2}.qc.fq.gz s2_se \
         ILLUMINACLIP:TruSeq2-PE.fa:2:40:15 \
         LEADING:2 TRAILING:2 \
         SLIDINGWINDOW:4:2 \
         MINLEN:25

      # save the orphans
      cat s1_se s2_se >> orphans.fq
done
```

Things to mention –

- # are comments;
- anywhere you see a '$' is replaced by the value of the variable after it, so e.g. $filename is replaced by each of the files matching _*R1*_.extract.fastq.gz, once for each time through the loop;
- we have to do complicated things to the filenames to get this to work, which is what the ${base/_R1_/_R2_} stuff is about.
- what's with 'orphans.fq'??

Questions:

- **how do you figure out if it's working?**
    - copy/paste it from Word

  - put in lots of echo
  - edit one line at a time
- how on earth do you figure out how to do this?!

4.9 6. Interleave the sequences

Next, we need to take these R1 and R2 sequences and convert them into interleaved form ,for the next step. To do this, we'll use scripts from the khmer package, which we need to install:

```
sudo pip install -U setuptools
sudo pip install khmer==1.3
```

Now let's use a for loop again - you might notice this is only a minor modification of the previous for loop...

```
for filename in *_R1_*.qc.fq.gz
do
      # first, make the base by removing .extract.fastq.gz
      base=$(basename $filename .qc.fq.gz)
      echo $base

      # now, construct the R2 filename by replacing R1 with R2
      baseR2=${base/_R1_/_R2_}
      echo $baseR2

      # construct the output filename
      output=${base/_R1_/}.pe.qc.fq.gz

      interleave-reads.py ${base}.qc.fq.gz ${baseR2}.qc.fq.gz | \
          gzip > $output
done

gzip orphans.fq
```

Next: Running digital normalization

CHAPTER 5

Running digital normalization

Next, we're going to apply abundance normalization to the data – known as "digital normalization", this approach was developed by our lab to make it possible to assemble large data sets more quickly and easily. You can read more about it in Brown et al., 2012, and also see some of its affects on transcriptome assembly in Lowe et al., 2014.

Digital normalization works by eliminating high abundance reads that are unnecessary for assembly.

First, we'll run it on the interleaved files we generated in the previous section:

```
cd /mnt/work
normalize-by-median.py -k 20 -p -C 20 -N 4 -x 2e9 -s normC20k20.ct *.pe.qc.fq.gz
```

(These parameters should work for essentially all mRNAseq data sets; see the khmer documentation for more information.)

Next, run diginorm on the orphaned reads (from trimming):

```
normalize-by-median.py -l normC20k20.ct -s normC20k20.ct orphans.fq.gz
```

Do k-mer abundance trimming on the reads, which will eliminate the majority of the errors (thus further decreasing the memory requirements) –:

```
filter-abund.py -V normC20k20.ct *.keep
```

See our paper *Zhang et al., 2014 <http://www.ncbi.nlm.nih.gov/pubmed/25062443>*'__, Table 3, for more information on k-mer trimming effects.

Now, take all of the paired-end files and split them into paired and orphaned reads:

```
for filename in *.pe.*.keep.abundfilt
do
   extract-paired-reads.py $filename
done
```

Put all the orphaned reads in one place:

```
cat *.se orphans.fq.gz.keep.abundfilt | gzip > orphans.dn.fq.gz
```

And now rename the paired-end files to something nice:

```
for filename in *.pe.qc.fq.gz.keep.abundfilt.pe
do
   base=$(basename $filename .pe.qc.fq.gz.keep.abundfilt.pe)
   output=${base}.dn.fq.gz
   gzip -c $filename > $output
done
```

Now, if you type:

```
ls *.dn.fq.gz
```

you'll see all of the files that you need to move on to the next step –

```
0Hour_ATCACG_L002001.dn.fq.gz  6Hour_CGATGT_L002002.dn.fq.gz
0Hour_ATCACG_L002002.dn.fq.gz  6Hour_CGATGT_L002003.dn.fq.gz
0Hour_ATCACG_L002003.dn.fq.gz  6Hour_CGATGT_L002004.dn.fq.gz
0Hour_ATCACG_L002004.dn.fq.gz  6Hour_CGATGT_L002005.dn.fq.gz
0Hour_ATCACG_L002005.dn.fq.gz  orphans.dn.fq.gz
6Hour_CGATGT_L002001.dn.fq.gz
```

Let's remove some of the detritus before moving on:

```
rm *.pe *.se *.abundfilt *.keep
rm normC20k20.ct
```

Next: Running the actual assembly

Running the actual assembly

Now we'll assemble all of these reads into a transcriptome, using the Trinity de novo transcriptome assembler.

First, install some prerequisites for Trinity:

```
sudo apt-get -y install bowtie samtools
```

Next, install Trinity v2.0.6:

```
cd
curl -L https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.0.6.tar.gz > trinity.tar.gz
tar xzf trinity.tar.gz
mv trinityrnaseq* trinity/

cd trinity
make
```

Go into the work directory, and prepare the data:

```
cd /mnt/work
for i in *.dn.fq.gz
do
   split-paired-reads.py $i
done

cat *.1 > left.fq
cat *.2 > right.fq
```

Now, run the Trinity assembler:

```
~/trinity/Trinity --left left.fq --right right.fq --seqType fq --max_memory 10G --bypass_java_version
```

This will give you an output file `trinity_out_dir/Trinity.fasta`, which you can get stats on like so:

```
curl -L -O https://github.com/ged-lab/khmer/raw/v1.3/sandbox/assemstats3.py
python assemstats3.py 300 trinity_out_dir/Trinity.fasta
```

Change the filename and rename all the sequences:

```
gzip -c trinity_out_dir/Trinity.fasta > trinity-nematostella-raw.fa.gz
curl -O http://2015-may-nonmodel.readthedocs.org/en/dev/_static/rename-with-partitions.py
chmod u+x rename-with-partitions.py
./rename-with-partitions.py nema trinity-nematostella-raw.fa.gz
```

This last command will give you `trinity-nematostella-raw.renamed.fasta.gz`, which contains all of the renamed sequences.

Next: BLASTing your assembled data

CHAPTER 7

BLASTing your assembled data

First, install a few prerequisites:

```
sudo apt-get -y install lighttpd blast2 git-core zlib1g-dev
```

Next, grab things needed for the BLAST server:

```
sudo pip install pygr whoosh Pillow Jinja2 \
    git+https://github.com/ctb/pygr-draw.git screed
sudo ln -s /usr/bin/blastall /usr/local/bin/
```

Install the BLAST server and configure it:

```
cd
git clone https://github.com/ctb/blastkit.git -b 2015-may-nonmodel
sudo ./blastkit/configure-lighttpd.sh

cd blastkit/www
sudo ln -fs $PWD /var/www/blastkit

mkdir files
chmod a+rxwt files
chmod +x /home/ubuntu

cd /home/ubuntu/blastkit
python ./check.py
```

Now, copy in your newly created transcriptome:

```
cd /mnt/work
gunzip -c trinity-nematostella-raw.renamed.fasta.gz > /home/ubuntu/blastkit/db/db.fa

cd /home/ubuntu/blastkit
formatdb -i db/db.fa -o T -p F
python index-db.py db/db.fa
```

You can now access your BLAST server at http://<amazon machine name>/blastkit/.

Note that you will need to enable HTTP access on your Amazon firewall settings; see `../amazon/enable-http`.

If you want to use something that will get you results with the test data set, try BLASTing zebrafish lethal giant larvae homolog.

CHAPTER 8

Annotation of denovo transcriptome

8.1 Identify the Gene/Transcript relationships:

we can generate this file like so:

```
cd /mnt/work
~/trinity/util/support_scripts/get_Trinity_gene_to_trans_map.pl trinity_out_dir/Trinity.fasta >  Tri
```

Let's have a look on the map:

```
less Trinity.fasta.gene_trans_map
```

Components, genes and isoforms:

- The different (i's) that correspond to the same (g) represent isoforms
- The different (g's) could represent different genes (or parts of genes)
- The component (TR|c) often contain related genes (paralogs or gene fragments).

  Check the Trinityseq forum for more details

8.2 Generate the longest-ORF peptide candidates from the Trinity Assembly:

We need to install Transdecoder to do this job:

```
cd
sudo cpan URI::Escape
```

Note: type yes for all interactive questions

```
curl -L https://github.com/TransDecoder/TransDecoder/archive/2.0.1.tar.gz > transdecoder.tar.gz
tar xzf transdecoder.tar.gz
mv TransDecoder* TransDecoder
cd TransDecoder
make
```

Now we can run the Transdecoder software to identify the longest-ORF peptide:

```
cd /mnt/work
~/TransDecoder/TransDecoder.LongOrfs -t trinity_out_dir/Trinity.fasta
```

Check the Transdecoder output:

```
less Trinity.fasta.transdecoder_dir/longest_orfs.pep
```

8.3 Capturing BLAST Homologies

Install BLAST+ (http://www.ncbi.nlm.nih.gov/books/NBK52640/):

```
sudo apt-get install -y ncbi-blast+
```

Get the required sequence databases and prepare local blast databases

1. SwissProt database: The UniProt Knowledgebase which include the Manually annotated proteins:

   ```
   cd /mnt/work
   wget ftp://ftp.broadinstitute.org/pub/Trinity/Trinotate_v2.0_RESOURCES/uniprot_sprot.trinotate_v
   mv uniprot_sprot.trinotate_v2.0.pep.gz uniprot_sprot.trinotate.pep.gz
   gunzip uniprot_sprot.trinotate.pep.gz
   makeblastdb -in uniprot_sprot.trinotate.pep -dbtype prot
   ```

   Run blast to find homologies.

   (a) search Trinity transcripts:

   ```
   blastx -query trinity_out_dir/Trinity.fasta -db uniprot_sprot.trinotate.pep -num_threads 4 -
   ```

   (b) search Transdecoder-predicted proteins:

   ```
   blastp -query Trinity.fasta.transdecoder_dir/longest_orfs.pep -db uniprot_sprot.trinotate.pe
   ```

2. Optional: Uniref90 which provides clustered sets of protein sequences in a way such that each cluster is composed of sequences that have at least 90% sequence identity to, and 80% overlap with, the longest sequence:

   ```
   wget ftp://ftp.broadinstitute.org/pub/Trinity/Trinotate_v2.0_RESOURCES/uniprot_uniref90.trinotat
   mv uniprot_uniref90.trinotate_v2.0.pep.gz uniprot_uniref90.trinotate.pep.gz
   gunzip uniprot_uniref90.trinotate.pep.gz
   makeblastdb -in uniprot_uniref90.trinotate.pep -dbtype prot
   ```

   perform similar searches using uniref90 as the target database, rename output files accordingly:

   ```
   blastx -query trinity_out_dir/Trinity.fasta -db uniprot_uniref90.trinotate.pep -num_threads 4 -m
   blastp -query Trinity.fasta.transdecoder_dir/longest_orfs.pep -db uniprot_uniref90.trinotate.pep
   ```

I have ran them overnight already. You can download these files to save time:

```
wget https://github.com/ngs-docs/2015-may-nonmodel/blob/master/_static/uniref90.blastp.outfmt6
wget https://github.com/ngs-docs/2015-may-nonmodel/blob/master/_static/uniref90.blastx.outfmt6
```

8.4 Characterization of functional annotation features

1. identify protein domains: we need to install HMMER and download the Pfam domains database:

```
     sudo apt-get install -y hmmer
```

Then we can run hmmer to identify the protein domains:

```
cd /mnt/work
wget ftp://ftp.broadinstitute.org/pub/Trinity/Trinotate_v2.0_RESOURCES/Pfam-A.hmm.gz
gunzip Pfam-A.hmm.gz
hmmpress Pfam-A.hmm
hmmscan --cpu 4 --domtblout TrinotatePFAM.out Pfam-A.hmm Trinity.fasta.transdecoder_dir/longest_orfs
```

2. We can predict other features like
    - signal peptides: using signalP
    - transmembrane regions: using tmHMM
    - rRNA transcripts: using RNAMMER

8.5 Integration of all annotations into one database

install Trinotate:

```
cd
curl -L https://github.com/Trinotate/Trinotate/archive/v2.0.2.tar.gz > trinotate.tar.gz
tar xzf trinotate.tar.gz
mv Trinotate* Trinotate
```

install sqlite

```
sudo apt-get install sqlite3
```

We need also the DBI perl package:

```
sudo cpan DBI
sudo cpan DBD::SQLite
```

Retrieve the Trinotate Pre-generated Resource SQLite database A pregenerated sqlite database that contains Uniprot (swissprot and uniref90)-related annotation information is available from the Trinity ftp site:

```
cd /mnt/work
wget "ftp://ftp.broadinstitute.org/pub/Trinity/Trinotate_v2.0_RESOURCES/Trinotate.sprot_uniref90.2015
gunzip Trinotate.sqlite.gz
```

Load transcripts and coding regions. We have three data types:

1. Transcript sequences (de novo assembled transcripts or reference transcripts)
2. Protein sequences (currently as defined by TransDecoder)
3. Gene/Transcript relationships

```
~/Trinotate/Trinotate Trinotate.sqlite init --gene_trans_map Trinity.fasta.gene_trans_map --transcrip
```

Loading BLAST homologies:

```
~/Trinotate/Trinotate Trinotate.sqlite LOAD_swissprot_blastp blastp.outfmt6
~/Trinotate/Trinotate Trinotate.sqlite LOAD_swissprot_blastx blastx.outfmt6
```

Optional: load Uniref90 blast hits:

```
~/Trinotate/Trinotate Trinotate.sqlite LOAD_trembl_blastp uniref90.blastp.outfmt6
~/Trinotate/Trinotate Trinotate.sqlite LOAD_trembl_blastx uniref90.blastx.outfmt6
```

Optional: Loading functional annotation features:

```
~/Trinotate/Trinotate Trinotate.sqlite LOAD_pfam TrinotatePFAM.out
```

8.6 Output an Annotation Report

```
~/Trinotate/Trinotate Trinotate.sqlite report -E 0.0001 > trinotate_annotation_report.xls
```

There are 2 arguments that we can use to control the accuracy of annotation

-E <float> : maximum E-value for reporting best blast hit and associated annotations.

–pfam_cutoff <string>

1. ‘DNC’ : domain noise cutoff (default)
2. ‘DGC’ : domain gathering cutoff
3. ‘DTC’ : domain trusted cutoff
4. ‘SNC’ : sequence noise cutoff
5. ‘SGC’ : sequence gathering cutoff
6. ‘STC’ : sequence trusted cutoff

let us see the output. Open a new shell:

```
scp -i YOUR_SECURITY_KEY.pem ubuntu@YOUR_AMAZONE_INSTANCE_ADDRESS:/mnt/work/trinotate_annotation_repo
```

CHAPTER 9

Quantification and Differential Expression

First, make sure you've downloaded all the original raw data:

```
cd /mnt/data
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/0Hour_ATCACG_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
curl -O -L http://dib-training.ucdavis.edu.s3.amazonaws.com/mRNAseq-non-2015-05-04/6Hour_CGATGT_L002_
```

...and link it in:

```
cd /mnt/work
ln -fs /mnt/data/*.fastq.gz .
```

9.1 Download Express

Now, get express:

```
cd
curl -L http://bio.math.berkeley.edu/eXpress/downloads/express-1.5.1/express-1.5.1-linux_x86_64.tgz >
tar xzf express.tar.gz
```

9.2 Align Reads with Bowtie

Next, build an index file for your assembly:

```
cd /mnt/work
gunzip -c trinity-nematostella-raw.renamed.fasta.gz > trinity-nematostella-raw.renamed.fasta
bowtie-build --offrate 1 trinity-nematostella-raw.renamed.fasta trinity-nematostella-raw.renamed
```

Using the index we built, we'll align the reads from a few of our samples back to our assembly:

```
bowtie -aS -X 800 --offrate 1 trinity-nematostella-raw.renamed \
    -1 <(zcat 0Hour_ATCACG_L002_R1_001.extract.fastq.gz) \
    -2 <(zcat 0Hour_ATCACG_L002_R2_001.extract.fastq.gz) \
    > 0Hour_ATCACG_L002_001.extract.sam

bowtie -aS -X 800 --offrate 1 trinity-nematostella-raw.renamed -1 <(zcat 0Hour_ATCACG_L002_R1_002.ext

bowtie -aS -X 800 --offrate 1 trinity-nematostella-raw.renamed -1 <(zcat 6Hour_CGATGT_L002_R1_001.ext
bowtie -aS -X 800 --offrate 1 trinity-nematostella-raw.renamed -1 <(zcat 6Hour_CGATGT_L002_R1_002.ext
```

9.3 Quantify Expression using eXpress

Finally, using eXpress, we'll get abundance estimates for our transcripts. eXpress uses a probabilistic model to efficiently assign mapped reads to isoforms and estimate expression level (see the website for additional details and relevant publications):

```
~/express-1.5.1-linux_x86_64/express --no-bias-correct -o 0Hour_ATCACG_L002_001.extract.sam-express
~/express-1.5.1-linux_x86_64/express --no-bias-correct -o 0Hour_ATCACG_L002_002.extract.sam-express

~/express-1.5.1-linux_x86_64/express --no-bias-correct -o 6Hour_CGATGT_L002_001.extract.sam-express
~/express-1.5.1-linux_x86_64/express --no-bias-correct -o 6Hour_CGATGT_L002_002.extract.sam-express
```

This will put the results in a new set of folders named like *<condition>_<barcode>_L002_<replicate>.extract.sam-express*. Each contains a file called *results.xprs* with the results. We'll look at the first ten lines of one of the files using the *head* command:

```
head 0Hour_ATCACG_L002_001.extract.sam-express/results.xprs
```

You should see something like this:

```
bundle_id   target_id   length  eff_length  tot_counts  uniq_counts est_counts  eff_counts  ambig_di
1   nema.id7.tr4    269 0.000000    0   0   0.000000    0.000000    0.000000e+00    0.000000e+00
2   nema.id1.tr1    811 508.137307  1301    45  158.338092  252.711602  4.777128e+01    4.816246e+02
2   nema.id2.tr1    790 487.144836  1845    356 1218.927626 1976.727972 1.111471e+02    8.063959e+01
2   nema.id3.tr1    852 549.122606  1792    3   871.770849  1352.610064 5.493335e+01    5.818711e+01
2   nema.id4.tr1    675 372.190166  1005    20  88.963433   161.343106  2.836182e+01    3.767281e+02
3   nema.id62.tr13  2150    1846.657210 9921    9825    9919.902997 11549.404689    1.704940e+03
3   nema.id63.tr13  406 103.720396  360 270 271.097003  1061.173959 1.934732e+02    1.567940e+04
3   nema.id61.tr13  447 144.526787  6   0   0.000000    0.000000    2.246567e+04    2.246565e+10
4   nema.id21.tr8   2075    1771.684102 2782    58  958.636395  1122.756883 1.223148e+02    2.476298
```

9.4 Differential Expression

First, install R and edgeR:

```
sudo apt-get install -y r-base-core r-bioc-edger csvtool
```

Now, we extract the columns we need from the eXpress outputs and convert it to the appropriate format:

```
csvtool namedcol -t TAB target_id,est_counts 0Hour_ATCACG_L002_001.extract.sam-express/results.xprs
csvtool namedcol -t TAB target_id,est_counts 0Hour_ATCACG_L002_002.extract.sam-express/results.xprs
csvtool namedcol -t TAB target_id,est_counts 6Hour_CGATGT_L002_001.extract.sam-express/results.xprs
csvtool namedcol -t TAB target_id,est_counts 6Hour_CGATGT_L002_002.extract.sam-express/results.xprs
```

We'll be using edgeR to do the basic differential expression analysis of our counts.

To run edgeR, you need to write a data loading and manipulation script in R. In this case, I've provided one – diff_exp.R. This script will load in two samples with two replicates, execute an MA plot, do an MDS analysis/plot, and provide a spreadsheet with differential expression information in it.

Links:

- False Discovery Rate
- Learn R with Swirl

So, download the script:

```
cd /mnt/work
curl -O http://2015-may-nonmodel.readthedocs.org/en/latest/_static/diff_exp.R
```

Now we run the differential expression script with:

```
Rscript diff_exp.R
```

This will produce three files, nema-edgeR-MA-plot.pdf, nema-edgeR-MDS.pdf, and nema-edgeR.csv. The CSV file can be opened directly in Excel; you can also look at it here. It consists of five columns: gene name, log fold change, P-value, and FDR-adjusted P-value.

You can also view more informative versions of these files generated from a different dataset: chick-edgeR-MA-plot.pdf, and chick-edgeR-MDS.pdf.

Remapping your reads to your assembled transcriptome

First, we'll need to make sure bowtie2 is installed:

```
sudo apt-get install -y bowtie2
```

Now, create a bowtie2 index out of your transcriptome:

```
cd /mnt/work
gunzip -c trinity-nematostella-raw.renamed.fasta.gz > trinity-nematostella-raw.renamed.fasta
bowtie2-build  trinity-nematostella-raw.renamed.fasta transcriptome
```

And then, finally, count the number of reads that map to your transcriptome:

```
zcat 0Hour_ATCACG_L002_R1_001.extract.fastq.gz | \
     head -400000 | \
     bowtie2 -U - -x transcriptome > /dev/null
```

You should get something like:

```
97.18% overall alignment rate
```

CHAPTER 11

Miscellaneous advice

11.1 Sequencing depth and number of samples

Hart et al. (2013) provides a nice description and a set of tools for estimating your needed sequencing depth and number of samples. They provide an Excel based calculator for calculating number of samples. Their numbers are surprisingly large to me ;).

In a proposal for an exploratory effort to discover differentially expressed genes, I would suggest 3-5 biological replicates with 30-50 million reads each. More reads is usually cheaper than more replicates, so 50-100m reads may give you more power to resolve smaller fold changes.

11.2 Downloading your data

If you do your sequencing at the MSU Core Facility, you'll get an e-mail from them when you're samples are ready. The e-mail will give you an FTP site, a username, and a password, as well as a URL. You can use these to download your data. For example, if you get:

```
hostname:       titan.bch.msu.edu
username:       rnaseqmodel
password:       QecheJa6

URI:            ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu
```

you can go to ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu in your Web browser; that is, it lets you combine your username and password to open that link.

In this case, you will see a 'testdata' directory. If you click on that, you'll see a bunch of fastq.gz files. These are the files that you want to get onto the HPC.

To download these files onto the HPC, log into the HPC, go to the directory on the HPC you want to put the files in, and run a 'wget' – for example, on the HPC:

```
mkdir ~/testdata
cd ~/testdata

wget -r -np -nH ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu/testdata/
```

This will download _all_ of the files in that directory. You can also do them one at a time, e.g. to get 'Ath_Mut_1_R1.fastq.gz', you would do

> wget ftp://rnaseqmodel:QecheJa6@titan.bch.msu.edu/testdata/Ath_Mut_1_R1.fastq.gz

Tada!

11.3 Developing your own pipeline

Even if all you plan to do is change the filenames you're operating on, you'll need to develop your own analysis pipeline. Here are some tips.

1. Start with someone else's approach; don't design your own. There are lots of partly done examples that you can find on the Web, including in this tutorial.
2. Generate a data subset (the first few 100k reads, for example).
2. Run commands interactively on an HPC dev node until you get all of the commands basically working; track all of your commands in a Word document or some such.
3. Once you have a set of commands that seems to work on small data, write a script. Run the script on the small data again; make sure that works.
4. Turn it into a qsub script (making sure you're in the right directory, have the modules loaded, etc.)
5. Make sure the qsub script works on your same small data.
6. Scale up to a big test data set.
7. Once that's all working, SAVE THE SCRIPT SOMEWHERE. Then, edit it to work on all your data sets (you may want to make subsets again, as much as possible).
8. Provide your scripts and raw counts files as part of any publication or thesis, perhaps via figshare.

Next: More resources

CHAPTER 12

More resources

12.1 Informational resources

UT (Austin) Sequencing Core prices - costs and yields for sequencing.

ANGUS - summer NGS course - lots of resources and materials and book reference

Data Carpentry - intro to R, etc.

Software Carpentry - more scripting, Python, etc.

12.2 Places to share data, scripts, and results files

Figshare.

CHAPTER 13

Miscellaneous questions

1. When should I use de novo assembly, and when should I use reference-guided (ab initio) assembly?

   This is always a judgement call, and you can always try both (although there aren't good methods for comparing the results).

   The short version is that if you have no nearby genomic sequence, you *must* use de novo assembly; if you have an incomplete genomic sequence you *may* want to use de novo assembly; and if you have a great genomic sequence, you *shouldn't* use de novo assembly.

   The positives of using de novo assembly are that you do not depend in any way on the reference. So, if the reference genome is missing, incomplete, or incorrect, you will not have biased results from doing it.

   The negatives are that you will get many more isoforms from de novo transcriptome assembly than you will from reference-based transcriptome assembly, and the process is probably a bit more computationally intensive (and certainly more subject to problems from bad data).

2. What are "transcript families"?

   Transcript families and components are computational terms for "transcripts that may share exons". The biological analogy to use is splice isoforms - but keep in mind that the computer can't necessarily tell the difference between transcripts that are "real" splice variants, noisy splicing, different allelic variants of transcripts, recent paralogs, etc. etc. - all the computer knows is that the transcripts share some amount of sequence.

   So, transcript families are Trinity's best guess at transcripts that come from the same locus.

3. What should we look at in FastQC results for RNAseq data?

   The main thing to pay attention to is the first graph, of quality scores vs position. If your average quality takes a big dip at a particular position, you might consider trimming at that position.

4. How do we transfer our data to Amazon (or any remote computer)?

   There are two options –

   If your data is on your local computer, you can use Cyberduck to transfer the data to Amazon. (see Tips and Tricks for working with Remote Computers).

   If the data is on a remote computer (like your sequencing center) you can probably use 'curl' or 'wget' to copy the data directly from the sequencing center to your Amazon computer. You should ask them what the full URL (with username and password) is to each of your data sets, or find your local computer expert to help out.

5. How do we use Amazon to run full analyses?

   See Tips and Tricks for working with Remote Computers, "Running full analyses".

6. Can we use XSEDE or iPlant or <insert other platform here> to run these analyses?

Yes, but you should omit all of the ‘apt-get’ and ‘pip install’ instructions - the sysadmins on those computers will need to install these programs for you.

7. How do we know if our reference transcriptome is “good enough”?

   See Remapping your reads to your assembled transcriptome.

8. How do I choose the set of tools to use?

   Our recommendations, in order:

   (a) Find a tool that a nearby lab is using, and start there.

   (b) Look at tools and workflows that are used in published papers by groups working in your area.

   (c) Look for good tutorials online.

CHAPTER 14

Tips and Tricks for working with Remote Computers

14.1 Use screen to run things that take a long time.

Often you want to run things that will take days or weeks to run. The ‘screen’ command will let you run programs and record the output, and then come back later and “reconnect”.

For example, try running the beginning bit of digital normalization (Running digital normalization) inside of screen:

```
screen
cd /mnt/work
normalize-by-median.py -k 20 -p -C 20 -N 4 -x 2e9 -s normC20k20.ct *.pe.qc.fq.gz
```

The normalize-by-median command will take a while, but now that it’s running in screen, you can “detach” from your remote computer and walk away for a bit. For example,

- close your terminal window;
- open up a new one and connect into your Amazon machine;
- type ‘screen -r’ to reconnect into your running screen.

(See `amazon/using-screen` for a more complete rundown on instructions.)

14.2 Use CyberDuck to transfer files

To transfer remote files to your local laptop, or local laptop files to the remote system, try using CyberDuck. We’ll walk through it in class.

14.3 Subsetting data

If you want to generate a small subset of a FASTQ file for testing, you can do something like this:

```
gunzip -c /mnt/data/SRR534005_1.fastq.gz | head -400000 | gzip > sample.fq.gz
```

This will take 400,000 lines (or 100,000 FASTQ records) from the beginning of the `SRR534005_1.fastq.gz` file and put them in the `sample.fq.gz` file.

14.4 Running full analyses on Amazon Web Services

You need to do three things to run a full analysis on AWS (or really any cloud machine) –

1. you need to get your data onto that machine.
2. you need to be prepared to let things run for a long time.
3. you need to have a large disk to store all the intermediate files. A good rule of thumb is that every 200 million reads requires about a TB of intermediate disk space.

Getting your data onto the machine can be done by using the 'curl' command to download data from (e.g.) your sequencing core. This will be core specific and it's something we can help you with when you need the help.

To let things run for a long time, you basically need to run them in screen (see above, "Use screen.")

By default, Amazon doesn't give you really big hard disks on your machine – you can use 'df' to take a look. On an m3.xlarge machine, you can ask about disk space on /mnt by using 'df' (disk free):

```
df -k /mnt
```

You should see something like this:

```
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/xvdb       38565344 20098736  16500940  55% /mnt
```

which tells you that /mnt has 40 GB of disk space.

To add disk space to your Amazon instance, see this set of instructions:

http://angus.readthedocs.org/en/2014/amazon/setting-up-an-ebs-volume.html

The simplest advice is to make /mnt a 1 TB disk, which should hold a half dozen mRNAseq data sets and all the intermediate data.

CHAPTER 15

Technical information

The github repository for this workshop is publicly available at https://github.com/ngs-docs/2015-oct-nonmodel.