Teaching Big Data Analtyics to Business School MS Students

Ramesh Shankar

Uconn School of Business

IT Teaching Workshop 2019, Wharton

MSBAPM Curriculum

Business Analytics

Required courses (5):

- Business Process Modeling and Data Management (OPIM 5272)
- Statistics in Business Analytics (OPIM 5603)
- Predictive Modeling (OPIM 5604)
- Business Decision Modeling (OPIM 5641)
- Data Mining and Business Intelligence (OPIM 5671)

Project Management

Required courses (4):

- Introduction to Project Management (OPIM 5270)
- Project Leadership and Communications (MGMT 5620)
- Project Risk and Cost Management (OPIM 5668)
- Advanced Business Analytics and Project Management (OPIM 5770)

Electives

- Visual Analytics (OPIM 5501)
- Big Data Analytics with Hadoop (OPIM5502)
- Adaptive Business Intelligence (OPIM 5504)
- Analytical Consulting for Financial Services (OPIM 5505)
- Agile Project Management (OPIM 5507)
- Healthcare Analytics and Research Methods (OPIM 5508)
- Introduction to Deep Learning (OPIM 5509)
- Web Analytics (OPIM 5510)
- Survival Analysis using SAS BASE (OPIM 5511)
- Data Science using Python (OPIM 5512)

MSBAPM Capstone Project

Quick Facts

Locations	Hartford or Stamford, Connecticut
Semesters	Fall, Spring, or Summer
Format	Full- or Part-time
Credits	37-credits (Curriculum)

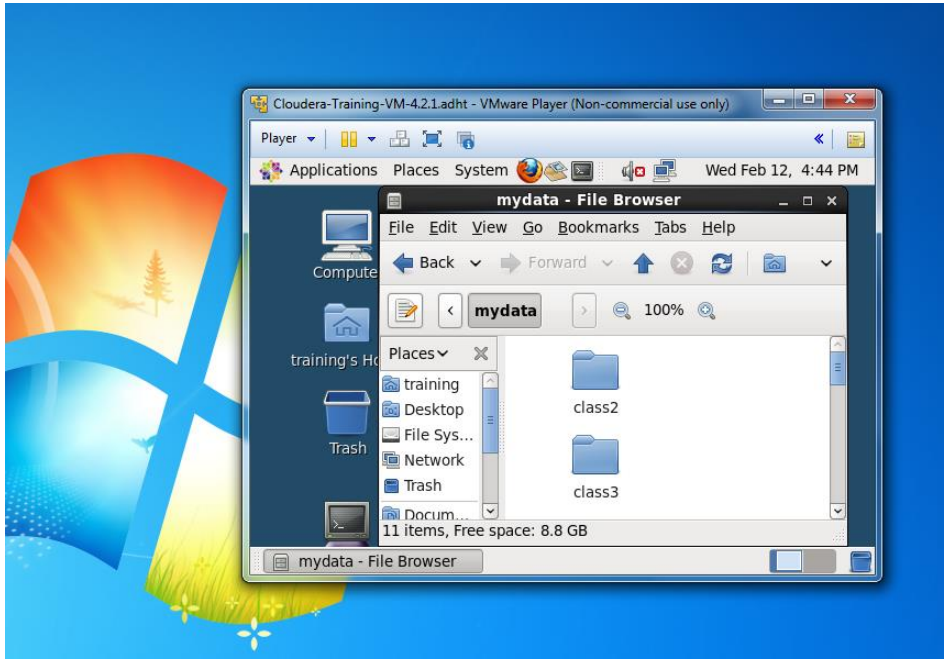
Hadoop books

Source: David Tilson, IT Teaching Workshop 2018

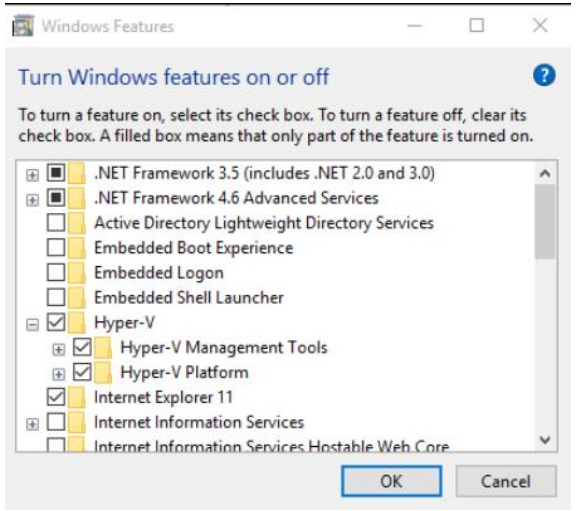
Hadoop resources

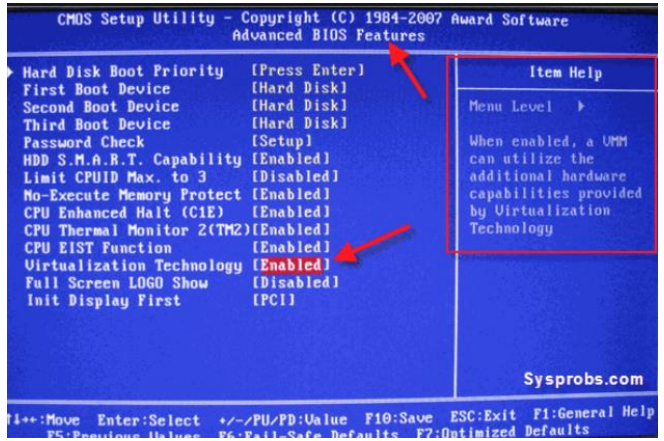
HORTONWORKS

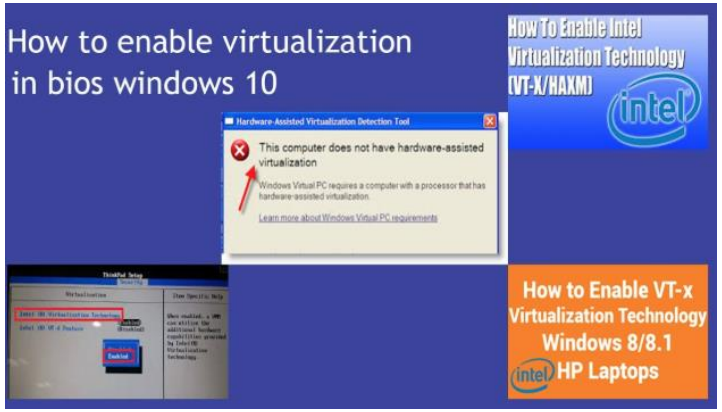
Cloudera VM



Enabling virtualization







AWS EMR (Elastic MapReduce) Cluster

Using a cluster is not for the faint-hearted

Estimated cost

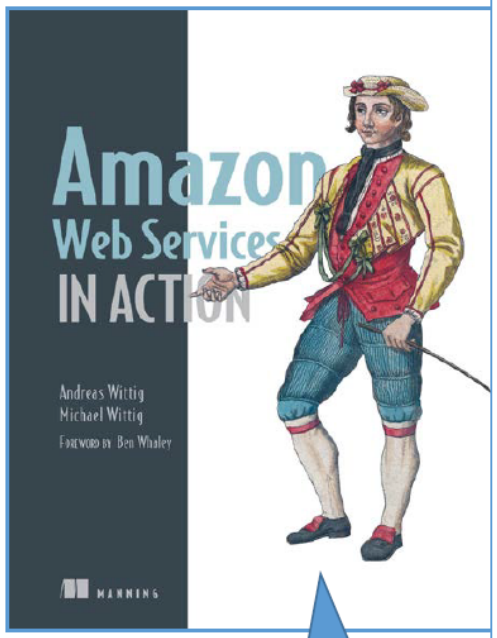
- $6k cluster time (Spark was most expensive part)
- $2k admin time
- $10k consulting time (one-off)

Asking AWS for $5k credit (~50 students) and they are considering more turn-key solution (no promises yest)

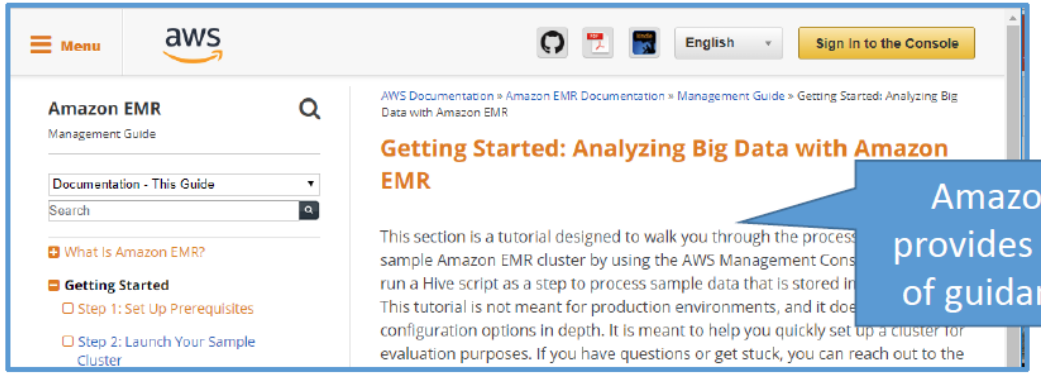
>50% cost reduction by active management (off at night, reset). Could be less than $50/student at 50 students for 10 weeks

AWS provide good support (solution architects)

- But they were learning too
- Multi-user different and complex across AWS, Linux, HDFS, Pig, Hive, S3, Hue, Spark, and Zeppelin
- Will use us as case study in multi-tenancy cluster operation



This book was very helpful in learning AWS terminology

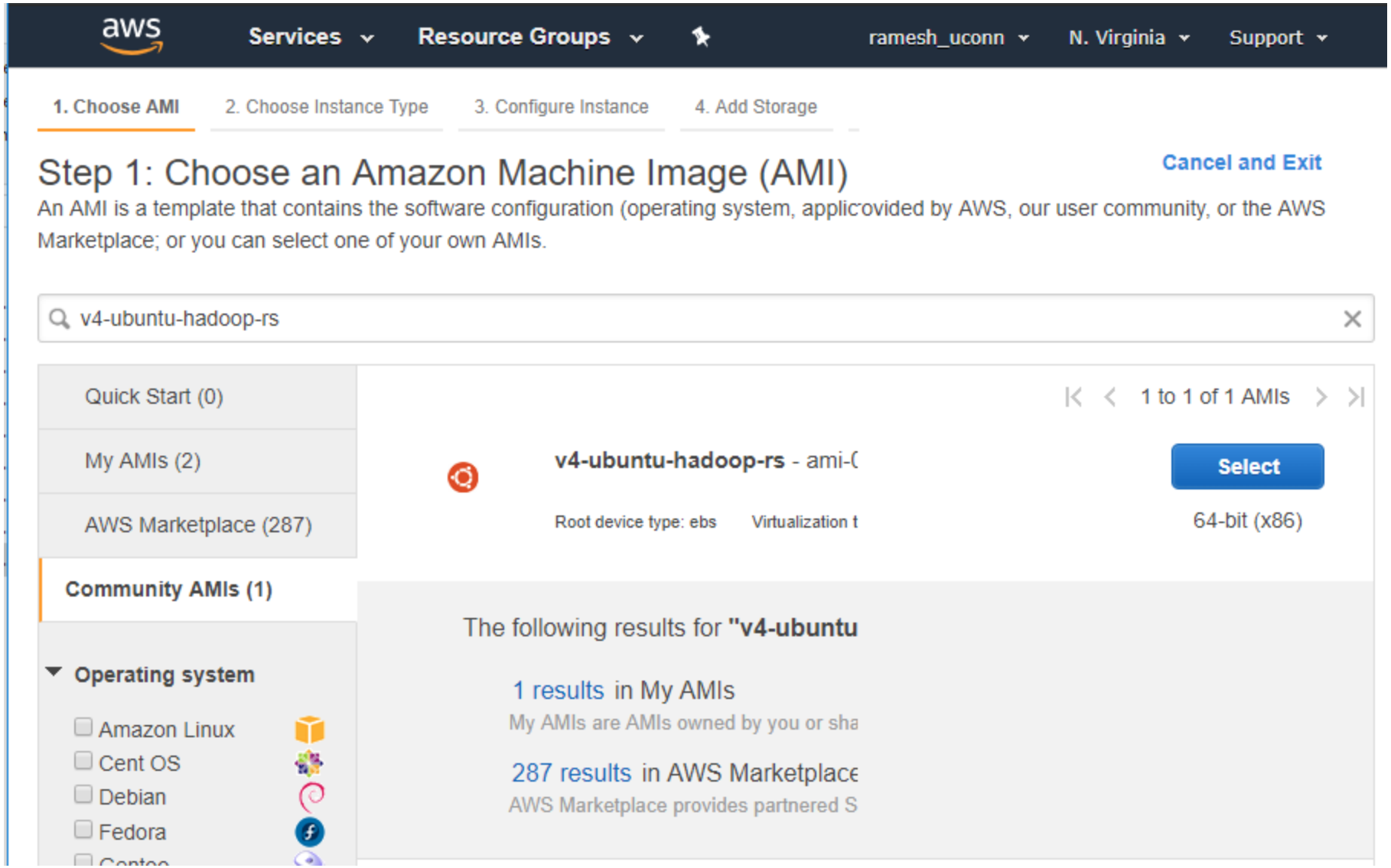


Amazon provides lots of guidance

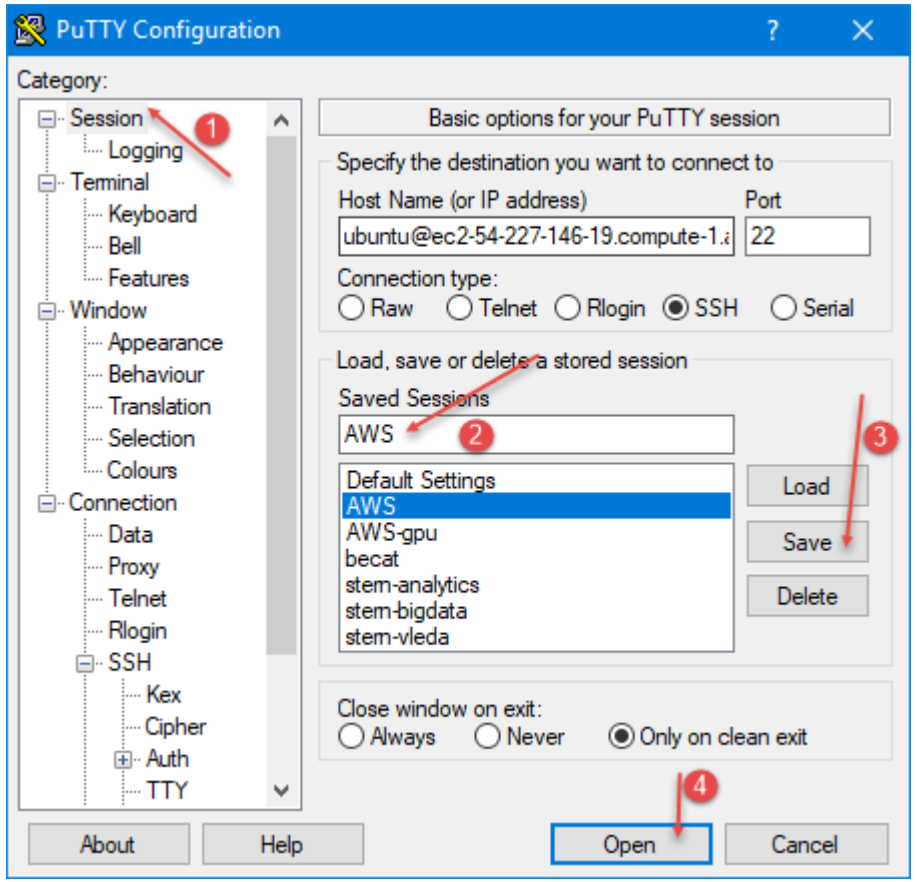
Guidance for getting started here: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html

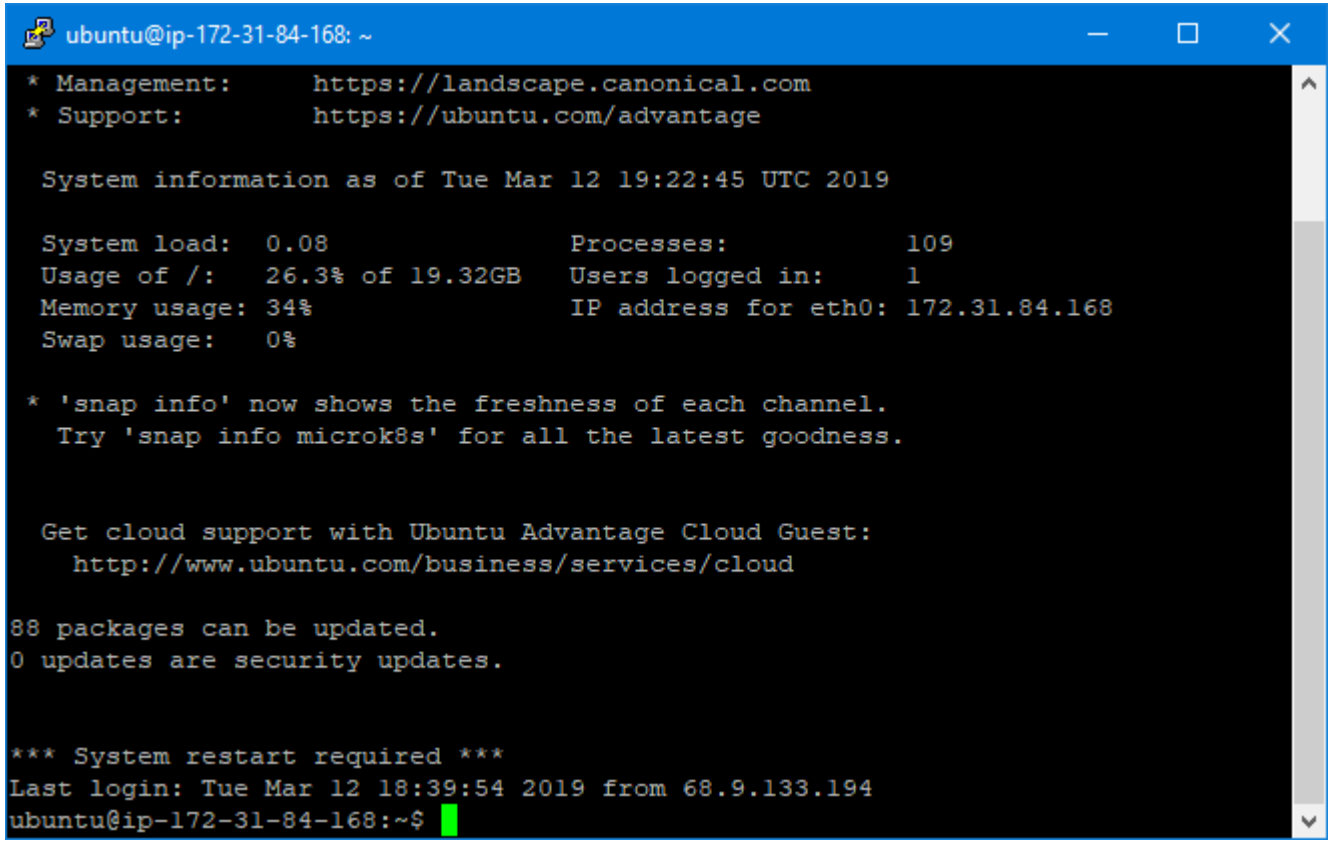
9

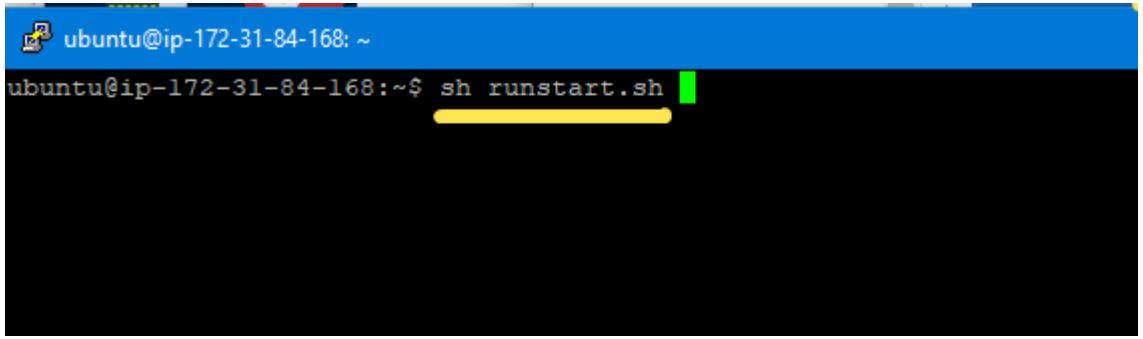
Source: David Tilson, IT Teaching Workshop 2018


aws
Services
Resource Groups
ramesh_uconn
N. Virginia
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
An AMI is a template that contains the software configuration (operating system, applicovided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
v4-ubuntu-hadoop-rs
Quick Start (0)
My AMIs (2)
AWS Marketplace (287)
Community AMIs (1)
Operating system
Amazon Linux
Cent OS
Debian
Fedora
1 to 1 of 1 AMIs
v4-ubuntu-hadoop-rs - ami-(
Root device type: ebs
Virtualization t
Select
64-bit (x86)
The following results for "v4-ubuntu
1 results in My AMIs
My AMIs are AMIs owned by you or sha
287 results in AWS Marketplace
AWS Marketplace provides partnered S

AWS EC2:


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Cipher
Auth
TTY
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
ubuntu@ec2-54-227-146-19.compute-1.
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
AWS
Default Settings
AWS
AWS-gpu
becat
stem-analytics
stem-bigdata
stem-vleda
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel
1
2
3
4


ubuntu@ip-172-31-84-168: ~
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
System information as of Tue Mar 12 19:22:45 UTC 2019
System load: 0.08
Usage of /: 26.3% of 19.32GB
Memory usage: 34%
Swap usage: 0%
Processes: 109
Users logged in: 1
IP address for eth0: 172.31.84.168
* 'snap info' now shows the freshness of each channel.
Try 'snap info microk8s' for all the latest goodness.
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
88 packages can be updated.
0 updates are security updates.
*** System restart required ***
Last login: Tue Mar 12 18:39:54 2019 from 68.9.133.194
ubuntu@ip-172-31-84-168:~$


ubuntu@ip-172-31-84-168: ~
ubuntu@ip-172-31-84-168:~$ sh runstart.sh

Topics covered

- Linux
- Hadoop Distributed File System
- Apache Sqoop
  - Extract data from RDBMS, into HDFS
- Apache Pig
  - Extract, Transform, Load (ETL) on data obtained via Sqoop
  - Schema on read, no permanent schema, flat files
- Apache Hive
  - Hadoop Data Warehousing Tool
  - Schema on read, permanent schema required, flat files
- MapReduce – conceptual overview
- Spark
  - In-memory Analytics
- Recommender Systems
  - Illustrates Spark

HDFS

```
[training@localhost mydata]$ hadoop fs -mkdir /test
[training@localhost mydata]$ hadoop fs -mkdir /test/test1
[training@localhost mydata]$ hadoop fs -put products.txt /test/test1/
[training@localhost mydata]$ hadoop fs -ls /test/test1
Found 1 items
-rw-r--r--   1 training supergroup         63 2017-10-17 12:25 /test/test1/products.txt
[training@localhost mydata]$ hadoop fs -rm /test/test1
rm: `/test/test1': Is a directory
[training@localhost mydata]$ hadoop fs -rmdir /test/test1
rmdir: `/test/test1': Directory is not empty
[training@localhost mydata]$ hadoop fs -rm -r /test/test1
Deleted /test/test1
[training@localhost mydata]$ hadoop fs -ls /test
[training@localhost mydata]$ hadoop fs -rmdir /test
```

```
~ $ hadoop fs -ls /
Found 6 items
drwxr-xr-x   - ubuntu supergroup  2019-03-11 20:28 /linkage
drwxr-xr-x   - ubuntu supergroup  2019-05-31 19:25 /mydata
drwxr-xr-x   - ubuntu supergroup  2019-03-11 20:36 /spark
drwx-w----   - ubuntu supergroup  2019-05-31 19:16 /tmp
drwxr-xr-x   - ubuntu supergroup  2019-05-31 19:16 /user
drwxr-xr-x   - ubuntu supergroup  2019-05-31 18:39 /userdata
```

Sqoop

```
$ sqoop import \
--connect jdbc:mysql://localhost/sakila \
--username ubuntu --password training \
--warehouse-dir /userdata \
--table actor
```

```
ubuntu@ip-172-31-89-101: ~
ubuntu@ip-172-31-89-101:~$ hadoop fs -ls /userdata
Found 1 items
drwxr-xr-x   - ubuntu supergroup          0 2019-03-11 21:08 /userdata/rental
ubuntu@ip-172-31-89-101:~$
```

```
~ $ hadoop fs -ls /userdata/rental
Found 9 items
-rw-r--r--   3 ubuntu supergroup          0 2019-05-31 18:08 /userdata/rental/_SUCCESS
-rw-r--r--   3 ubuntu supergroup        393 2019-05-31 18:08 /userdata/rental/part-m-00000
-rw-r--r--   3 ubuntu supergroup        553 2019-05-31 18:08 /userdata/rental/part-m-00001
-rw-r--r--   3 ubuntu supergroup        555 2019-05-31 18:08 /userdata/rental/part-m-00002
-rw-r--r--   3 ubuntu supergroup        685 2019-05-31 18:08 /userdata/rental/part-m-00003
-rw-r--r--   3 ubuntu supergroup        558 2019-05-31 18:09 /userdata/rental/part-m-00004
-rw-r--r--   3 ubuntu supergroup        480 2019-05-31 18:09 /userdata/rental/part-m-00005
-rw-r--r--   3 ubuntu supergroup        643 2019-05-31 18:09 /userdata/rental/part-m-00006
-rw-r--r--   3 ubuntu supergroup        486 2019-05-31 18:09 /userdata/rental/part-m-00007
```

```
~ $ hadoop fs -cat /userdata/rental/part-m-00000 | head -3
972,2005-05-30 20:21:07.0,2,411,2005-06-06 00:36:07.0,1,2006-02-15 21:30:53.0
2117,2005-06-17 20:24:00.0,2,170,2005-06-23 17:45:00.0,2,2006-02-15 21:30:53.0
361,2005-05-27 07:03:28.0,6,587,2005-05-31 08:01:28.0,1,2006-02-15 21:30:53.0
```

Pig

```
listings = LOAD '/mydata/class3/listings.txt' AS
(
listing_id:int,date_listed:chararray,
list_price:float,sq_feet:int,
address:chararray,town:chararray
);
bytown = GROUP listings BY town;
DESCRIBE bytown;
--optional step:
--byproduct = LIMIT byproduct 5;
--Top 2 most expensive homes per town
top_homes = FOREACH bytown {
        sorted = ORDER listings BY
                list_price DESC;
        most_expensive = LIMIT sorted 2;
        GENERATE group, most_expensive;
};
DESCRIBE top_homes;
DUMP top_homes;
```

Load data

Group by town

In each group, sort by list_price

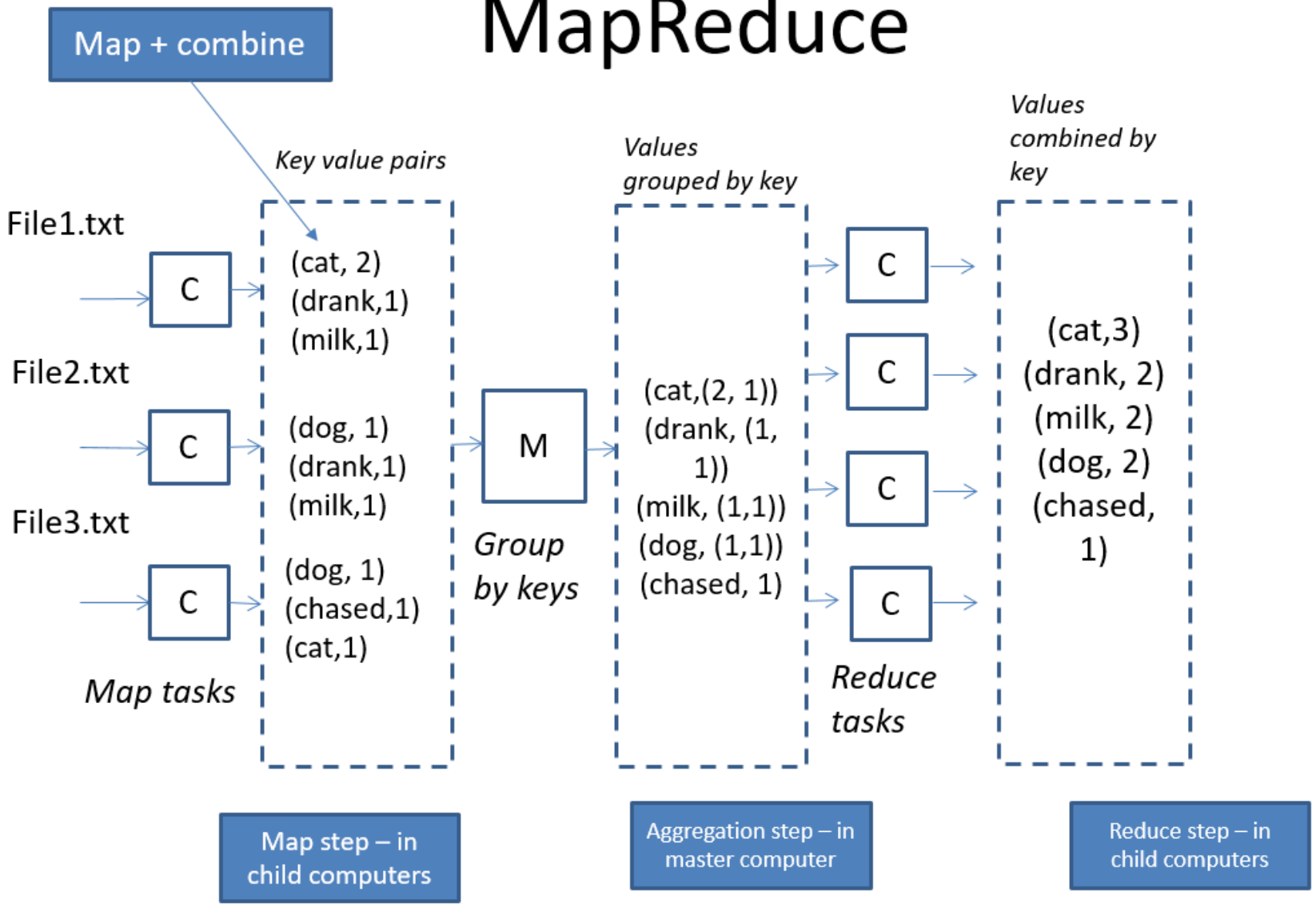
Limit to top 2 most expensive homes

Generate new (group) record in top_homes table

Hive

```
hive> SELECT * FROM homes;
OK
listing_id      list_price      sqft    realtor_id      town
1       146000.0        1750    25      Storrs
2       235000.0        2100    17      Storrs
3       101000.0        1550    53      Hartford
4       376000.0        2900    17      Storrs
5       291000.0        2400    17      Hartford
6       409000.0        3500    25      Stamford
Time taken: 0.129 seconds
hive> SELECT * FROM realtors;
OK
realtor_id      realtor
17      Alec Baldwin
25      Al Pacino
53      Kevin Spacey
Time taken: 0.484 seconds
hive> SELECT r.realtor, h.list_price, h.sqft, h.town
    > FROM realtors r JOIN homes h
    > ON r.realtor_id = h.realtor_id
    > ;
realtor list_price      sqft    town
Alec Baldwin    235000.0        2100    Storrs
Alec Baldwin    376000.0        2900    Storrs
Alec Baldwin    291000.0        2400    Hartford
Al Pacino       146000.0        1750    Storrs
Al Pacino       409000.0        3500    Stamford
Kevin Spacey    101000.0        1550    Hartford
Time taken: 81.428 seconds
```

MapReduce


Map + combine
Key value pairs
Values grouped by key
Values combined by key
File1.txt
File2.txt
File3.txt
C
C
C
(cat, 2)
(drank,1)
(milk,1)
(dog, 1)
(drank,1)
(milk,1)
(dog, 1)
(chased,1)
(cat,1)
Map tasks
M
Group by keys
(cat,(2, 1))
(drank, (1, 1))
(milk, (1,1))
(dog, (1,1))
(chased, 1)
C
C
C
C
Reduce tasks
(cat,3)
(drank, 2)
(milk, 2)
(dog, 2)
(chased, 1)
Map step – in child computers
Aggregation step – in master computer
Reduce step – in child computers

(userid artistid playcount)

user_artist_data.txt

```
1000002 1 55
1000002 1000006 33
1000002 1000007 8
1000002 1000009 144
1000002 1000010 314
1000002 1000013 8
1000002 1000014 42
1000002 1000017 69
1000002 1000024 329
```

(artistid artist_name)

artist_data.txt

```
1   Portishead
100 Phoenix
1000006 Phil Collins Big Band
1000007 The Phil Collins Big Band
1000009 A Perfect Circle
1000010 Aerosmith
1000013 MC Hawking
1000014 Pantera
1000017 Judas Priest
```

(badid goodid)

artist_alias.txt

```
1000434 1000518
1021484 1234336
1014609 1014609
1000287 1239413
1004729 1003612
1006586 1021625
1008128 1019469
1002081 1013150
1939    6785079
```

Spark – recommender system (ALS)

```
artistByID.filter { case (id, name) =>
existingProducts.contains(id)
}.values.collect().foreach(println)
```

Some of the artists this person listens to:

```
Sonny Rollins
Thelonious Monk
Sublime
Weather Report
Bob Dylan
Pink Floyd
Nine Inch Nails
Otis Redding
Stevie Wonder
```

Make 5 recommendations to this user:

```
scala> val recommendations = model.recommendProducts(1000002,5)
scala> recommendations.foreach(println)
```

User ID, Artist ID, estimated rating

```
Rating(1000002,1003433,1.1815765349909169)
Rating(1000002,719,1.1266192051236328)
Rating(1000002,1001172,1.1254197498630407)
Rating(1000002,1000840,1.097728376342873)
Rating(1000002,1034635,1.0651848760166387)
```

```
scala> val recommendedProductIDs = recommendations.map(_.product).toSet

scala> artistByID.filter { case (id, name) =>
     | recommendedProductIDs.contains(id)
     | }.values.collect().foreach(println)


Lee Ritenour
60ft Dolls
Belly
Ella Fitzgerald
```