L07: SQL, SparkSQL, Hive, and more

ANLY 502: Massive Data Fundamentals

Simson Garfinkel

March 13, 2017

Outline for today's class — Databases!

What's next — Final projects

Tech:

- Regular Expressions
- New Instance Types
- Accessing Jupyter Notebook on Amazon (revised)

Databases

- Database data models: records, tables, relational, object, document
- Database access models: Embedded, Client/Server, Distributed
- Database persistence/durability models: ephemeral, durable, streaming
- Database options at Amazon

SQL

- Flavors of SQL
- Database Normalization
- Worked example: tracking files with SQLite.

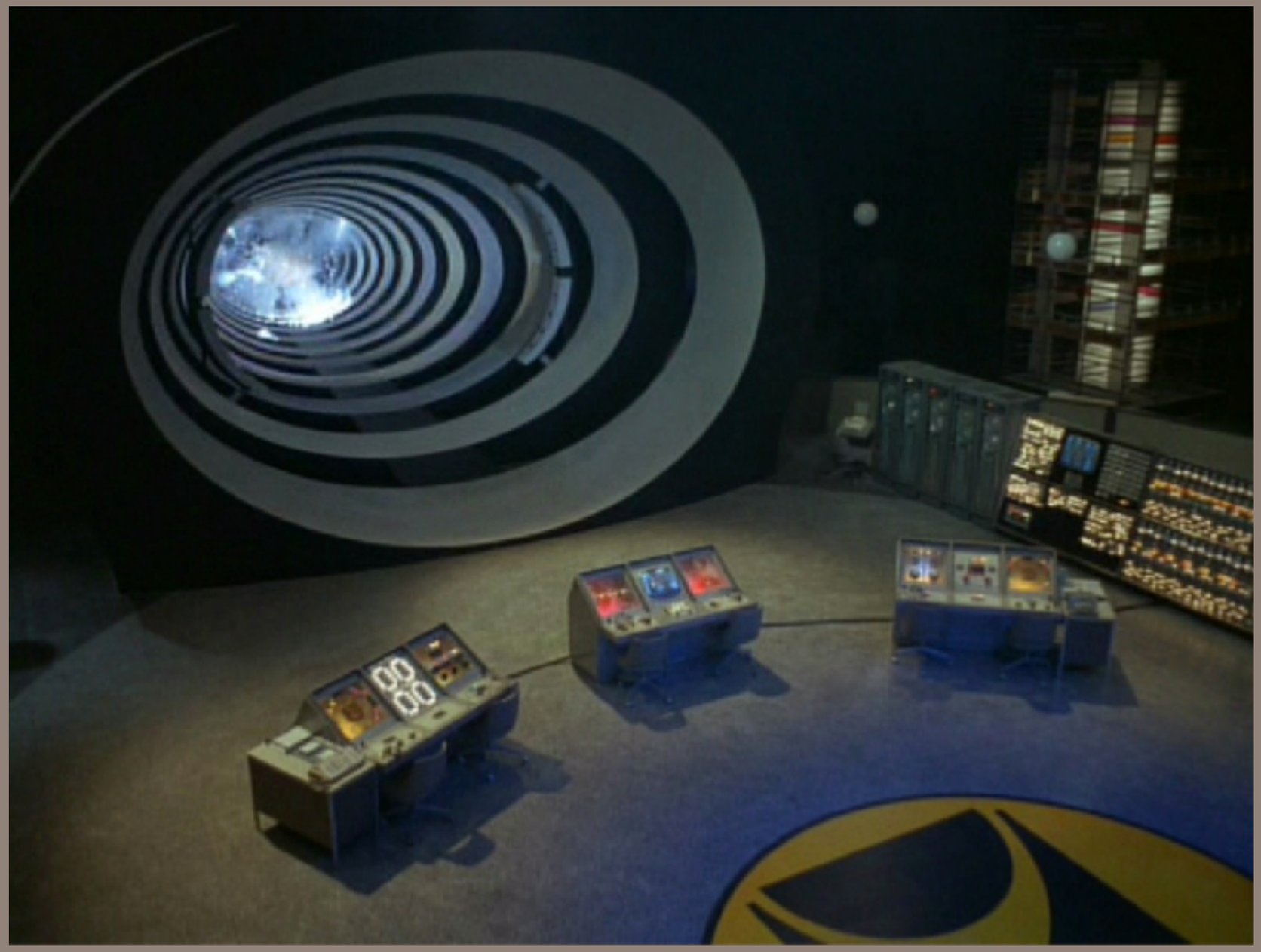
Lab:

- Accessing Forensicswiki logs in SQL server

SparkSQL

- Building a database from text files
- Building a database from MySQL

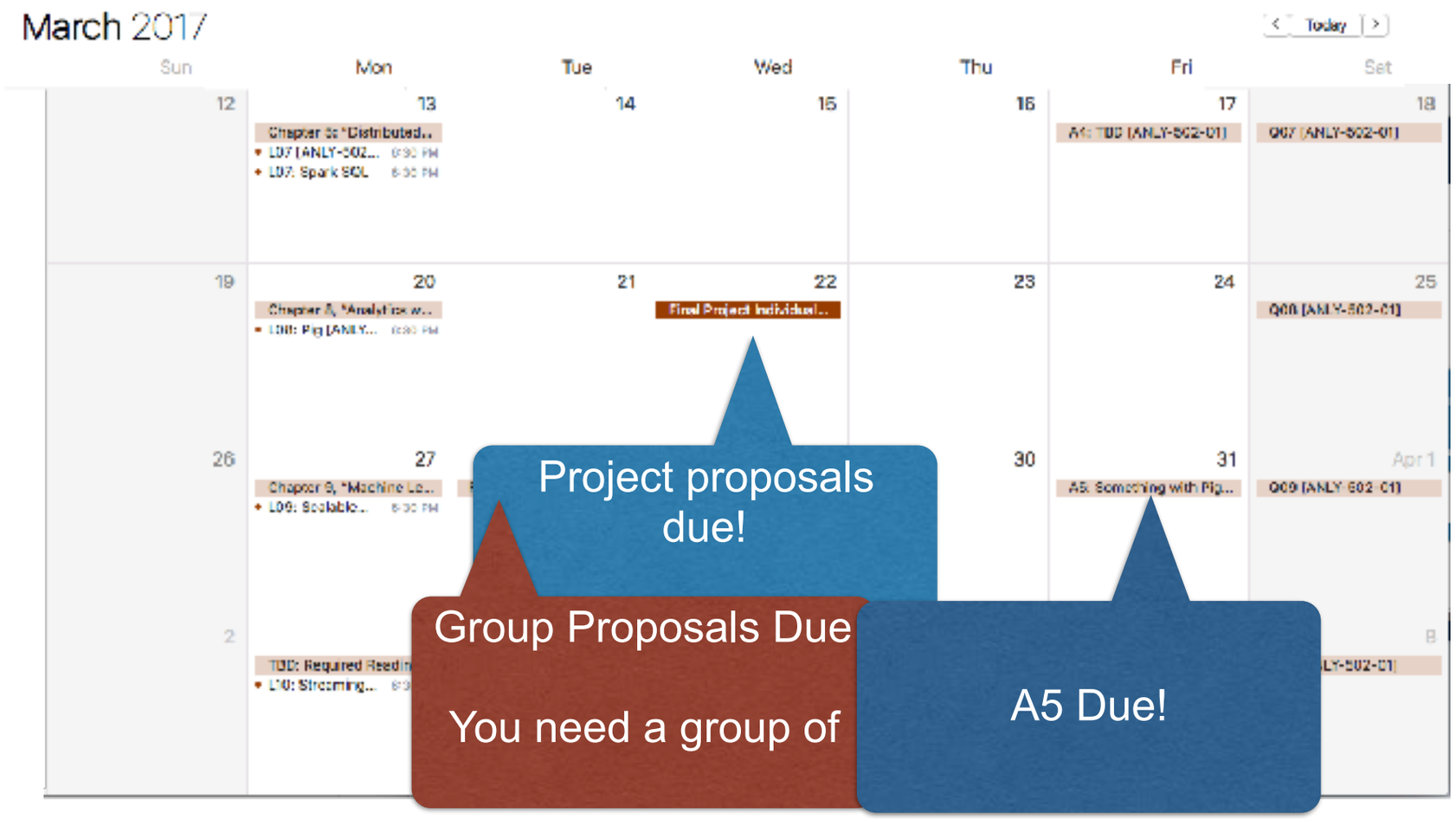
Hive



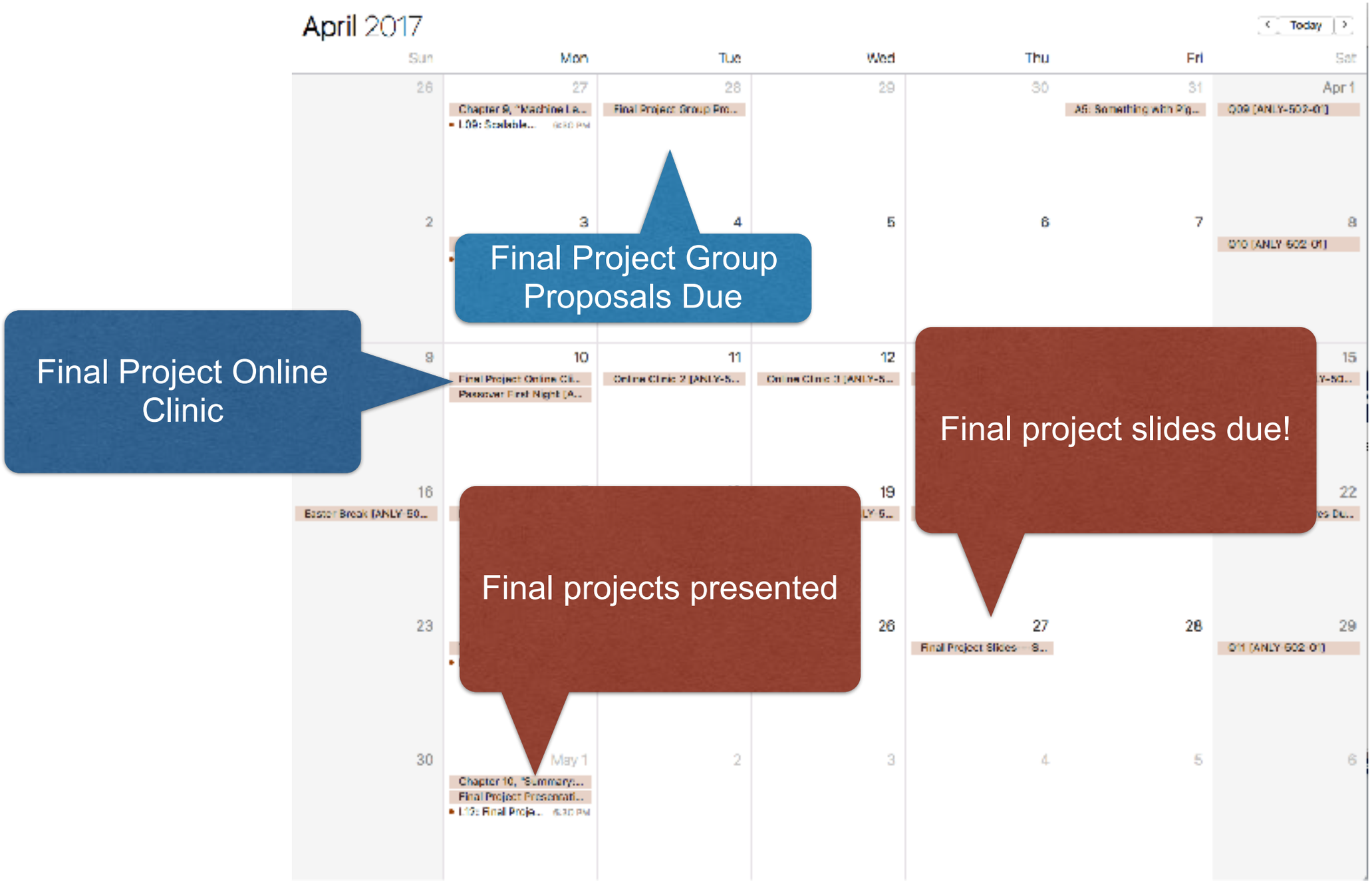
"Time Tunnel", 1966, ABC Television

Part 1: The Future & Databases

March 2017


March 2017
Today
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Project proposals due!
Group Proposals Due
You need a group of
A5 Due!

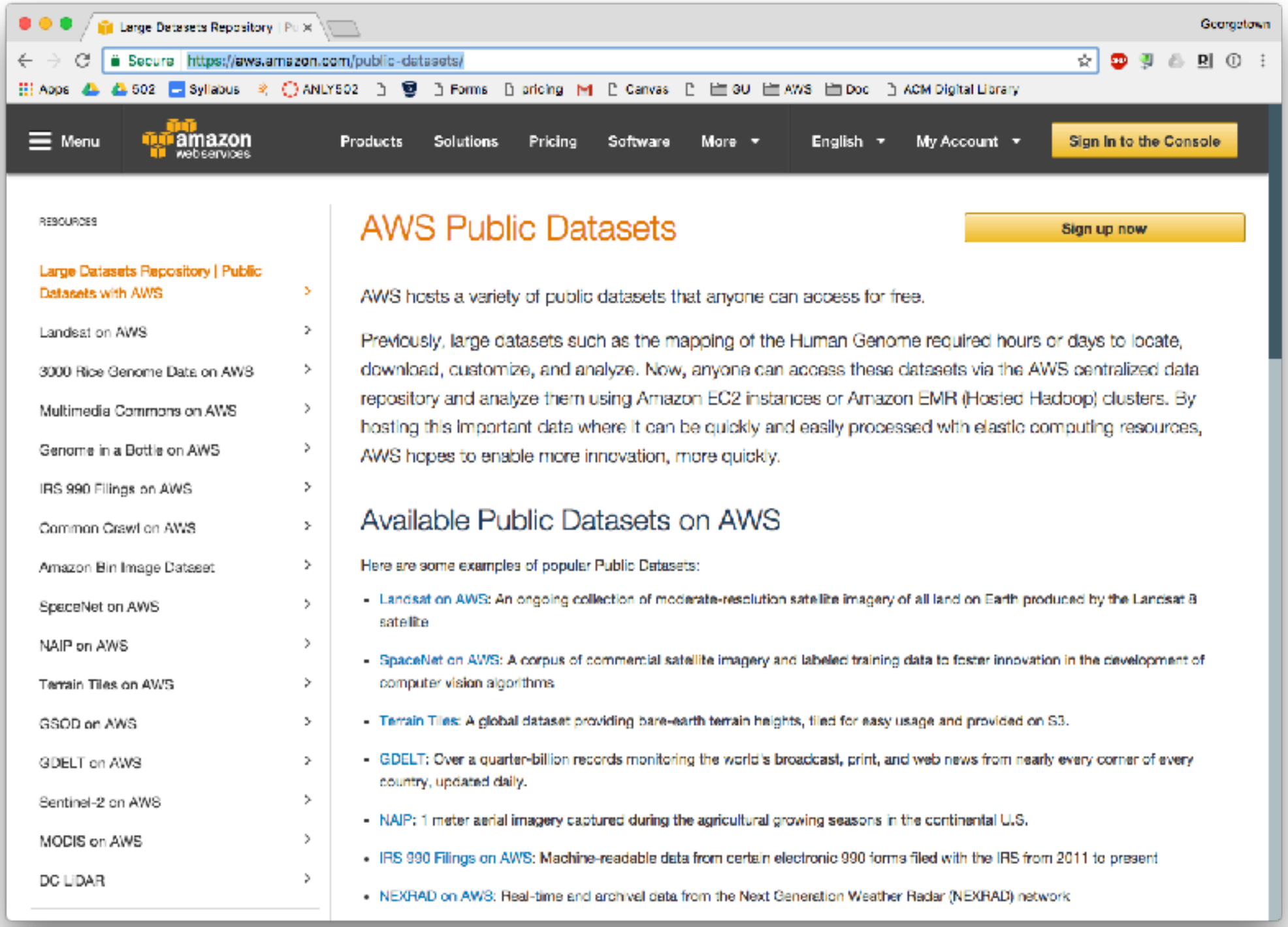
April 2017


April 2017
Today
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Final Project Group Proposals Due
Final Project Online Clinic
Final project slides due!
Final projects presented

Here are some final project ideas!

Browse the AWS Public Datasets:

—*https://aws.amazon.com/public-datasets/*



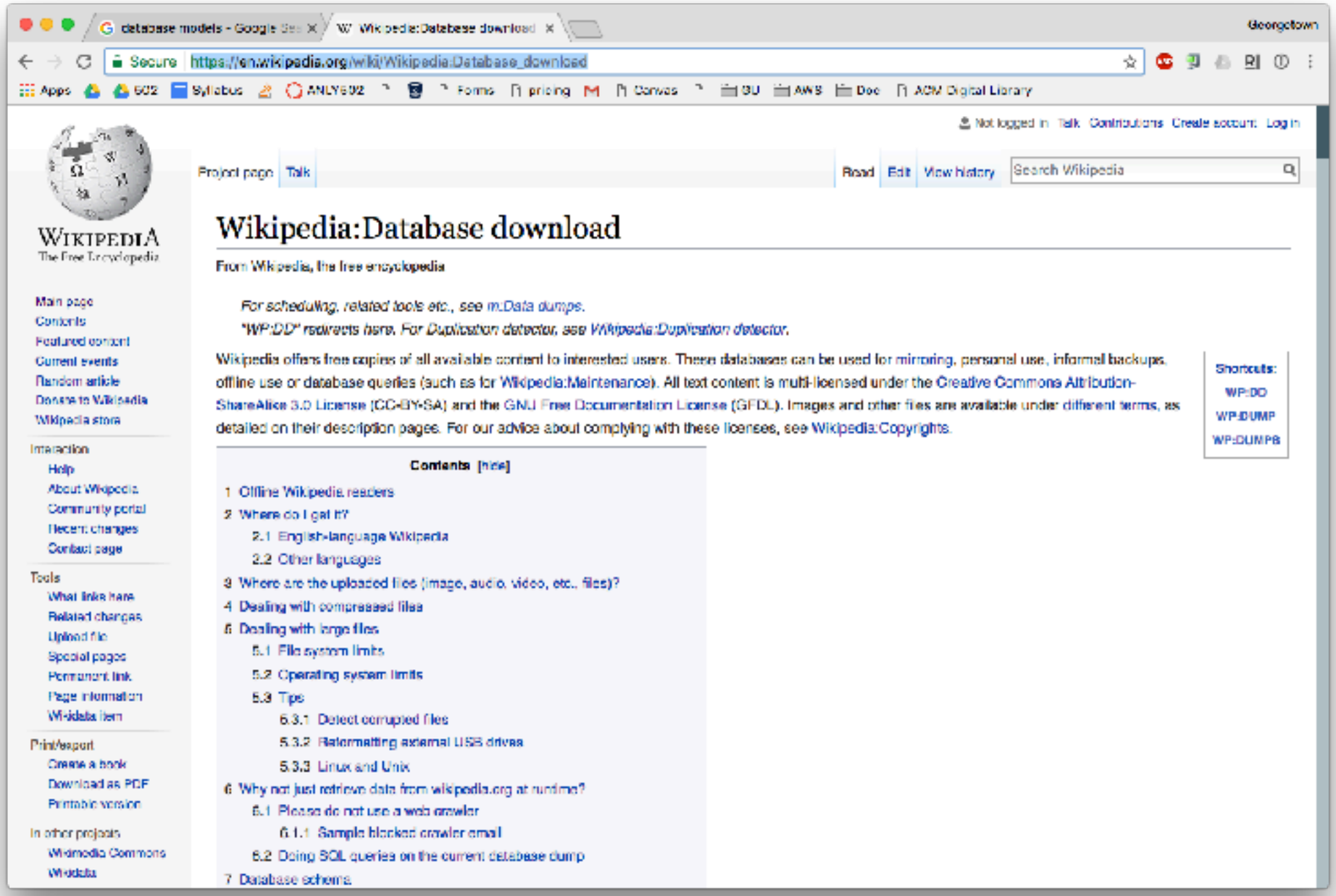
Perform a deep analysis of Wikipedia!

Articles

Authors

Art / Multimedia

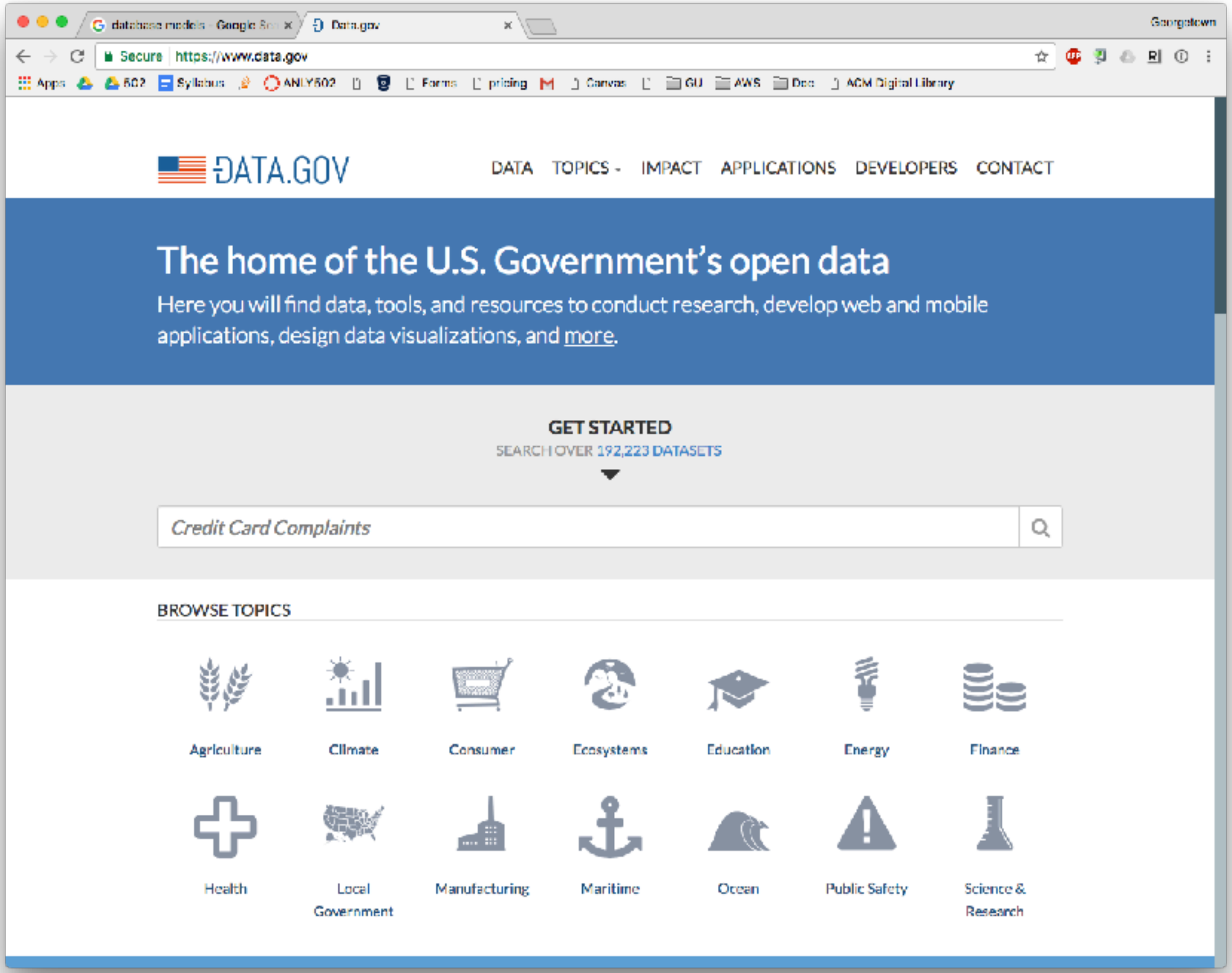
Edit wars



Need ideas for analyzing wikipedia? Look at ACM Digital Library

—*http://dl.acm.org/results.cfm?query=wikipedia*

data.gov —


Data.gov
Secure https://www.data.gov
DATA.GOV
DATA TOPICS IMPACT APPLICATIONS DEVELOPERS CONTACT
The home of the U.S. Government's open data
Here you will find data, tools, and resources to conduct research, develop web and mobile applications, design data visualizations, and more.
GET STARTED
SEARCH OVER 192,223 DATASETS
Credit Card Complaints
BROWSE TOPICS
Agriculture
Climate
Consumer
Ecosystems
Education
Energy
Finance
Health
Local Government
Manufacturing
Maritime
Ocean
Public Safety
Science & Research

Other references:

https://www.forbes.com/sites/bernardmarr/2016/02/12/big-data-35-brilliant-and-free-data-sources-for-2016/

http://www.jenunderwood.com/2016/01/14/my-favorite-public-data-sources/

https://www.analyticsvidhya.com/blog/2016/11/25-websites-to-find-datasets-for-data-science-projects/

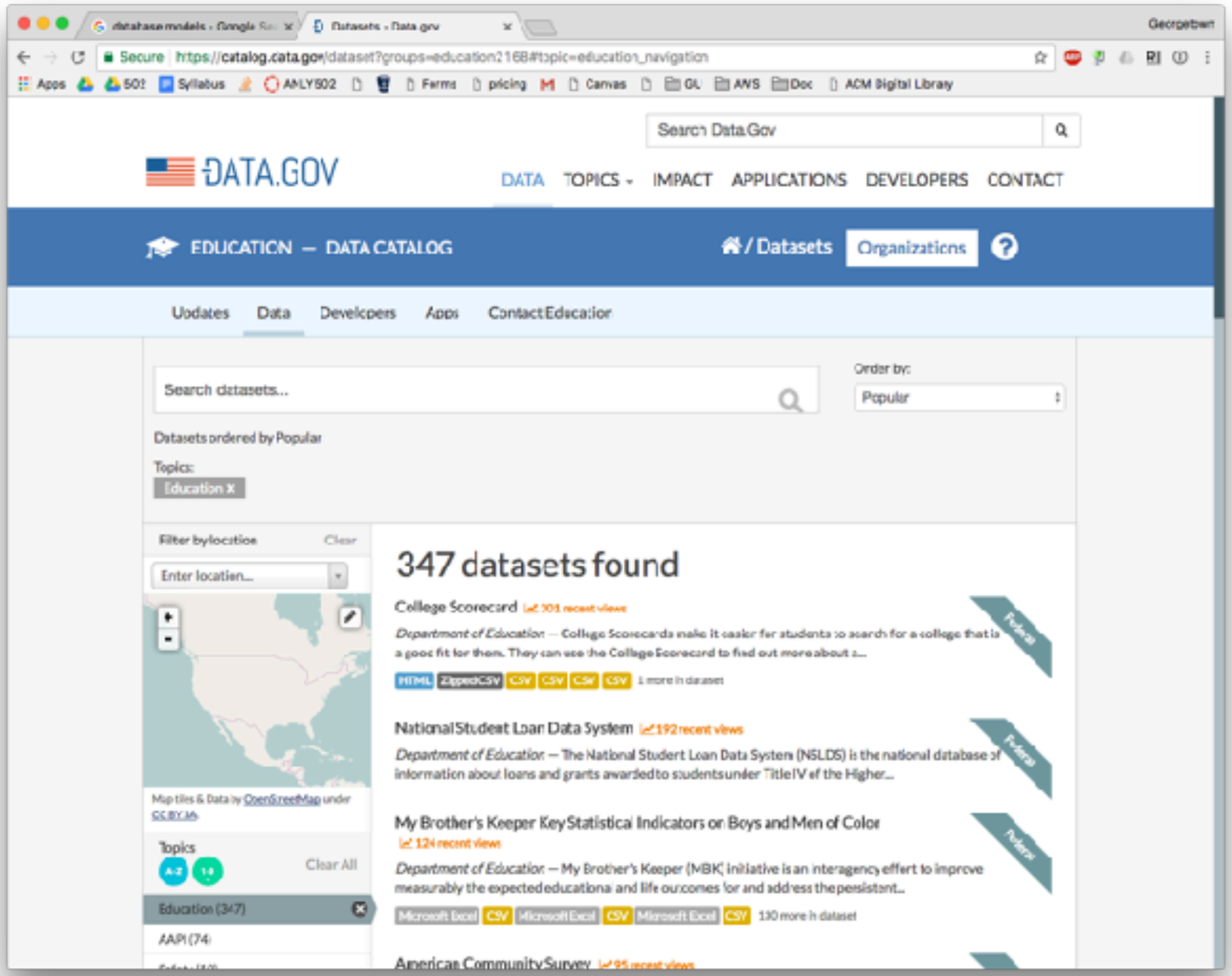
https://www.dataquest.io/blog/free-datasets-for-projects/

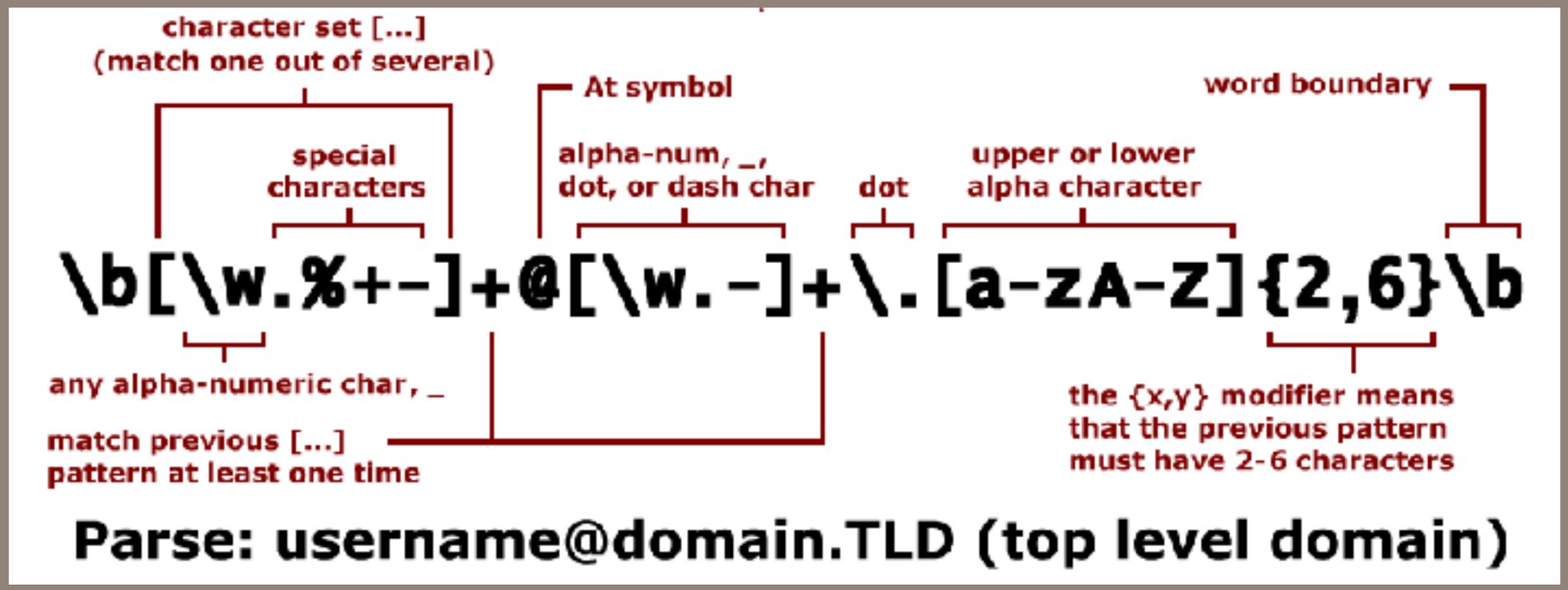
http://blog.thedataincubator.com/2014/10/data-sources-for-cool-data-science-projects-part-1/

data.gov datasets are not massive… but there are a massive number of them!

You could analyze:

- What kinds of datasets are available?
- What is the data quality?
- Is there inadvertent personally identifiable information (PII)?





Regular Expressions

Using and not using regular expressions

Regular expressions are better than hacking strings.

- Hacking strings:

```
s = "234  456"
(a,b) = s.split("\t")
```

- Regular expressions:

```
s = "234  456"
r = re.search("(\d+)\s+(\d+)", s)
if r:
    (a,b) = r.group(1,2)
```

So why use regular expressions?

- " " might not be a tab!
  - *Might be spaces, etc.*
- Definition of a *space* depends on *localization.*
  - "*If UNICODE is set, this will match the characters [\t\n\r\f\v] plus whatever is classified as space in the Unicode character properties database.*"
- Better error handling

When not to use regular expressions

Parsing HTML

- Use BeautifulSoup or a similar parser.

Parsing programming languages.

- Use a parser for the language.

On public websites that allow user-specified searches.

- Regular expressions can be used for denial-of-service attacks.

—*https://en.wikipedia.org/wiki/ReDoS*

Regular expressions are text strings that describe search patterns

Python and Java both use regular expressions. They are *mostly* the same languages.

Common to both Java and Python:

String	Matches
x	*Match character* x *(unless x is special)*
.	*any character*
[abc]	a, b *or* c
[a-z]	*characters* a-z
x?	*nothing* or x
x*	*nothing or any number of* x
x+	*one or more* x
x|y	x *or* y
^x	x *at the beginning of a line*
x$	x *at the end of a line*

Python-specific additions

https://docs.python.org/3.4/library/re.html

number — Matches the existing group

`"(.+) \1"` matches `"the the"` or `"55 55"` but not `"thethe"`

\d — Any decimal digit

\D — Any non-decimal digit

\s — Any whitespace

\S — Any non-whitespace

\w — Any "word" character

\W — Any non-word character [^a-zA-Z0-9_]

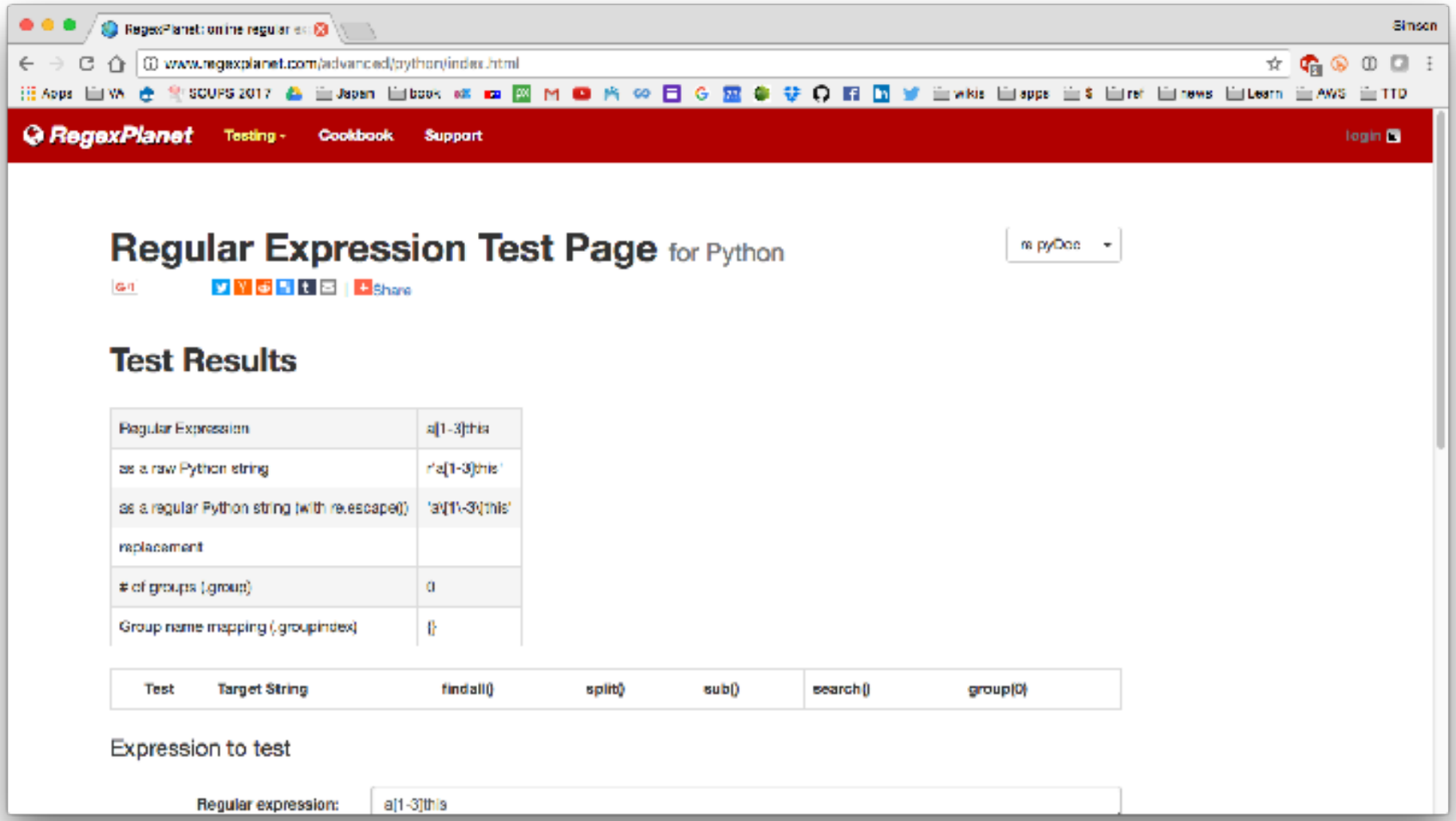
\000-\177 — Octal character 000-177

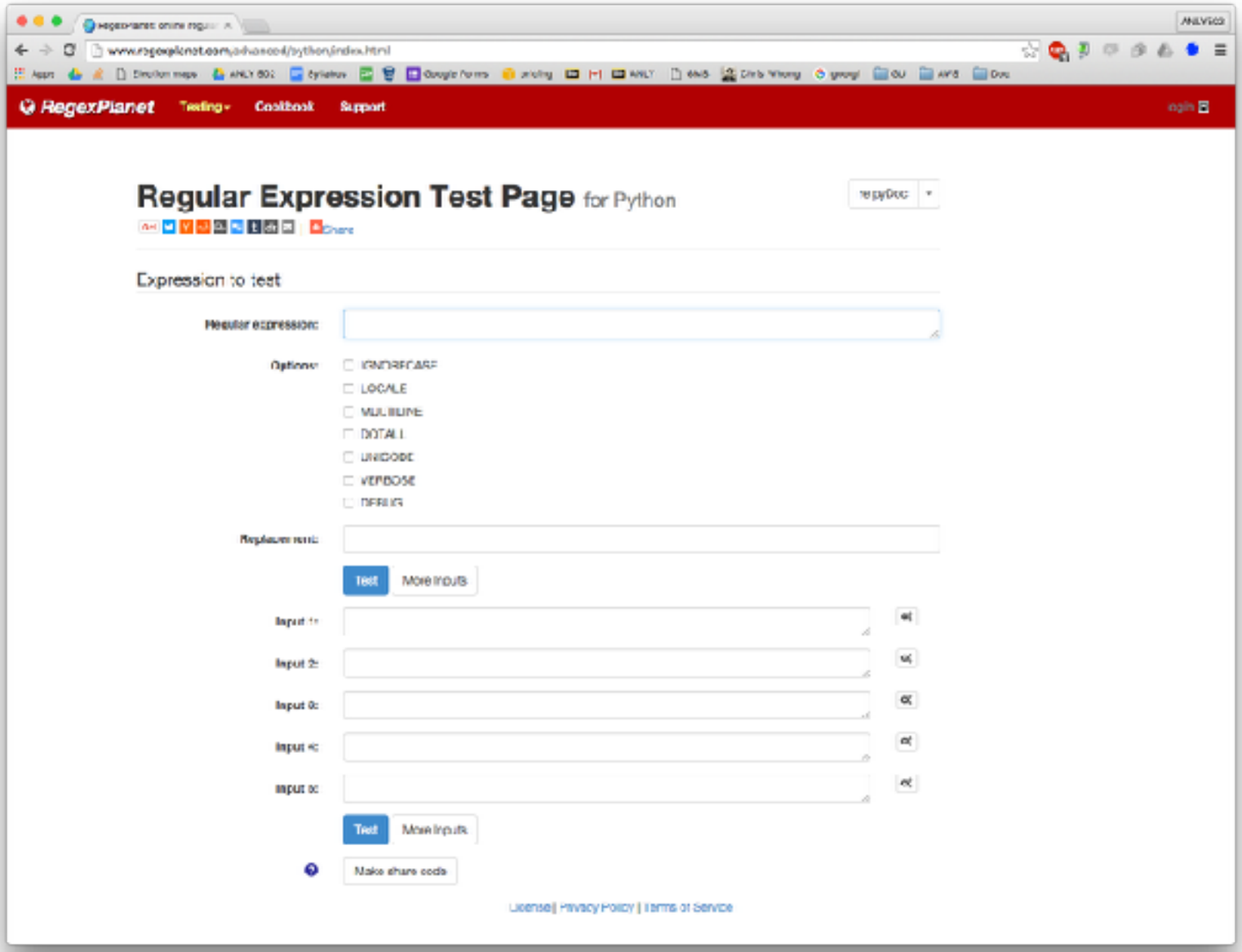
\uXXXX — Unicode character XXXX

\UXXXXXXXX — Unicode character XXXXXXXX

RegexPlanet has an online regular expression tester http://www.regexplanet.com/advanced/python/index.html

Python




RegexPlanet
Testing
Cookbook
Support
Regular Expression Test Page for Python
Expression to test
Regular expression:
Options:
LOCALE
MULTILINE
DOTALL
UNICODE
VERBOSE
Test
More inputs
Test
More inputs
Make share code

amazon new instance types 20 | Now Available: Amazon EC2 I3 | Georgetown

Secure https://aws.amazon.com/about-aws/whats-new/2017/02/now-available-amazon-ec2-i3-instances-next-generation-storage-optimized-high-i-o-instances/

Apps | 502 | Syllabus | ANLY502 | Forms | pricing | Canvas | GU | AWS | Doc | ACM Digital Library

Menu | amazon webservices | Products | Solutions | Pricing | Software | Support | More | English | My Account | Sign In to the Console

ABOUT AWS

- About AWS
- Global Infrastructure
- What's New
- AWS in the News
- Events & Webinars

RELATED LINKS

- What is Cloud Computing?
- AWS Free Usage Tier
- AWS Blog
- AWS Careers
- AWS Training

Manage Your Resources

Sign In to the Console

Now Available: Amazon EC2 I3 Instances, next-generation Storage Optimized High I/O instances

Posted On: Feb 23, 2017

Amazon EC2 I3 instances are the latest generation of Storage Optimized High-I/O instances, designed for the most demanding High I/O workloads, featuring low-latency Non-Volatile Memory Express (NVMe) based SSDs. I3 instances are ideal for workloads like transactional processing systems, relational and NoSQL databases, data warehousing applications, analytics workloads and Elasticsearch workloads.

I3 instances will come in six sizes, with up to 64 vCPUs, 488 GiB of memory, and 15.2 TB of locally attached SSD storage. I3 instances will deliver up to 3.3 million random IOPS at 4KB block size and up to 16 GB/s of sequential disk throughput. I3 instances are powered by dual socket custom Intel® Xeon® E5-2686 v4 Broadwell processors running at a base clock frequency of 2.3 GHz, feature Enhanced Networking with Amazon EC2 Elastic Network Adapter (ENA) and are EBS-optimized by default at no additional cost.

Amazon EC2 I3 instances are offered On-Demand or as Reserved Instances, for prices and region availability visit the EC2 pricing page. For details and to get started visit Amazon EC2, AWS Management Console, AWS Command Line Interface (CLI), and AWS SDKs.

AWS on Twitter | AWS on Facebook | AWS on Google+ | AWS Blog | What's New? RSS | Sign In to the Console

AWS & Cloud Computing | Solutions | Resources & Training | Manage Your Account | Amazon Web Services is Hiring.

New Instance Types!

Storage Optimized

I3 – High I/O Instances

This family includes the High Storage Instances that provide Non-Volatile Memory Express (NVMe) SSD backed instance storage optimized for low latency, very high random I/O performance, high sequential read throughput and provide high IOPS at a low cost.

Features:

- High Frequency Intel Xeon E5-2686 v4 (Broadwell) Processors with base frequency of 2.3 GHz
- NVMe SSD Storage
- Support for TRIM
- Support for Enhanced Networking
- High Random I/O performance and High Sequential Read throughput

Model	vCPU	Mem (GiB)	Networking Performance	Storage (TB)	
i3.large	2	15.25	Up to 10 Gigabit	1 x 0.475 NVMe SSD	$0.156/hour
i3.xlarge	4	30.5	Up to 10 Gigabit	1 x 0.95 NVMe SSD	$0.312/hour
i3.2xlarge	8	61	Up to 10 Gigabit	1 x 1.9 NVMe SSD	$0.624/hour
i3.4xlarge	16	122	Up to 10 Gigabit	2 x 1.9 NVMe SSD	$1.248/hour
i3.8xlarge	32	244	10 Gigabit	4 x 1.9 NVMe SSD	$2.496/hour
i3.16xlarge	64	488	20 Gigabit	8 x 1.9 NVMe SSD	$4.992/hour

Use Cases

NoSQL databases like Cassandra, MongoDB, Redis, in-memory databases such as Aerospike, scale out transactional databases, data warehousing, Elasticsearch, analytics workloads.

Databases

Databases — A place to put data

1832 — Semen Korsakov uses punch cards for data storage in Russia

1890 — Herman Hollerith invests paper punch cards for 1890 Census

- 1896 Hollerith founds Tabulating Machine Company, which later becomes IBM

1964 — SABRE system goes online (American Airlines & IBM)

- Two database models: network model (CODASYL) and hierarchical model (IMS)

1970-1972 — E. F. Codd invents relational database model and early SQL

- http://dl.acm.org/citation.cfm?doid=362384.362685

1986 - Structured Query Language (SQL) standardized by ANSI

1990s — Object databases created for object-oriented languages

2000s — Speciality databases created. Neo4j Graph database; CouchDB "NoSQL" database

References:

- https://www.ssa.gov/history/cronin.html
- http://www.quickbase.com/articles/timeline-of-database-history
- https://www.sabre.com/files/Sabre-History.pdf
- http://blog.knuthaugen.no/2010/03/a-brief-history-of-nosql.html

Relations stored as text files

Four students: Alice, Bob, Charlie & Debra

students.txt:

```
1, Alice
2, Bob
3, Charlie
4, Debra
```

Relations stored as text files

Four students: Alice, Bob, Charlie & Debra

Four courses: ANLY501, ANLY502, ANLY503, ANLY511

students.txt:

```
1, Alice
2, Bob
3, Charlie
4, Debra
```

courses.txt:

```
1, ANLY501
2, ANLY502
3, ANLY503
4, ANLY511
```

Relations stored as text files

Four students: Alice, Bob, Charlie & Debra

Four courses: ANLY501, ANLY502, ANLY503, ANLY511

Alice is enrolled in ANLY502 & ANLY503, Bob and Charlie are in ANLY 502

students.txt:

```
1, Alice
2, Bob
3, Charlie
4, Debra
```

courses.txt:

```
1, ANLY501
2, ANLY502
3, ANLY503
4, ANLY511
```

enrollments.txt:

```
1, 2
1, 3
2, 2
3, 2
```

Key database concepts — Relational Databases

Data stored in "tables" with relations between them.

- Each table: **Schema, Rows, Primary Key:**
- Tables can have **Foreign Keys**.

Key **Foreign Keys**

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

Enrollments

Alice is enrolled in ANLY502 and ANLY503

Bob is enrolled in ANLY502

Charlie is enrolled in ANLY502

Debra is not enrolled in any course.

Advantages of database keys

Efficiency — Faster to look up by key than by string

Referential Integrity — Can't have invalid relations

Easy to update — Each Name is only represented once

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

Key	Foreign Keys	
EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

Enrollments

Advantages of database keys

Efficiency — Faster to look up by key than by string

Referential Integrity — Can't have invalid relations

Easy to update — Each Name is only represented once

Students

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Courses

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Enrollments

Key	Foreign Keys	
EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

Extensibility — You can add columns:

CID	Name	Full Name
1	ANLY501	Introduction to Data Analytics
2	ANLY502	Massive Data Fundamentals
3	ANLY503	Scientific and Analytics Visualizations
4	ANLY511	Probabilistic Modeling and Statistical Computing

Document Databases

Data is stored as a series of "documents" (typically JSON)

```
{"SID":1, "name":"Alice", "courses":["ANLY502","ANLY503"]}
{"SID":2, "name":"Bob", "courses":["ANLY502","ANLY503"]}
{"SID":3, "name":"Charlie", "courses":["ANLY502"]}
{"SID":4, "name":"Debra", "courses":[]}
```

Alice is enrolled in ANLY502 and ANLY503

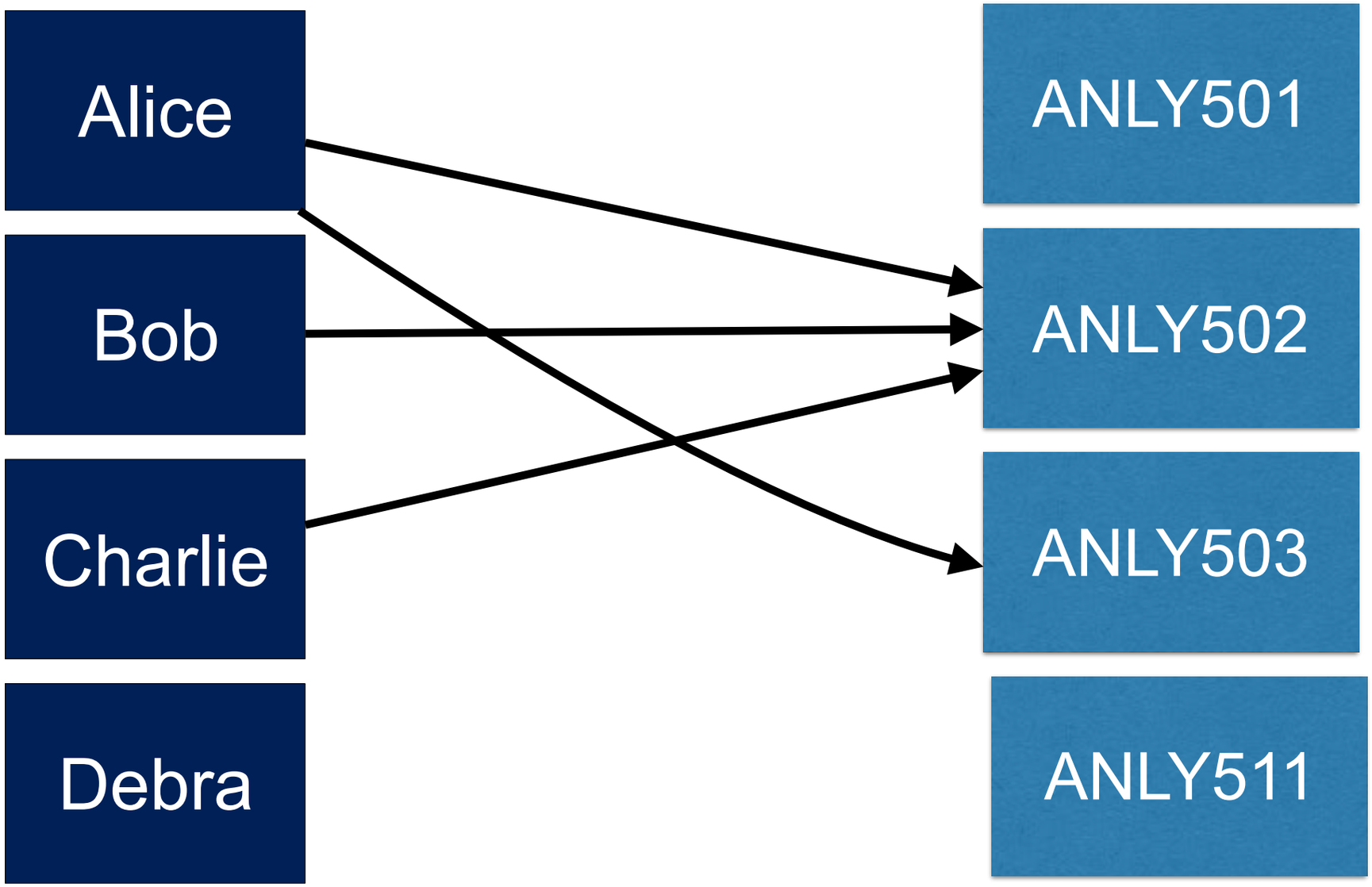
Bob is enrolled in ANLY502

Charlie is enrolled in ANLY502

Debra is not enrolled in any course.

Object-oriented / Network / Graph databases

Data is stored as objects (typically Java)



Alice is enrolled in ANLY502 and ANLY503

Bob is enrolled in ANLY502

Charlie is enrolled in ANLY502

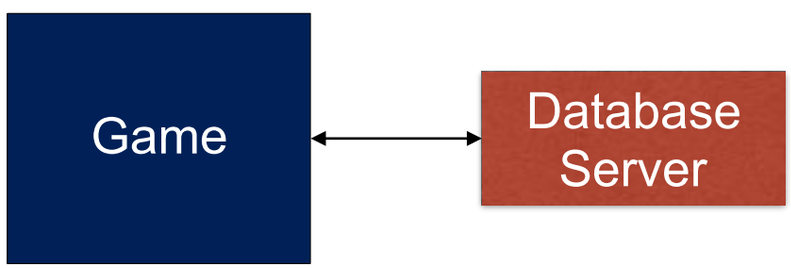
Debra is not enrolled in any course

Database access models

Embedded — The database is part of your program

Example: SQLite
Advantages: Easy to install and configure
Disadvantages: No security; single-user

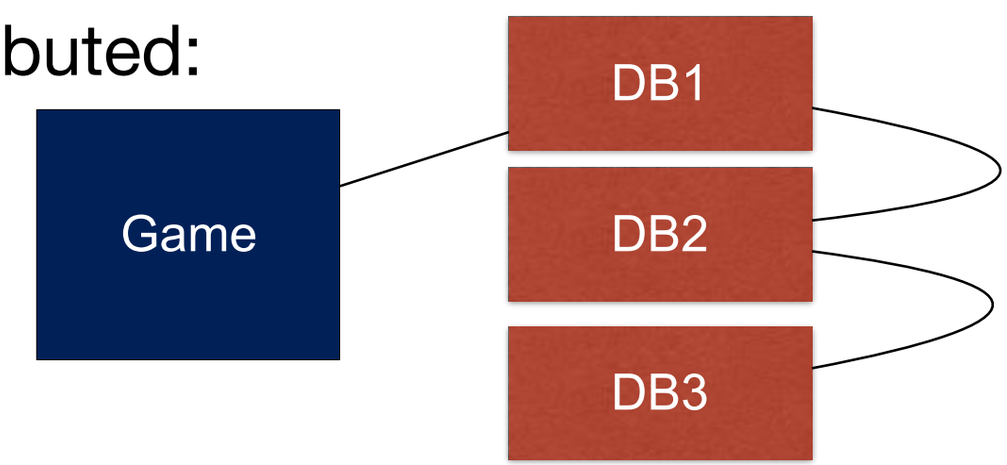
Client-Server — The database runs in a different process



Example: MySQL
Advantages:
- **Durability — Database stored elsewhere**
- **Security — Game only can access through API**

Disadvantages: Harder to set up

Distributed:



Example: Cassandra, HDFS/HBase/CouchDB
Advantages:
- **Scalability**
- **Redundancy**

Disadvantages: Even harder to set up

Updating the database — Transactions

What if we want to add two new students — Ellen & Fong?

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra
5	Ellen
6	Fong

If additions happen at the same time, there can be problems.

Transactional Databases — ACID Properties

Databases support READ & WRITE operations.

- READ — User reads data stored in database.
- WRITE — User update data in database.

A — Automocity — Transactions are atomic.

- An update either happens or it doesn't.

C — Consistency

- The database is always consistent.

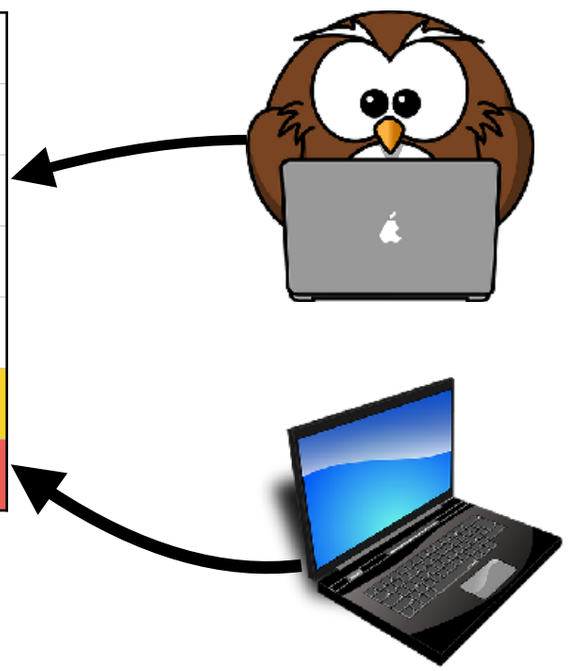
I — Isolation

- Transactions are isolated from each other

D — Durability

- The data is not lost, even if the database crashes.
- Data is replicated to other systems.

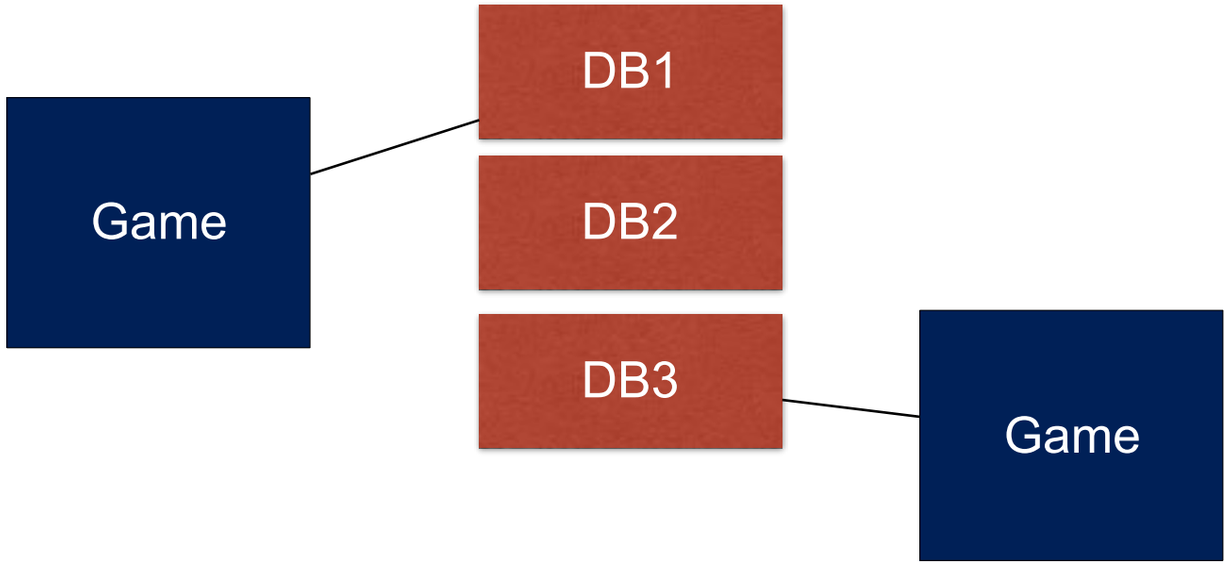
SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra
5	Ellen
6	Fong



Options for distributed databases:

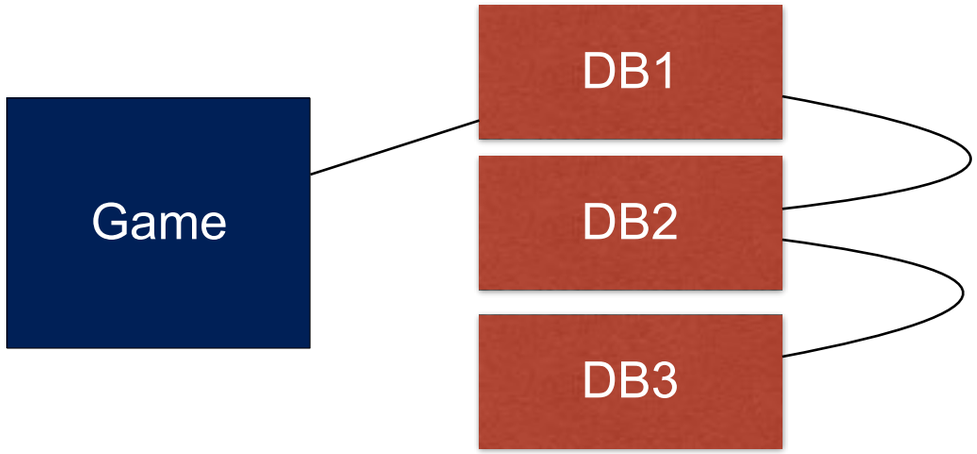
"Sharding"

- Different databases shared by different users.
- No connections between the users.
- Offers isolation & scalability, but



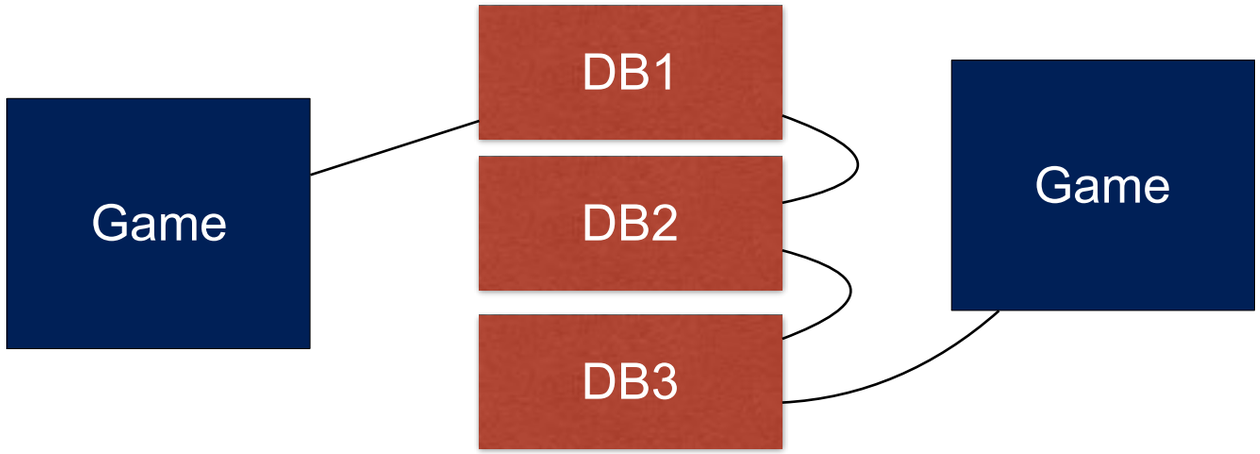
Replication with multiple readers:

- Clients WRITE to a single database.
- WRITE distributed to other databases
- READs from all databases



Fully distributed system

- Protocols needed for fail-over, master designation, etc.



Databases at Amazon

Database options at Amazon

"Hadoop" databases

- Store data in flat files in S3 or HDFS; access with Hadoop ecosysmtem.

RDS — Amazon Relational Databases (MySQL, Oracle, MS SQL Server, etc)

DynamoDB — "A fast and flexible NoSQL database service"

Redshift — "A fast, fully managed, petabyte-scale data warehouse solution"

DMS — AWS Database Migration Service

RDS · AWS Console

Georgetown

Secure https://console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:ct=ge...

Apps 502 Syllabus ANLY502 Forms pricing Canvas GU AWS Doc

Services Resource Groups EMR S3 Simson Garfinkel N. Virginia Support

Step 1: Select Engine

Free tier eligible only

Select Engine

To get started, choose a DB Engine below and click Select.

Amazon Aurora

MySQL

MariaDB

PostgreSQL

ORACLE

SQL Server

Aurora

Select

Amazon Aurora is a high-performance, MySQL-compatible, enterprise-class database at a tenth the cost of commercial databases.

- Up to 5 times the throughput of MySQL
- Up to 15 promotable Read Replicas with less than 10 ms lag.
- Up to 64 TB of Auto Scaling storage replicated over multiple Availability Zones.

Cancel

Feedback English

Privacy Policy Terms of Use

Comparing Amazon database offerings

	License	Max Size	Scaling	Replication
Amazon Aurora	Propriety	64 TB	0-15 promotable Read Replicas with 10 ms lag	read/write over multiple availability zones
MySQL	Open Source	6TB	32 vCPUs and 244GiB RAM	cross-region read replicas
MariaDB	Open Source	6TB	32 vCPUs and 244 GiB RAM	cross-region read replicas.
PostgreSQL	Open Source	> 16TB Must use multi-volume	32 vCPUs and 244 GiB RAM	Within DBMS
Oracle	Propriety	> 16TB Must use multi-volume	32 vCPUs and 244 GiB RAM	Within DBMS
Microsoft SQLServer	Proprietary	> 16TB Must use multi-volume	32 vCPUs and 244 GiB RAM	Within DBMS

Setting up Aurora

RDS - AWS Console

Secure https://console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:ct=gettingStarted:

Services | Resource Groups | EMR | S3 | Simson Garfinkel | N. Virginia | Support

Step 1: Select Engine

Step 2: Specify DB Details

Step 3: Configure Advanced Settings

The following selections disqualify the instance from being eligible for the free tier:

- DB Instance Class
- Engine

You will be charged normal RDS Prices. Learn More.

Estimate your monthly costs for the DB Instance using the RDS Instance Cost Calculator.

Specify DB Details

Instance Specifications

DB Engine Aurora - compatible with MySQL 5.6.10a

DB Instance Class db.t2.small — 1 vCPU, 2 GiB RAM

Multi-AZ Deployment No

Settings

DB Instance Identifier* ANLY502DB

Master Username*

Master Password*

Confirm Password*

Specify an alphanumeric string that defines the login ID for the master user. You use the master user login to start defining all users, objects, and permissions in the databases of your DB Instance. Master Username must start with a letter, as in 'awsuser'.

* Required

Cancel | Previous | Next Step

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

Co

RDS - AWS Console

Secure https://console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:ct=gettingStarted:

Apps 502 Syllabus ANLY502 Forms pricing Canvas GU AWS Doc ACM Digital Library

Services Resource Groups EMR S3 Simson Garfinkel N. Virginia Support

Step 1: Select Engine

Step 2: Specify DB Details

Step 3: Configure Advanced Settings

Configure Advanced Settings

Network & Security

VPC*	Default VPC (vpc-13401067)
Subnet Group	default
Publicly Accessible	Yes
Availability Zone	No Preference
VPC Security Group(s)	Create new Security Group ElasticMapReduce-master (VPC) ElasticMapReduce-slave (VPC) SSH Open (VPC)

Database Options

DB Cluster Identifier	
Database Name	any502
Database Port	3306
DB Parameter Group	default.aurora5.6
DB Cluster Parameter Group	
Option Group	default:aurora-5-6
Enable Encryption	No

Select the DB parameter group that defines the configuration settings you want applied to this DB instance. Learn More.

Feedback English © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

RDS - AWS Console

Secure https://console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:ct=gettingStarted:

Services | Resource Groups | EMR | S3 | Simson Garfinkel | N. Virginia | Support

Enable Encryption: No

Failover

Priority: - Select One -

Backup

Backup Retention Period: 1 days

Monitoring

Enable Enhanced Monitoring: Yes

Monitoring Role: Default

Granularity: 1 second(s)

I authorize RDS to create the IAM role rds-monitoring-role.

Maintenance

Auto Minor Version Upgrade: Yes

Maintenance Window: Select Window

Start Day: Thursday

Start Time: 08 : 00 UTC

Duration: 0.5 hours

Select the period in which you want pending modifications (such as changing the DB instance class) or patches applied to the DB instance by Amazon RDS. Any such maintenance should be started and completed within the selected period. If you do not select a period, Amazon RDS will assign a period randomly. Learn More.

* Required

Cancel | Previous | Launch DB Instance

Feedback | English | © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

An introduction to SQL

SQL — Structured Query Language

A single language for communicating with databases.

Used by all kind of databases — embedded, client/server, distributed

Used by SparkSQL for accessing data frames.

SQL commands — designed to read like English.

Most have VERB NOUN ADVERB structure.

- e.g. "WALK HERE QUICKLY."

SQL Commands you need to know:

- CREATE — Creates a database or table
- SELECT — Gets data
- INSERT — Inserts new data
- UPDATE — Changes existing data

CREATE TABLE — Creates a table
DROP TABLE — Deletes a table

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

Syntax: CREATE TABLE *table_name* (col1 datatype [, ...]);

Convention:

- UPPERCASE — SQL Statements
- lowercase — user-defined tables, columns, etc.

Example:

```
$ sqlite3
sqlite> CREATE TABLE students (sid PRIMARY KEY, name VARCHAR(255));
sqlite> CREATE TABLE courses (cid PRIMARY KEY, name VARCHAR(255));
```

CREATE TABLE — Creates a table
DROP TABLE — Deletes a table

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

```
CREATE TABLE students (sid INTEGER PRIMARY KEY, name VARCHAR(255));
CREATE TABLE courses (cid INTEGER PRIMARY KEY, name VARCHAR(255));
```

Sqlite3 Data types (4):

```
INTEGER
VARCHAR(nn)
BLOB
REAL
```

MySQL Data data types (35):

```
TEXT(size)
LONGTEXT(size)
DECIMAL
BIGINT
DATE
TIME
DATETIME
TIMESTAMP
```

Modifiers:

```
[ NULL | NOT NULL ]
[ DEFAULT default_value ]
[ UNIQUE KEY | PRIMARY KEY ]
[ AUTO_INCREMENT ] (MySql only)
```

INSERT — Inserts data
SELECT — Shows the data

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

```
sqlite> INSERT INTO students (name) VALUES ("Alice");
sqlite> SELECT * FROM students;
1|Alice
sqlite> INSERT INTO students (sid,name) VALUES (2,"Bob");
sqlite> SELECT * FROM students;
1|Alice
2|Bob
sqlite> INSERT INTO students (sid,name) values (3,"Charlie"),(4,"Debra");
sqlite> SELECT * FROM students;
1|Alice
2|Bob
3|Charlie
4|Debra
```

INSERT — Inserts data
SELECT — Shows the data

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

```
sqlite> INSERT INTO courses (cid,name) VALUES (1,"ANLY501"),
(2,"ANLY502"),(3,"ANLY503"),(4,"ANLY511");
```

Constraints link tables together

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

Enrollments

```
sqlite> PRAGMA foreign_keys = ON;
sqlite> CREATE TABLE enrollments (
    eid INTEGER NOT NULL,
    sid INTEGER NOT NULL,
    cid INTEGER NOT NULL,
    PRIMARY KEY (eid),
    FOREIGN KEY (sid) REFERENCES students (sid) ON DELETE CASCADE,
    FOREIGN KEY (cid) REFERENCES courses (cid) ON DELETE CASCADE
);
```

Constraints link tables together

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Students

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Courses

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

Enrollments

```
sqlite> INSERT INTO enrollments (eid,sid,cid) VALUES (1,1,1);
sqlite> INSERT INTO enrollments (eid,sid,cid) VALUES (1,1,1);
Error: UNIQUE constraint failed: enrollments.eid
sqlite> INSERT INTO enrollments (eid,sid,cid) VALUES (2,10,1);
Error: FOREIGN KEY constraint failed
sqlite> INSERT INTO enrollments (eid,sid,cid) VALUES (2,1,3),
(3,2,2),(4,3,2);
```

Simple statistics with WHERE and sub-selects

Students

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Courses

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Enrollments

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

```
sqlite> SELECT count(*) FROM students;
4
sqlite> SELECT sid FROM students WHERE name='Alice';
1
sqlite> SELECT count(*) FROM enrollments WHERE sid=1;
2
sqlite> SELECT count(*) FROM enrollments WHERE sid IN (SELECT sid
FROM students WHERE name='Alice');
2
sqlite>
```

Joins

Students

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Courses

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Enrollments

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

```
sqlite> SELECT count(*) FROM students;
4
sqlite> SELECT sid FROM students WHERE name='Alice';
1
sqlite> SELECT count(*) FROM enrollments WHERE sid=1;
2
sqlite> SELECT count(*) FROM enrollments WHERE sid IN (SELECT sid FROM
students WHERE name='Alice');
2
sqlite> SELECT students.name,courses.name FROM enrollments
           LEFT JOIN students ON students.sid=enrollments.sid
           LEFT JOIN courses ON courses.cid=enrollments.cid;
Alice|ANLY502
Alice|ANLY503
Bob|ANLY502
Charlie|ANLY502
```

SQLite3 output formats

```
sqlite> .mode csv
sqlite> SELECT * FROM students;
1,Alice
2,Bob
3,Charlie
4,Debra
sqlite> .mode tabs
sqlite> SELECT * FROM students;
1   Alice
2   Bob
3   Charlie
4   Debra
sqlite> .mode html
sqlite> SELECT * FROM students;
<TR><TD>1</TD>
<TD>Alice</TD>
</TR>
<TR><TD>2</TD>
<TD>Bob</TD>
</TR>
<TR><TD>3</TD>
<TD>Charlie</TD>
</TR>
<TR><TD>4</TD>
<TD>Debra</TD>
</TR>
sqlite> .mode insert
sqlite> SELECT * FROM students;
INSERT INTO table VALUES(1,'Alice');
INSERT INTO table VALUES(2,'Bob');
INSERT INTO table VALUES(3,'Charlie');
INSERT INTO table VALUES(4,'Debra');
sqlite>
```

"Database normalization" — Organizing your database.

Database normalization was created to:

- Reduce redundancy & Improve data integrity

Basic idea — this is normalized:

Students

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Courses

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Enrollments

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

This is de-normalized:

SID	Name	Course1	Course2
1	Alice	ANLY502	ANLY503
2	Bob	ANLY503	
3	Charlie	ANLY502	
4	Debra	ANLY502	

"Database normalization" — Organizing your database.

Database normalization was created to:

- Reduce redundancy & Improve data integrity

Basic idea — this is normalized:

Students

SID	Name
1	Alice
2	Bob
3	Charlie
4	Debra

Courses

CID	Name
1	ANLY501
2	ANLY502
3	ANLY503
4	ANLY511

Enrollments

EID	SID	CID
1	1	2
2	1	3
3	2	2
4	3	2

This is de-normalized:

SID	Name	Course1	Course2
1	Alice	ANLY502	ANLY503
2	Bob	ANLY503	
3	Charlie	ANLY502	
4	Debra	ANLY502	

This is tidy data

SID	Name	Course
1	Alice	ANLY502
1	Alice	ANLY503
2	Bob	ANLY503
3	Charlie	ANLY502
4	Debra	ANLY502

"Normal Forms" — properties of normalized databases.
"First Normal Form" — The domain of each attribute is atomic.

This is NOT first normal form:

SID	Name	Courses
1	Alice	ANLY502, ANLY503
2	Bob	ANLY503
3	Charlie	ANLY502
4	Debra	ANLY502

With 1NF, you should not need string operations.

This is 1NF:

SID	Name	Course
1	Alice	ANLY502
1	Alice	ANLY503
2	Bob	ANLY503
3	Charlie	ANLY502
4	Debra	ANLY502

"Second Normal Form"
All non-key attributes depend on the primary key

This is NOT second normal form:

SID	Name	Course	Course Name
1	Alice	ANLY502	Massive Database Fundamentals
1	Alice	ANLY503	Scientific and Analytics Visualizations
2	Bob	ANLY503	Scientific and Analytics Visualizations
3	Charlie	ANLY502	Massive Database Fundamentals
4	Debra	ANLY502	Massive Database Fundamentals

2NF requires representing the relationship between Course and Course Name:

SID	Name	Course
1	Alice	ANLY502
1	Alice	ANLY503
2	Bob	ANLY503
3	Charlie	ANLY502
4	Debra	ANLY502

Course	Course Name
ANLY502	Massive Database Fundamentals
ANLY503	Scientific and Analytics Visualizations

"Second Normal Form"
All non-key attributes depend on the primary key

2NF does not require that keys be integers.

Option #1:

SID	Name	Course
1	Alice	ANLY502
1	Alice	ANLY503
2	Bob	ANLY503
3	Charlie	ANLY502
4	Debra	ANLY502

Course	Course Name
ANLY502	Massive Database Fundamentals
ANLY503	Scientific and Analytics Visualizations

Option #2:

SID	Name	CID
1	Alice	2
1	Alice	3
2	Bob	3
3	Charlie	2
4	Debra	2

CID	Course	Course Name
2	ANLY502	Massive Database Fundamentals
3	ANLY503	Scientific and Analytics

Integer keys are more efficient & take less space.

Lab #1

Run SQLite3 (anywhere) and try these commands.

```
PRAGMA foreign_keys = ON;
CREATE TABLE students (sid PRIMARY KEY, name VARCHAR(255));
CREATE TABLE courses (cid PRIMARY KEY, name VARCHAR(255));
CREATE TABLE enrollments (
    eid INTEGER NOT NULL,
    sid INTEGER NOT NULL,
    cid INTEGER NOT NULL,
    PRIMARY KEY (eid),
    FOREIGN KEY (sid) REFERENCES students (sid) ON DELETE CASCADE,
    FOREIGN KEY (cid) REFERENCES courses (cid) ON DELETE CASCADE
);

INSERT into students (name) VALUES ...
INSERT into courses (name) VALUES ...
INSERT into enrollments ...

SELECTs...

JOINS...
```

Connecting to Jupyter & Hue web-based interfaces
SparkSQL
Hive

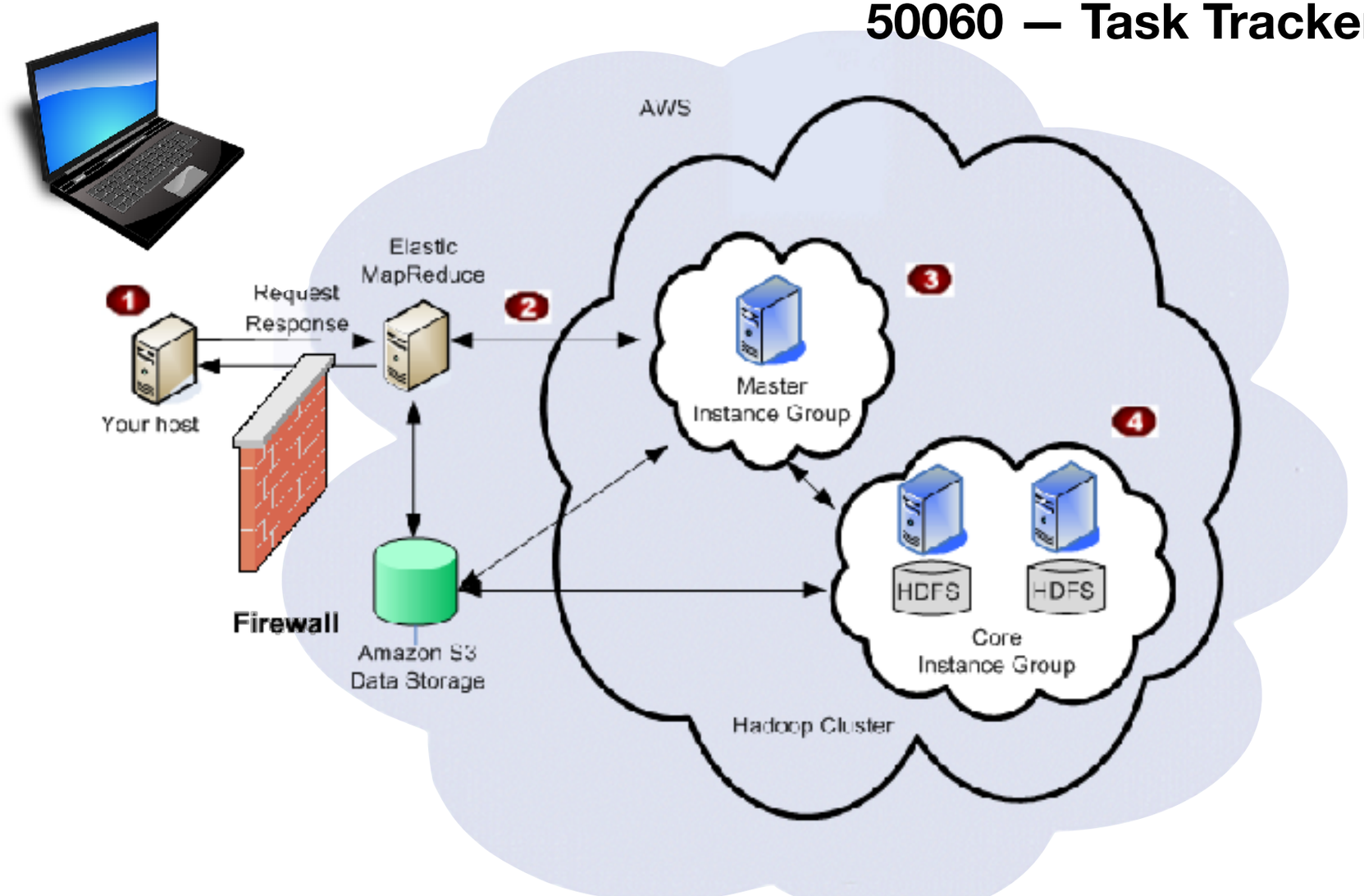
Part 2

Accessing Jupyter and Hue via SSH

We communicate with EMR with SSH over port 22

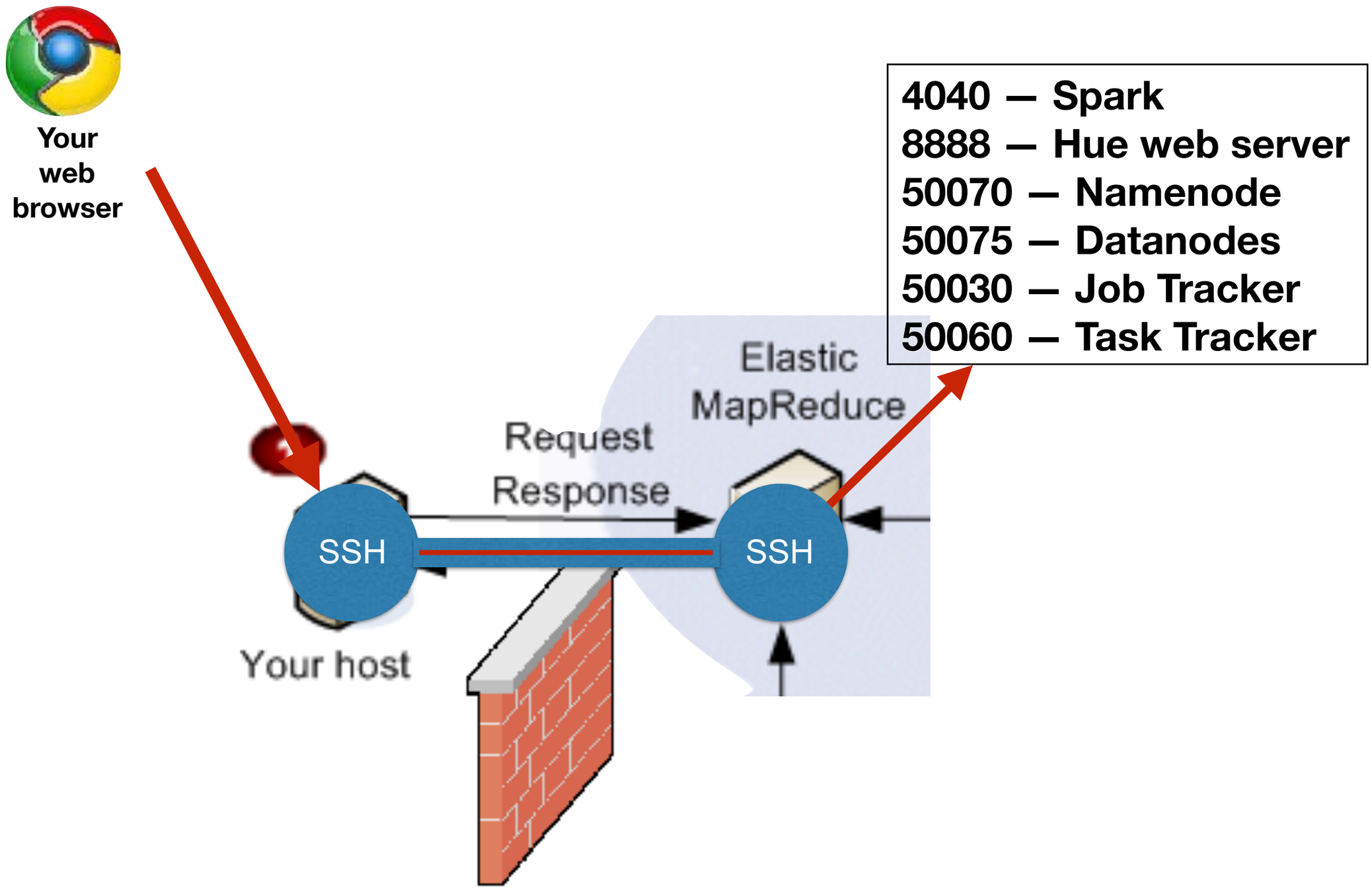
4040 — Spark
8888 — Hue web server
50070 — Namenode
50075 — Datanodes
50030 — Job Tracker
50060 — Task Tracker

Your web browser



http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-manage-resize.html

We use SSH to proxy the web connections



Easy way to use SSH — Port forwarding

$ ssh -L *localport:remoteServer:remotePort*

Several programs set up remote web servers for monitoring:

Port 8888 — Hue web server
Port 9095 — Apache HBASE ThriftServer
Port *whatever* — *Jupyter Notebook*

Jupyter notebook — It's a process with an embedded web server

You specify:

- Run the program (e.g. jupyter-notebook)
- Specify the port (--port=XXXX) (default is 8888, but that's also used by Hue!)
- Jupyter shouldn't launch a browser (--no-browser), because you will.

So:

- Log into Master with SSH port forwarding for a specific port (e.g. 9999)
- Run Jupyter with that port.

Example with port 9999:

```
$ ssh hadoop@hostname  -A  -L9999:localhost:9999
$ jupyter-notebook --no-browser --port=9999
```

—*then open* `http://localhost:9999/`

—*watch for errors in the ssh window.*

Example of errors in SSH window:

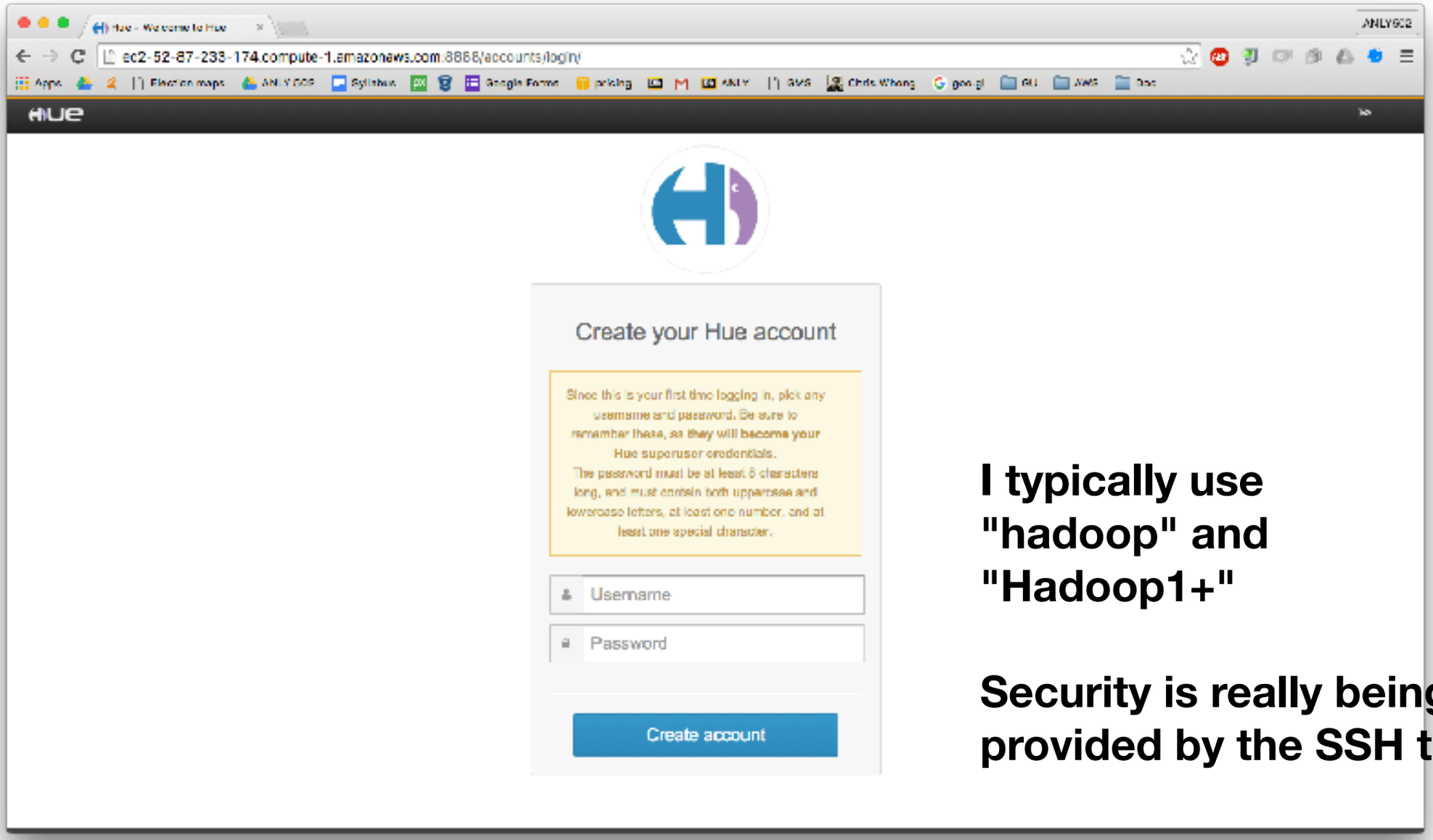
m4.large:

```
17/03/12 17:42:08 WARN Main$$anon$1: No external magics provided to PluginManager!
[W 17:42:11.508 NotebookApp] Timeout waiting for kernel_info reply from 183cc429-41b7-4570-8f79-bbd137aff0a0
17/03/12 17:42:11 WARN Shell: Parent header is null for message F4E60EC7DC4843ACB45254F4805D61E9 of type
comm_info_request
17/03/12 17:42:11 WARN Shell: Parent header is null for message F18D2ED801764B00AB482966B81EED9C of type comm_open
17/03/12 17:42:11 WARN Shell: Parent header is null for message D19F7EE93D52470082B777981E8626FD of type comm_open
17/03/12 17:42:12 WARN StandardComponentInitialization$$anon$1: Locked to Scala interpreter with SparkIMain until
decoupled!
17/03/12 17:42:12 WARN StandardComponentInitialization$$anon$1: Unable to control initialization of REPL class server!
17/03/12 17:42:16 ERROR SparkContext: Error initializing SparkContext.
java.lang.IllegalArgumentException: Required executor memory (4608+460 MB) is above the max threshold (3072 MB) of
this cluster! Please check the values of 'yarn.scheduler.maximum-allocation-mb' and/or
'yarn.nodemanager.resource.memory-mb'.
    at org.apache.spark.deploy.yarn.Client.verifyClusterResources(Client.scala:333)
    at org.apache.spark.deploy.yarn.Client.submitApplication(Client.scala:167)
    at org.apache.spark.scheduler.cluster.YarnClientSchedulerBackend.start(YarnClientSchedulerBackend.scala:56)
    at org.apache.spark.scheduler.TaskSchedulerImpl.start(TaskSchedulerImpl.scala:156)
    at org.apache.spark.SparkContext.<init>(SparkContext.scala:509)
    at org.apache.spark.SparkContext$.getOrCreate(SparkContext.scala:2313)
    at org.apache.spark.sql.SparkSession$Builder$$anonfun$6.apply(SparkSession.scala:868)
    at org.apache.spark.sql.SparkSession$Builder$$anonfun$6.apply(SparkSession.scala:860)
    at scala.Option.getOrElse(Option.scala:121)
    at org.apache.spark.sql.SparkSession$Builder.getOrCreate(SparkSession.scala:860)
    at org.apache.toree.kernel.api.Kernel.createSparkContext(Kernel.scala:349)
    at org.apache.toree.kernel.api.Kernel.createSparkContext(Kernel.scala:368)
    at
org.apache.toree.boot.layer.StandardComponentInitialization$class.initializeSparkContext(ComponentInitialization.scala
:103)
    at org.apache.toree.Main$$anon$1.initializeSparkContext(Main.scala:35)
    at
org.apache.toree.boot.layer.StandardComponentInitialization$class.initializeComponents(ComponentInitialization.scala:
88)
    at org.apache.toree.Main$$anon$1.initializeComponents(Main.scala:35)
    at org.apache.toree.boot.KernelBootstrap.initialize(KernelBootstrap.scala:101)
    at org.apache.toree.Main$.delayedEndpoint$org$apache$toree$Main$1(Main.scala:40)
    at org.apache.toree.Main$delayedInit$body.apply(Main.scala:24)
    at scala.Function0$class.apply$mcV$sp(Function0.scala:34)
    at scala.runtime.AbstractFunction0.apply$mcV$sp(AbstractFunction0.scala:12)
    at scala.App$$anonfun$main$1.apply(App.scala:76)
    at scala.App$$anonfun$main$1.apply(App.scala:76)
```

"java.lang.IllegalArgumentException: Required executor memory (4608+460 MB) is above the max threshold (3072 MB) of this cluster! Please check the values of 'yarn.scheduler.maximum-allocation-mb' and/or 'yarn.nodemanager.resource.memory-mb'."

Model	vCPU	Mem (GiB)	SSD Storage (GB)	Dedicated EBS Bandwidth (Mbps)
m4.large	2	8	EBS-only	450
m4.xlarge	4	16	EBS-only	750
m4.2xlarge	8	32	EBS-only	1,000
m4.4xlarge	16	64	EBS-only	2,000
m4.10xlarge	40	160	EBS-only	4,000
m4.16xlarge	64	256	EBS-only	10,000

The Hue account needs to be created for each cluster



I typically use "hadoop" and "Hadoop1+"

Security is really being provided by the SSH tunnel.

Hue has a Quick Start with Examples…

Hue - Quick Start

localhost:8888/about/

Query Editors | Notebooks | Data Browsers | Workflows

About Hue | Quick Start | Configuration | Server Logs

Quick Start Wizard - Hue™ 3.11.0 - The Hadoop UI

Step 1: Check Configuration | Step 2: Examples | Step 3: Users | Step 4: Go!

Checking current configuration

Configuration files located in `/etc/hue/conf.empty`

All OK. Configuration check passed.

Back | Next

Hue and the Hue logo are trademarks of Cloudera, Inc.

Hue - Quick Start

localhost:8888/about/#step2

Search

Query Editors | Notebooks | Data Browsers | Workflows

About Hue | Quick Start | Configuration | Server Logs

Quick Start Wizard - Hue™ 3.11.0 - The Hadoop UI

Step 1: Check Configuration | Step 2: Examples | Step 3: Users | Step 4: Go!

Install individual application examples

- Hive
- Oozie Editor/Dashboard
- HBase Browser

Installing...

Back | Next

With Hue, you can easily monitor the progress of your MapReduce jobs.

Hue - Job Browser

localhost:8888/jobbrowser/

Query Editors | Notebooks | Data Browsers | Workflows

Job Browser

Username hadoop Text Search for text

Succeeded | Running | Failed | Killed

Logs	ID	Name	Application Type	Status	User	Maps	Reduces	Queue	Priority	Duration	Submitted
	1489344211472_0001	PySparkShell	SPARK	RUNNING	hadoop	10%	10%	default	N/A	9s	03/12/17 18:25:14

Showing 1 to 1 of 1 entries

← Previous | 1 | Next →

Hue has a HDFS browser

Hue - File Browser

localhost:8888/filebrowser/

Search

Query Editors | Notebooks | Data Browsers | Workflows

File Browser

Search for file name | Actions | Move to trash | Upload | New

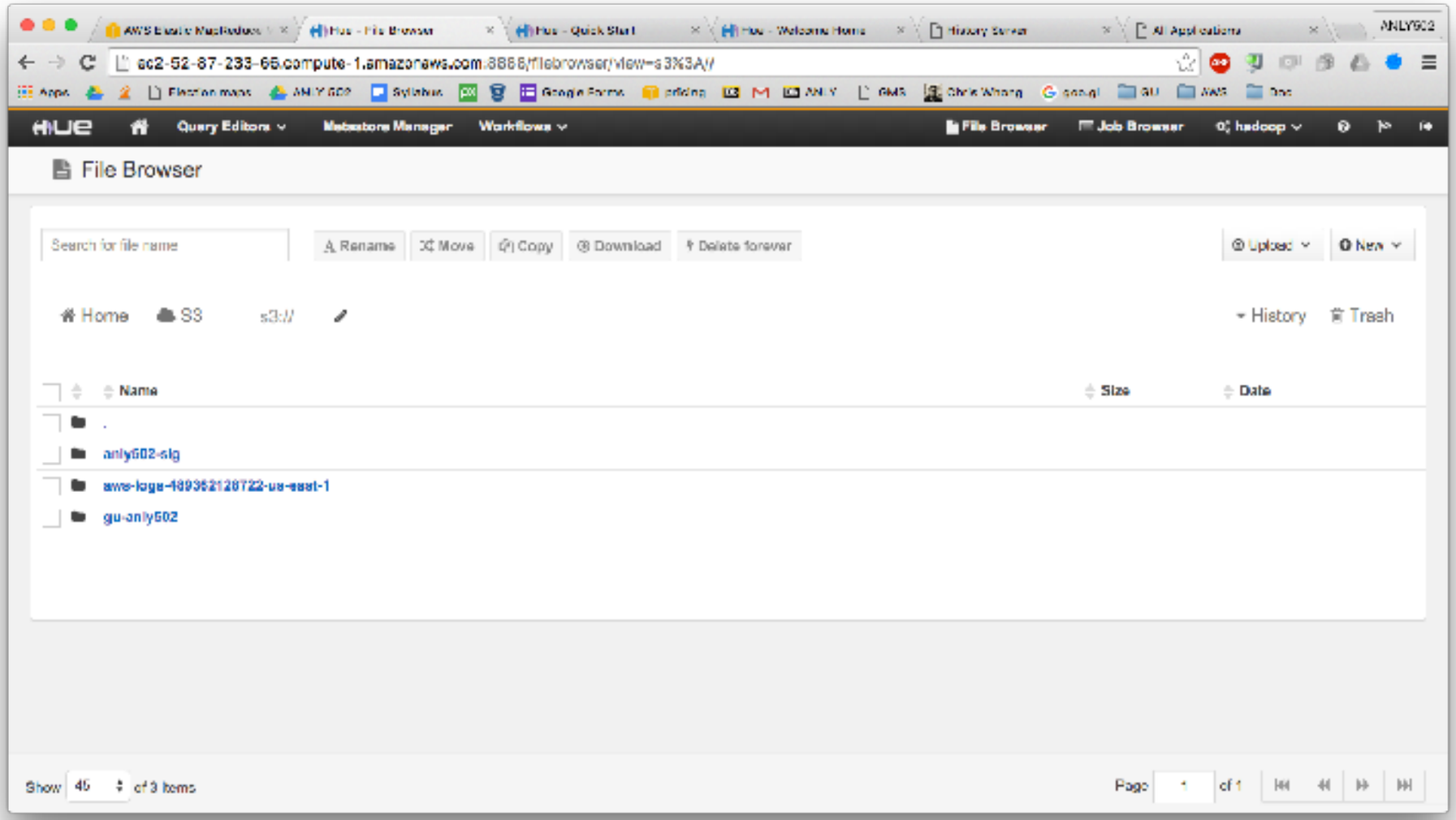
Home / user / hadoop | History | Trash

Name	Size	User	Group	Permissions	Date
1		hdfs	hadoop	drwxr-xr-x	March 12, 2017 06:21 PM
.		hadoop	hadoop	drwxrwxrwx	March 12, 2017 11:43 AM

Show 45 of 0 items

Page 1 of 1

The Hue browser will also browse S3



You can also monitor jobs with the command line

My error:

```
In [3]: 17/03/13 01:30:23 WARN YarnScheduler: Initial job has not accepted any resources; check your
cluster UI to ensure that workers are registered and have sufficient resources
17/03/13 01:30:38 WARN YarnScheduler: Initial job has not accepted any resources; check your cluster
UI to ensure that workers are registered and have sufficient resources
```

Kill the jobs:

```
[hadoop@ip-172-31-32-37 ~]$ yarn application -list
17/03/13 01:30:46 INFO impl.TimelineClientImpl: Timeline service address: http://
ip-172-31-32-37.ec2.internal:8188/ws/v1/timeline/
17/03/13 01:30:47 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-32-37.ec2.internal/
172.31.32.37:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED, RUNNING]):1
                Application-Id      Application-Name    Application-Type        User      Queue
State        Final-State         Progress
application_1489344211472_0002          PySparkShell               SPARK      hadoop    default
RUNNING          UNDEFINED              10%
[hadoop@ip-172-31-32-37 ~]$ yarn application -kill application_1489344211472_0002
17/03/13 01:31:00 INFO impl.TimelineClientImpl: Timeline service address: http://
ip-172-31-32-37.ec2.internal:8188/ws/v1/timeline/
17/03/13 01:31:00 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-32-37.ec2.internal/
172.31.32.37:8032
Killing application application_1489344211472_0002
17/03/13 01:31:00 INFO impl.YarnClientImpl: Killed application application_1489344211472_0002
```

Spark SQL

This is easy!

Spark SQL —

Spark Summit 2013 — December 2013

- 1 Developer
- Able to run simple queries on data stored in Hive

Spark Summit 2014

- 44 contributors
- Supports data stored in Hive, Parquet, JSON
- Bindings for Scala, Java & Python

Spark Summit 2016

- SparkSQL is part of main DataFrame abstraction.

Catalyst Optimizer
Relational algebra + Expressions
Query Optimization

Spark SQL Core
Execution of queries as
RDDsReading in Parquet, JSON

Hive Support
HQL, MetaStore, SerDes, UDFs

Spark SQL is an SQL-interface to Spark.

Create a data source (RDD, List of JSON objects, ...)

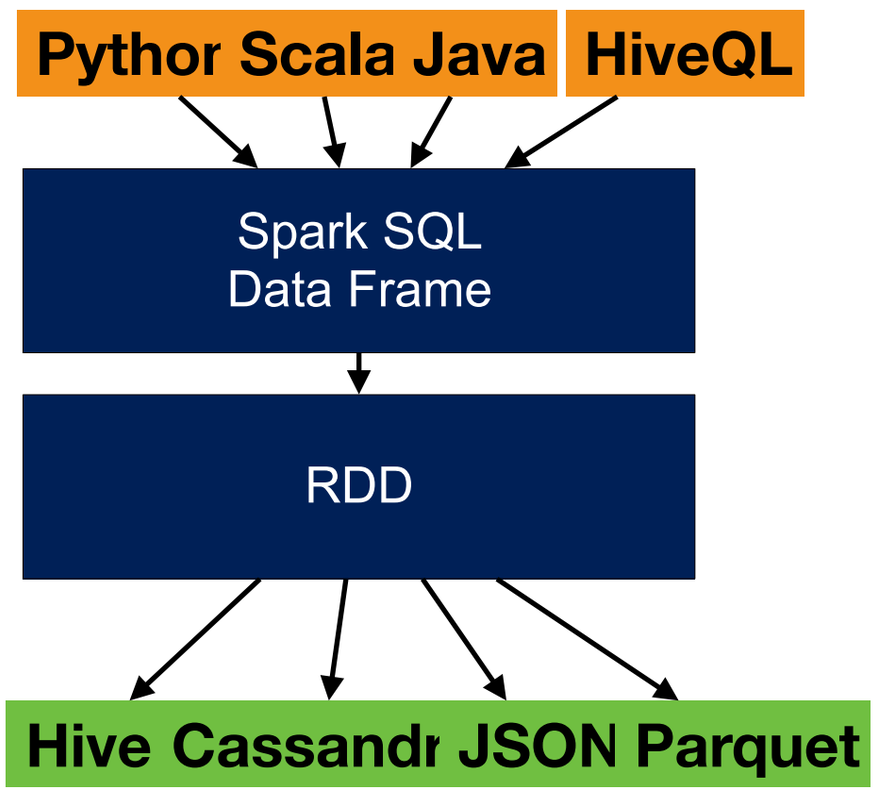
Bind the data to a Data Frame

- CreateTemporaryTable

Access via SQL

- SELECT * statements return Data Frames

Collection of Row() objects



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.1.0
      /_/

Using Python version 3.4.3 (default, Sep  1 2016 23:33:38)
SparkSession available as 'spark'.
```

Creating an RDD from a CSV file

s3://gu-anly502/data/presidents.csv:

```
num,name,url,begin,end,party,state
1,George Washington,http://en.wikipedia.org/wiki/George_Washington,
30/04/1789,4/03/1797,Independent,Virginia
2,John Adams,http://en.wikipedia.org/wiki/John_Adams,
4/03/1797,4/03/1801,Federalist,Massachusetts
3,Thomas Jefferson,http://en.wikipedia.org/wiki/Thomas_Jefferson,
4/03/1801,4/03/1809,Democratic-Republican,Virginia
4,James Madison,http://en.wikipedia.org/wiki/James_Madison,
4/03/1809,4/03/1817,Democratic-Republican,Virginia
...
```

Create an RDD, select name & party, and save in a file:

```
In [44]: df =
sqlContext.read.format("com.databricks.spark.csv").options(header="true",inferschema="true").load("s3://
gu-anly502/data/presidents.csv")

In [45]: df
Out[45]: DataFrame[num: int, name: string, url: string, begin: string, end: string, party: string, state:
string]

In [46]: df.select("name","party").write.format("com.databricks.spark.csv").save("name-party.csv")
[hadoop@ip-172-31-40-139 ~]$ hdfs dfs -ls name-party.csv
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2017-03-13 02:51 name-party.csv/_SUCCESS
-rw-r--r--   1 hadoop hadoop       1280 2017-03-13 02:51 name-party.csv/part-00000-29a15b43-bbdc-4eb6-
ad07-1b4dd353306f.csv
[hadoop@ip-172-31-40-139 ~]$ hdfs dfs -cat name-party.csv/* | head -5
George Washington,Independent
John Adams,Federalist
Thomas Jefferson,Democratic-Republican
James Madison,Democratic-Republican
James Monroe,Democratic-Republican
```

Registering dataframe as an SQL table

Turn it into an DataFrame / SQL Table:

```
In [7]: df.registerTempTable("pres")
In [8]: sqlCtx.sql("select name,party from pres limit 5").collect())
```

```
Out[8]:
[Row(num=1, name='George Washington', url='http://en.wikipedia.org/
wiki/George_Washington', begin='30/04/1789', end='4/03/1797',
party='Independent', state='Virginia'),
 Row(num=2, name='John Adams', url='http://en.wikipedia.org/wiki/
John_Adams', begin='4/03/1797', end='4/03/1801', party='Federalist',
state='Massachusetts'),
 Row(num=3, name='Thomas Jefferson', url='http://en.wikipedia.org/
wiki/Thomas_Jefferson', begin='4/03/1801', end='4/03/1809',
party='Democratic-Republican', state='Virginia'),
 Row(num=4, name='James Madison', url='http://en.wikipedia.org/wiki/
James_Madison', begin='4/03/1809', end='4/03/1817',
party='Democratic-Republican', state='Virginia'),
 Row(num=5, name='James Monroe', url='http://en.wikipedia.org/wiki/
James_Monroe', begin='4/03/1817', end='4/03/1825',
party='Democratic-Republican', state='Virginia')]
```

Convert the array of dictionaries into a Data Frame:

sqlCtx is the SparkSQL Context.

— *Use sqlCtx.createDataFrame():*

```
df.registerTempTable("pres")
```

Registers the table

Then issue SQL:

Uses registered table

```
rdd = sqlCtx.sql("select name,party from pres")
```

This works for a Data Frame of 45 rows, or 45 million rows.

Hive tables are pre-registered

You can do a lot with a data frame!

Arbitrary SQL:

```
rdd = sqlCtx.sql("select name,party from pres where name like 'George%' ")
```

User defined functions! (in any language)

```
>>> from pyspark.sql.types import IntegerType
>>> sqlContext.udf.register("stringLengthInt", lambda x: len(x),
IntegerType())
>>> sqlContext.sql("SELECT stringLengthInt('test')").collect()
```

Great functions:

```
df.show()          – Show the contents of the Data Frame
df['name']         – Refers to the column 'name'
df.filter()        – Filter an DataFrame to product another one
```

JDBC:

- JDBC client to connect to other databases
- JDBC server for clients.

Read the docs!

- http://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Introducing Hive

Apache Hive

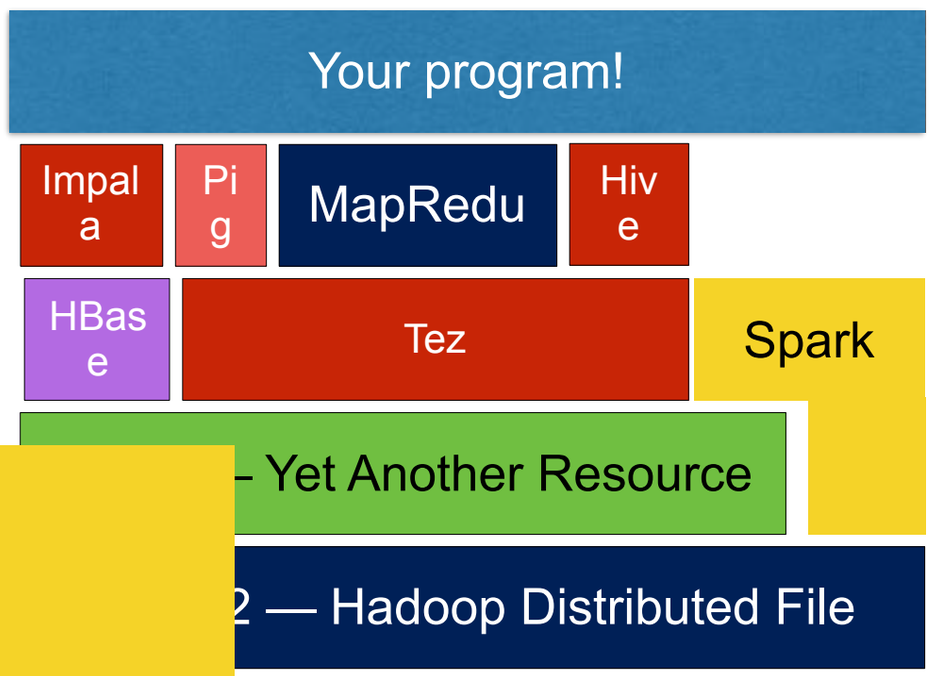
A standard interface for data analysis in Hadoop

SQL-like language called HiveQL for querying data.

Scalable — Turns HQL queries into MapReduce Jobs

Extensible

- Plug-ins for new data sources
- User Defined Functions
- Also works with HBase, Spark, etc.



Hive and Impala both provide SQL interfaces.
Hive may scale better.
See:
http://hortonworks.com/blog/impala-vs-hive-performance-benchmark/

Hive is Hadoop's "data warehouse."

Hive gives an SQL-like interface to "big data"

Hadoop provides:

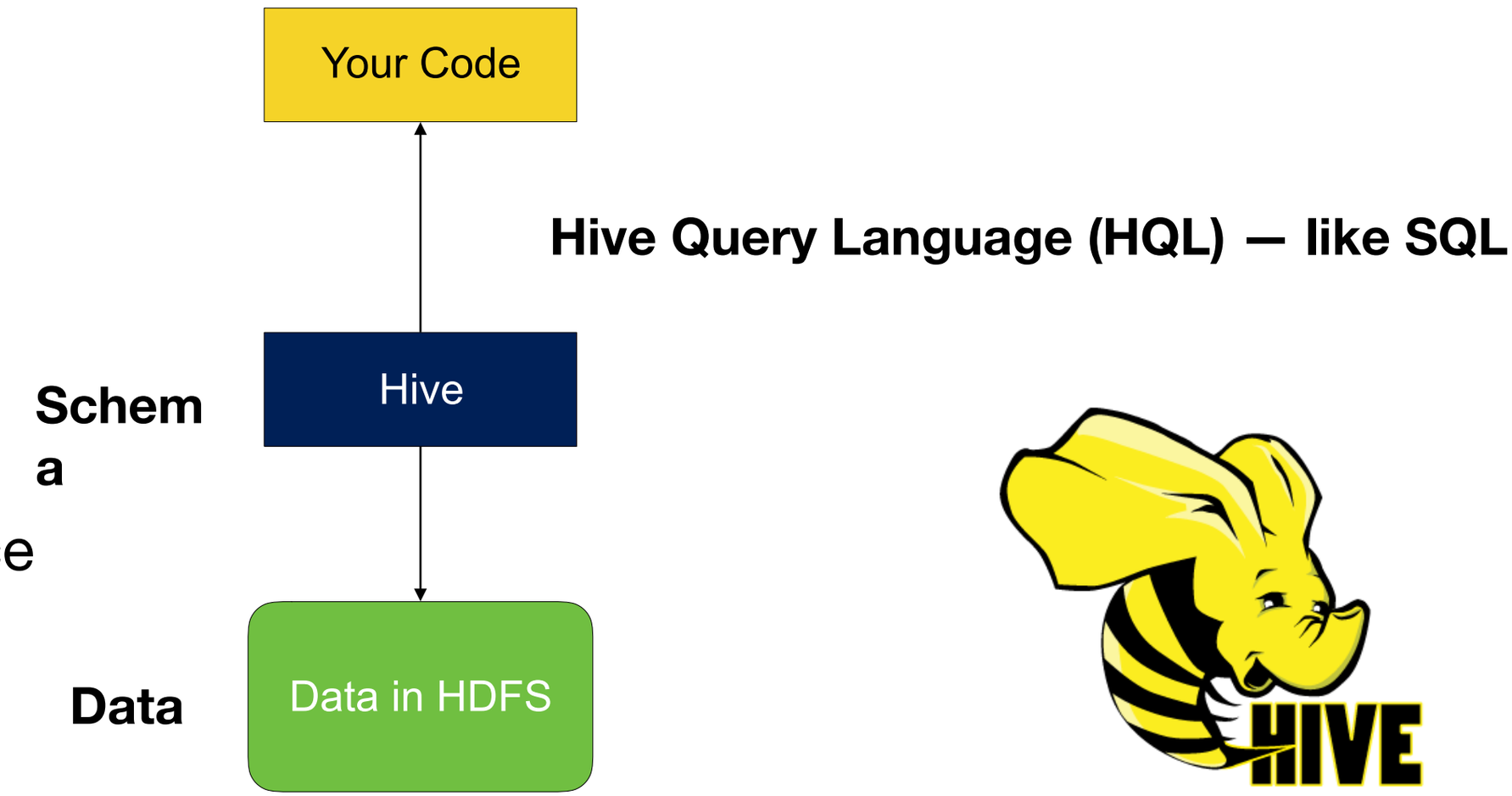
- Massive scale
- Fault tolerance

Hive provides:

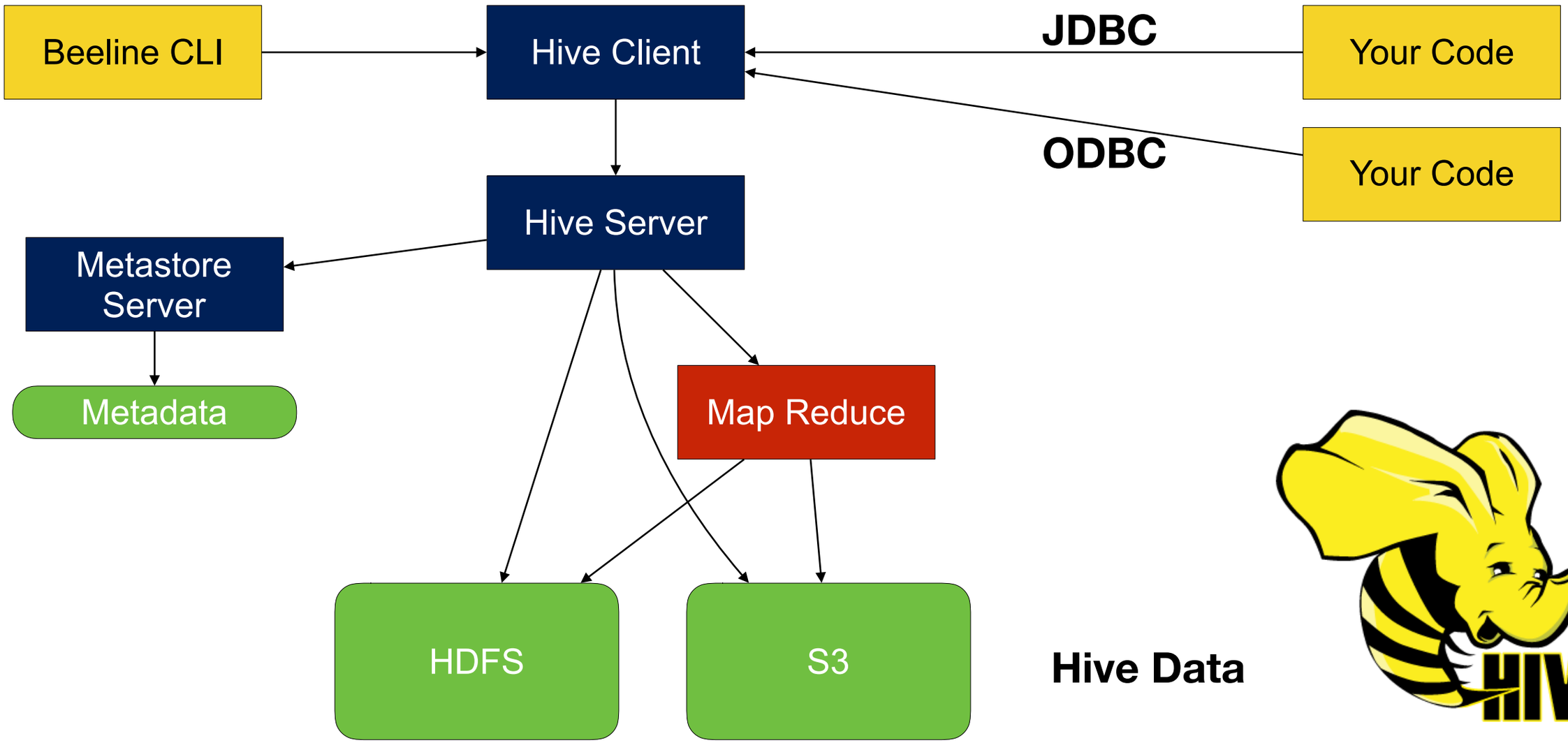
- Data summarization
- ad-hoc querying
- Simple query interface

But it's still map reduce:

- Batch processing



Hive is Hadoop's "data warehouse."



Hive does not store data. Hive points to data.

Database:
Multiple
Tables

Table 1
Rows of data with the same schema

Table 2

Table 3

Partitions.
Refers to a subset of data in the table

Hive data types — similar to MySQL (and Pig, and Java...)

"Types are associated with the columns in the tables.
The following Primitive types are supported:"

- Integers
  - *TINYINT - 1 byte integer*
  - *SMALLINT - 2 byte integer*
  - *INT - 4 byte integer*
  - *BIGINT - 8 byte integer*
- Boolean type
  - *BOOLEAN - TRUE/FALSE*
- Floating point numbers
  - *FLOAT - single precision*
  - *DOUBLE - Double precision*
- String type
  - *STRING - sequence of characters in a specified character set*

Command line interface

Hive 1 — "hive" command

- hive command can run as either client or server.
- Supports interactive & batch modes (like pig)

```
$ hive
```

Note: Does not work on EMR

Hive 2 — "Hive" server; many clients.

- Clients communicate with server via JODBC or Thrift

```
$ beeline
beeline> !connect jdbc:hive2://localhost:10000
scan complete in 6ms
Connecting to jdbc:hive2://localhost:10000
Enter username for jdbc:hive2://localhost:10000: hadoop
Enter password for jdbc:hive2://localhost:10000: ******
Connected to: Apache Hive (version 1.0.0-amzn-2)
Driver: Hive JDBC (version 1.0.0-amzn-2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000>
```

EMR: hadoop/hadoop

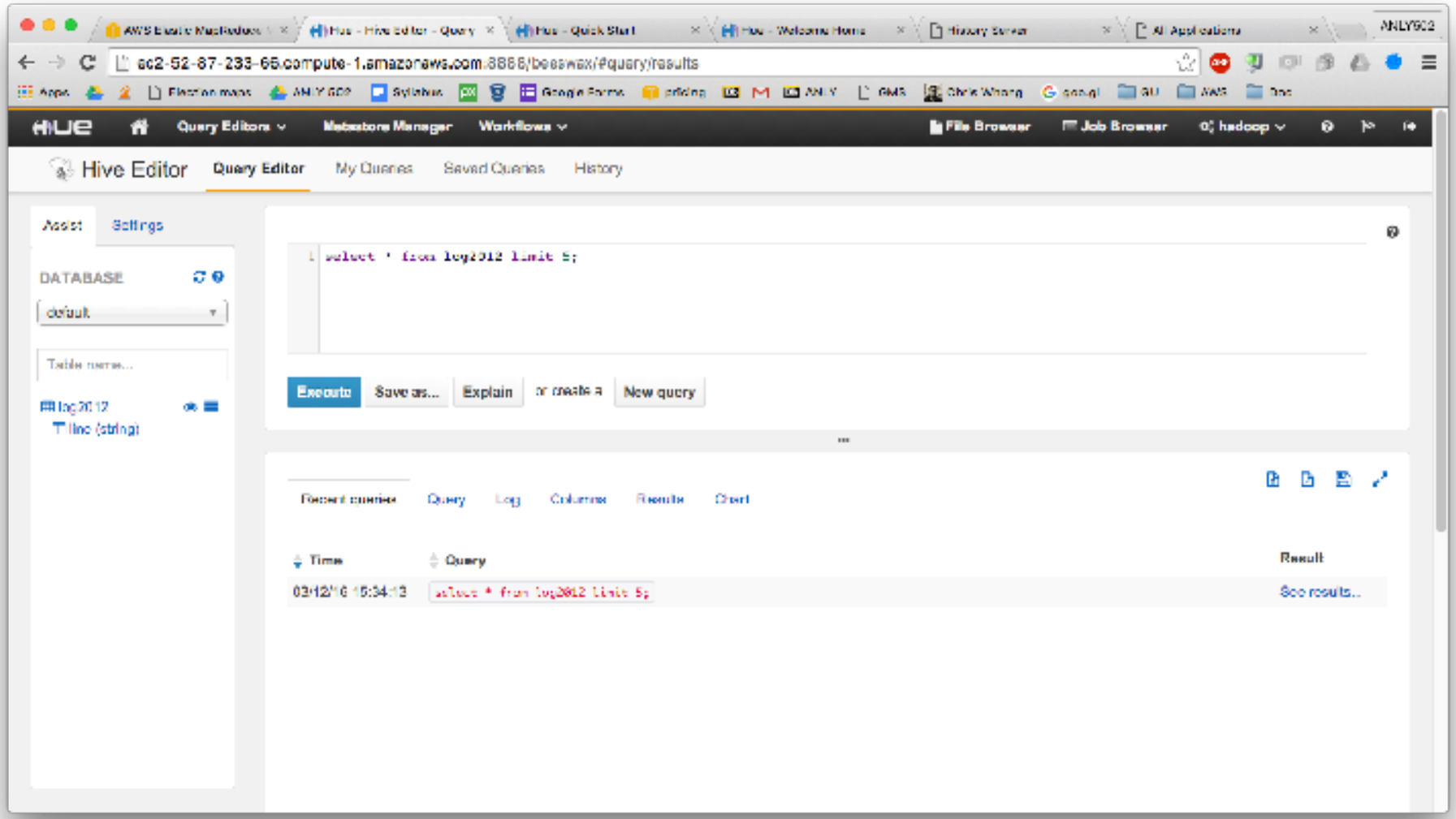
Cloudera VM: cloudera/cloudera

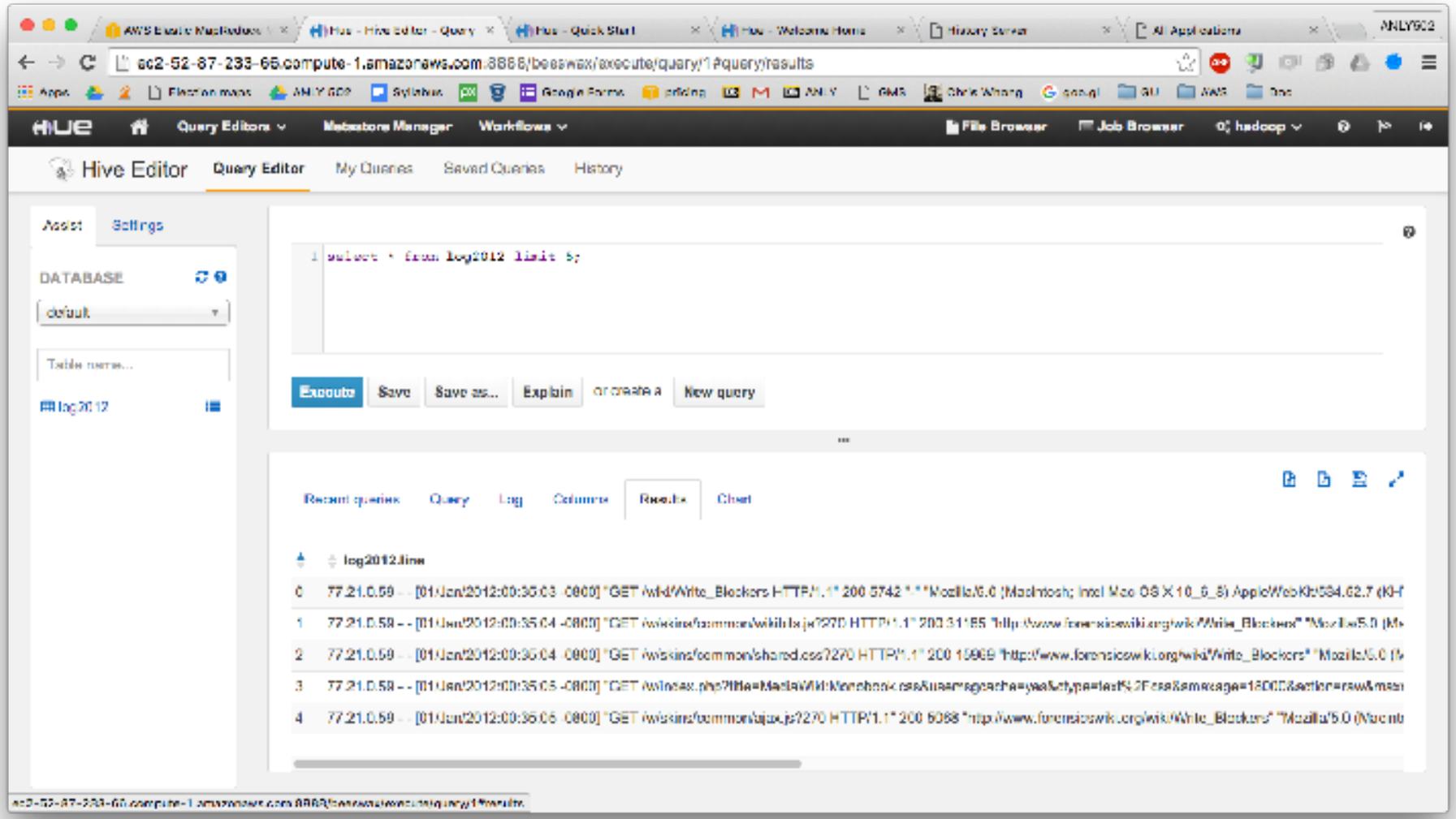
You can have multiple SQL connections at a

Create a table from the forensicswiki logfile

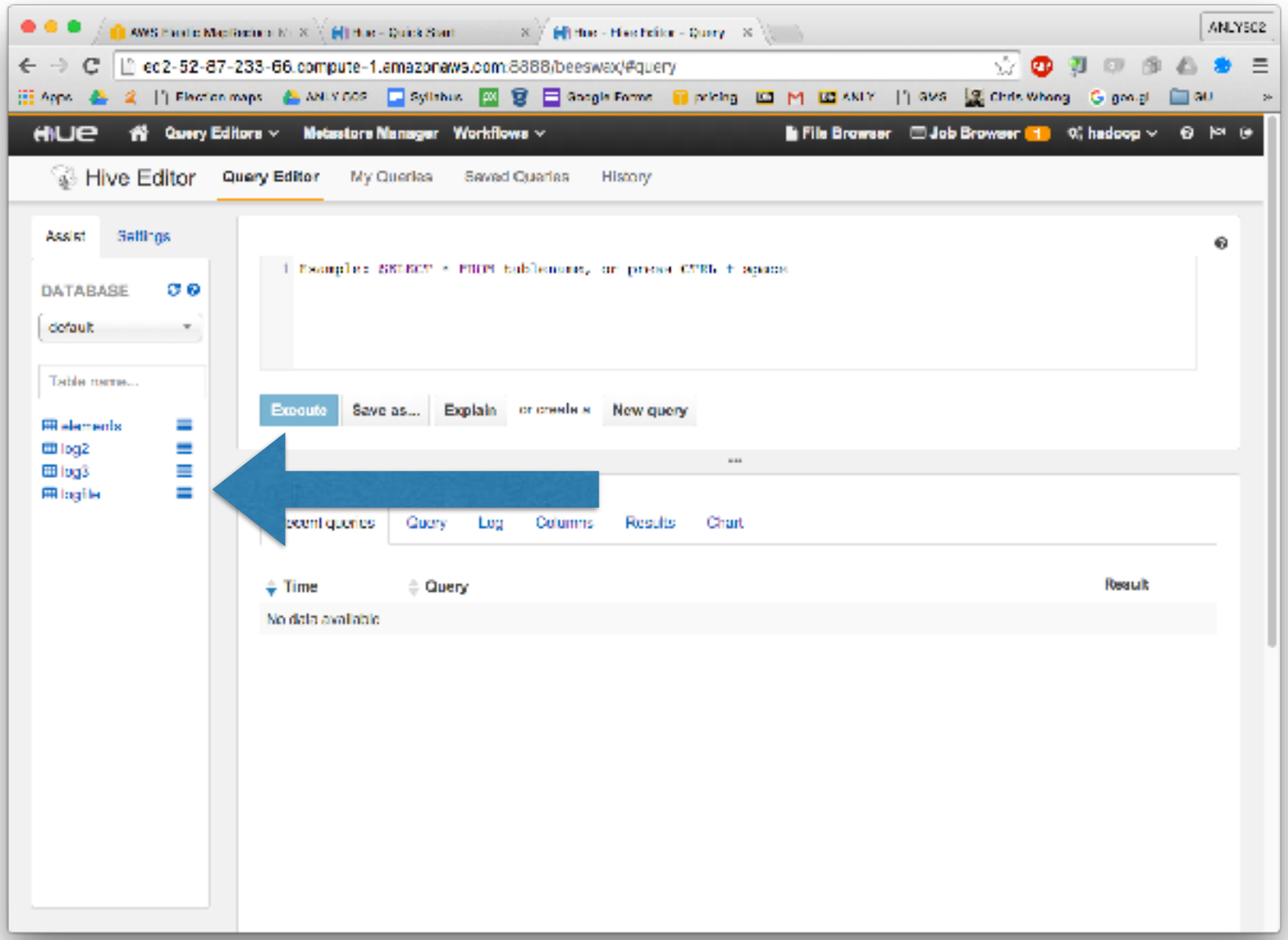
```
$ beeline
beeline> !connect jdbc:hive2://localhost:10000
...
0: jdbc:hive2://localhost:10000> create external table log2012 (line string) location 's3://
gu-anly502/ps03/forensicswiki/';
No rows affected (0.721 seconds)
0: jdbc:hive2://localhost:10000> select * from log2012 limit 3;
+--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------+--+
|
log2012.line
|
+--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------+--+
| 77.21.0.59 - - [01/Jan/2012:00:35:03 -0800] "GET /wiki/Write_Blockers HTTP/1.1" 200 5742 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.52.7 (KHTML, like Gecko)
Version/5.1.2 Safari/534.52.7"                                                              |
| 77.21.0.59 - - [01/Jan/2012:00:35:04 -0800] "GET /w/skins/common/wikibits.js?270 HTTP/1.1"
200 31165 "http://www.forensicswiki.org/wiki/Write_Blockers" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_6_8) AppleWebKit/534.52.7 (KHTML, like Gecko) Version/5.1.2 Safari/534.52.7"   |
| 77.21.0.59 - - [01/Jan/2012:00:35:04 -0800] "GET /w/skins/common/shared.css?270 HTTP/1.1"
200 15969 "http://www.forensicswiki.org/wiki/Write_Blockers" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_6_8) AppleWebKit/534.52.7 (KHTML, like Gecko) Version/5.1.2 Safari/534.52.7"    |
+--------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------+--+
3 rows selected (6.632 seconds)
0: jdbc:hive2://localhost:10000>
```

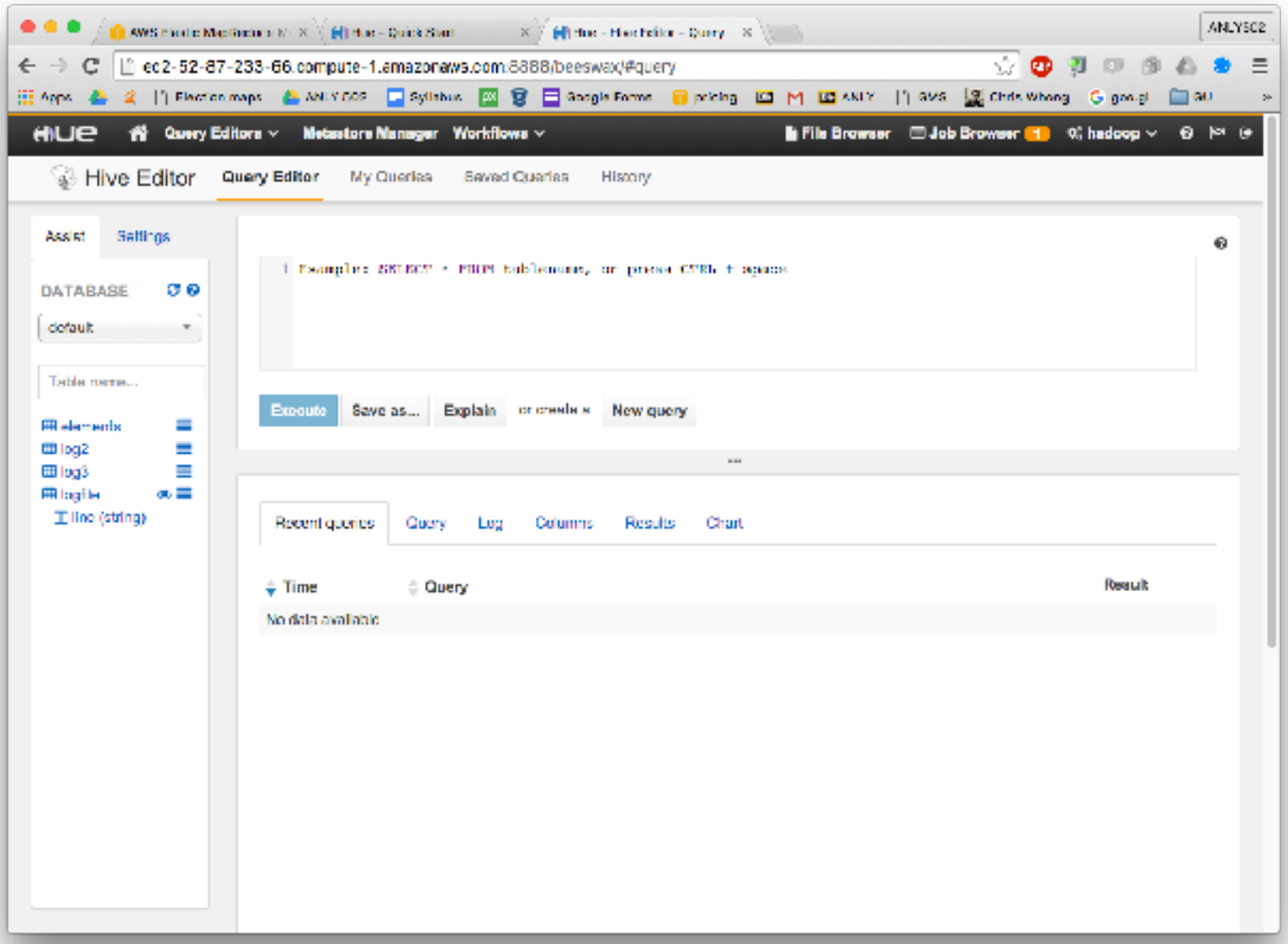
You can run this query from Hue

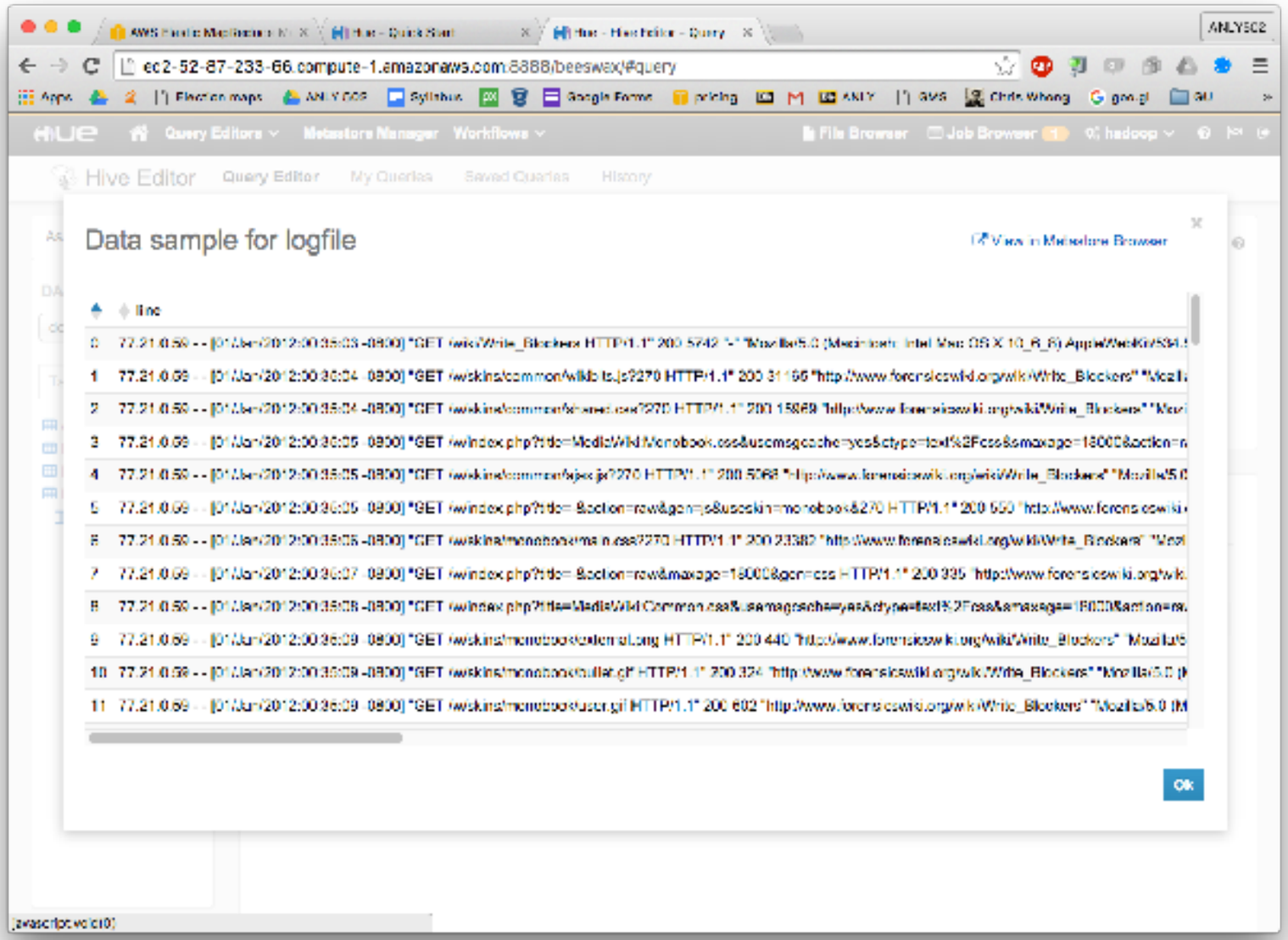



Hive Editor
Query Editor
My Queries
Saved Queries
History
Assist
Settings
DATABASE
default
log2012
select * from log2012 limit 5;
Execute
Save
Save as...
Explain
New query
Recent queries
Query
Log
Columns
Results
Chart
log2012.line

You can also browse the tables directly




ec2-52-87-233-66.compute-1.amazonaws.com:8888/beeswax/#query
Query Editors
Metastore Manager
Workflows
File Browser
Job Browser
hadoop
Hive Editor
Query Editor
My Queries
Saved Queries
History
Assist
Settings
DATABASE
default
Execute
Save as...
Explain
New query
Recent queries
Query
Log
Columns
Results
Chart
Time
Query
Result
No data available


ec2-52-87-233-66.compute-1.amazonaws.com:8888/beeswax/#query
Hive Editor
Query Editor
My Queries
Saved Queries
History
Data sample for logfile
View in Metastore Browser
line
Ok

Things to note:

Hive will load:

- HDFS "directories" e.g. hdfs:///user/hadoop/forensicswiki/"
- HDFS files e.g. hdfs:///user/hadoop/forensicswiki.2012.txt
- S3 "directories" e.g. s3://gu-anly502/ps03/forensicswiki/

But Hive will *not* load a single file for S3. (Appears to be a bug)

Hive will load directories *recursively* if you set two variables:

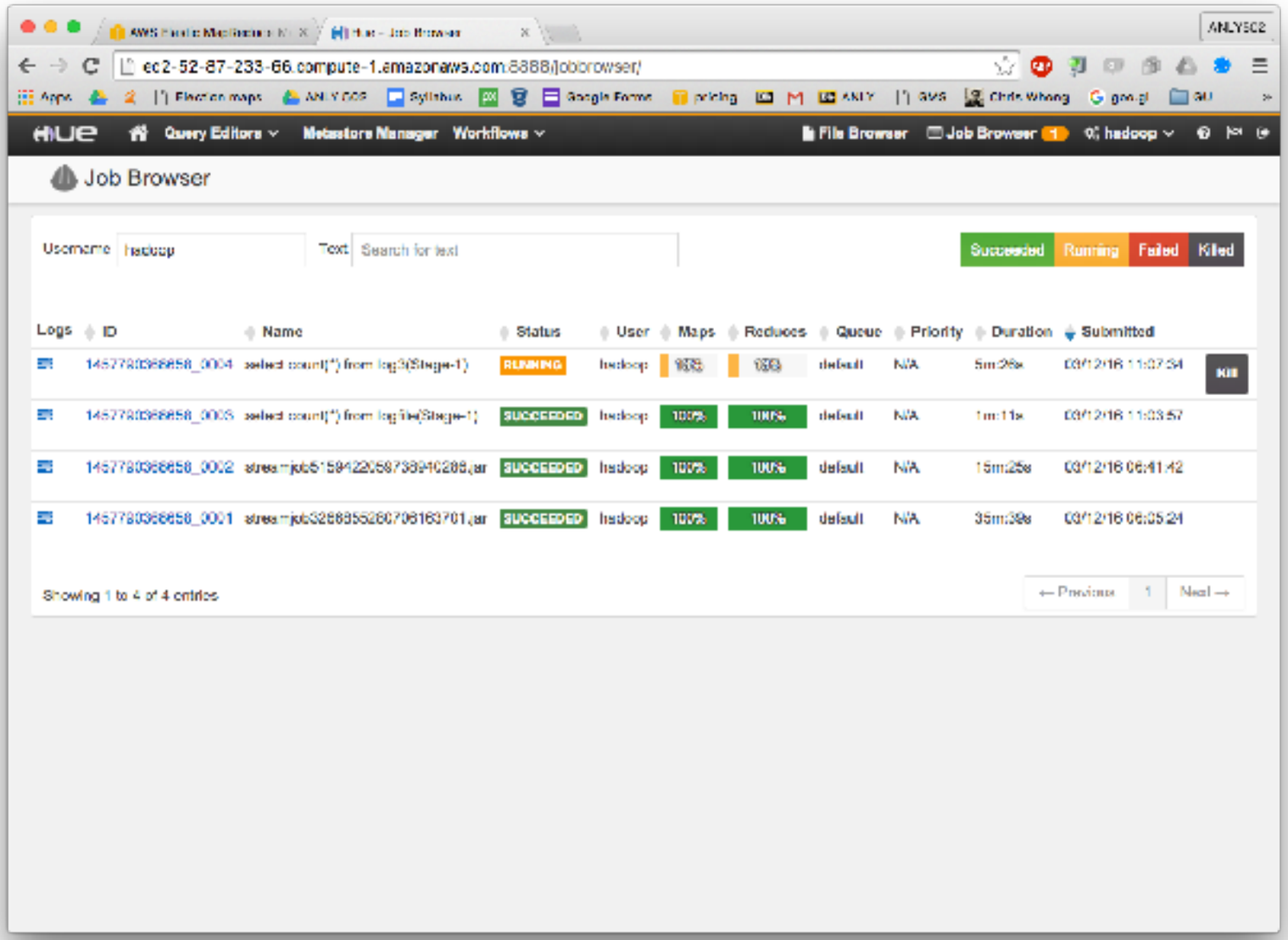
```
0: jdbc:hive2://localhost:10000> SET
mapred.input.dir.recursive=true;
0: jdbc:hive2://localhost:10000> SET
hive.mapred.supports.subdirectories=true;
```

Let's use Hive to count the number of lines in s3://gu-anly502/logs/forensicswiki.2012.txt

```
$ beeline
beeline> !connect jdbc:hive2://localhost:10000
...
0: jdbc:hive2://localhost:10000> create external table 2012 (line
string) location 's3://gu-anly502/logs/forensicswiki.2012.txt
No rows affected (0.721 seconds)
0: jdbc:hive2://localhost:10000> select count(*) from log2012;
...
```

```
0: jdbc:hive2://localhost:10000> select count(*) from log3;
INFO  : Number of reduce tasks determined at compile time: 1
INFO  : In order to change the average load for a reducer (in bytes):
INFO  :   set hive.exec.reducers.bytes.per.reducer=<number>
INFO  : In order to limit the maximum number of reducers:
INFO  :   set hive.exec.reducers.max=<number>
INFO  : In order to set a constant number of reducers:
INFO  :   set mapreduce.job.reduces=<number>
INFO  : number of splits:365
INFO  : Submitting tokens for job: job_1457790368658_0004
INFO  : The url to track the job: http://ip-172-31-44-166.ec2.internal:20888/proxy/
application_1457790368658_0004/
INFO  : Starting Job = job_1457790368658_0004, Tracking URL = http://ip-172-31-44-166.ec2.internal:20888/
proxy/application_1457790368658_0004/
INFO  : Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1457790368658_0004
INFO  : Hadoop job information for Stage-1: number of mappers: 365; number of reducers: 1
INFO  : 2016-03-12 19:07:40,540 Stage-1 map = 0%,  reduce = 0%
INFO  : 2016-03-12 19:08:05,631 Stage-1 map = 1%,  reduce = 0%, Cumulative CPU 13.79 sec
INFO  : 2016-03-12 19:08:12,479 Stage-1 map = 2%,  reduce = 0%, Cumulative CPU 41.36 sec
INFO  : 2016-03-12 19:08:36,114 Stage-1 map = 3%,  reduce = 0%, Cumulative CPU 76.79 sec
INFO  : 2016-03-12 19:08:52,007 Stage-1 map = 4%,  reduce = 0%, Cumulative CPU 90.08 sec
INFO  : 2016-03-12 19:09:02,345 Stage-1 map = 5%,  reduce = 0%, Cumulative CPU 118.71 sec
INFO  : 2016-03-12 19:09:31,063 Stage-1 map = 6%,  reduce = 0%, Cumulative CPU 154.39 sec
INFO  : 2016-03-12 19:09:34,425 Stage-1 map = 7%,  reduce = 0%, Cumulative CPU 169.24 sec
INFO  : 2016-03-12 19:09:59,468 Stage-1 map = 8%,  reduce = 0%, Cumulative CPU 197.72 sec
INFO  : 2016-03-12 19:10:21,173 Stage-1 map = 9%,  reduce = 0%, Cumulative CPU 225.28 sec
INFO  : 2016-03-12 19:10:28,107 Stage-1 map = 10%,  reduce = 0%, Cumulative CPU 246.17 sec
INFO  : 2016-03-12 19:10:50,493 Stage-1 map = 11%,  reduce = 0%, Cumulative CPU 274.83 sec
INFO  : 2016-03-12 19:10:57,153 Stage-1 map = 12%,  reduce = 0%, Cumulative CPU 297.25 sec
INFO  : 2016-03-12 19:11:18,439 Stage-1 map = 13%,  reduce = 0%, Cumulative CPU 326.12 sec
INFO  : 2016-03-12 19:11:43,724 Stage-1 map = 14%,  reduce = 0%, Cumulative CPU 354.71 sec
INFO  : 2016-03-12 19:11:49,329 Stage-1 map = 15%,  reduce = 0%, Cumulative CPU 377.45 sec
INFO  : 2016-03-12 19:12:14,303 Stage-1 map = 16%,  reduce = 0%, Cumulative CPU 407.04 sec
INFO  : 2016-03-12 19:12:28,107 Stage-1 map = 17%,  reduce = 0%, Cumulative CPU 439.22 sec
INFO  : 2016-03-12 19:12:48,660 Stage-1 map = 18%,  reduce = 0%, Cumulative CPU 473.09 sec
INFO  : 2016-03-12 19:13:12,521 Stage-1 map = 19%,  reduce = 0%, Cumulative CPU 502.13 sec
```

View running jobs…



Why is this taking so long?

```
INFO  : number of splits:365

INFO  : 2016-03-12 19:19:44 Stage-1 map = 43%,  reduce = 14%, Cumulative CPU 1168.69 sec
INFO  : 2016-03-12 19:20:0 map = 44%,  reduce = 14%, Cumulative CPU 1191.84 sec
INFO  : 2016-03-12 19:20:0 map = 44%,  reduce = 15%, Cumulative CPU 1200.04 sec
INFO  : 2016-03-12 19:20:21, Stage-1 map = 45%,  reduce = 15%, Cumulative CPU 1220.86 sec
INFO  : 2016-03-12 19:20:40,49 Stage-1 map = 46%,  reduce = 15%, Cumulative CPU 1254.78 sec
INFO  : 2016-03-12 19:20:49,226 Stage-1 map = 47%,  reduce = 15%, Cumulative CPU 1268.63 sec
INFO  : 2016-03-12 19:20:50,314 Stage-1 map = 47%,  reduce = 16%, Cumulative CPU 1268.67 sec
INFO  : 2016-03-12 19:21:08,951 Stage-1 map = 48%,  reduce = 16%, Cumulative CPU 1300.92 sec
INFO  : 2016-03-12 19:21:26,297 Stage-1 map = 49%,  reduce = 16%, Cumulative CPU 1331.44 sec
INFO  : 2016-03-12 19:21:40,455 Stage-1 map = 50%,  reduce = 16%, Cumulative PU 1351.64 sec
INFO  : 2016-03-12 19:21:42,669 Stage-1 map = 50%,  reduce = 17%, Cumulati PU 1351.67 sec
INFO  : 2016-03-12 19:21:57,839 Stage-1 map = 51%,  reduce = 17%, Cumulat PU 1379.21 sec
INFO  : 2016-03-12 19:22:14,363 Stage-1 map = 52%,  reduce = 17%, Cumul 5 sec
INFO  : 2016-03-12 19:22:31,819 Stage-1 map = 53%,  reduce = 17%, Cum sec
INFO  : 2016-03-12 19:22:33,963 Stage-1 map = 53%,  reduce = 18%, Cumu sec
INFO  : 2016-03-12 19:22:50,276 Stage-1 map = 54%,  reduce = 18%, Cumula PU 1458.65 sec
INFO  : 2016-03-12 19:22:57,997 Stage-1 map = 55%,  reduce = 18%, Cumulati PU 1478.61 sec
INFO  : 2016-03-12 19:23:15,373 Stage-1 map = 56%,  reduce = 18%, Cumulative PU 1504.5 sec
```

365 splits?

Lots of mapping and reducing?

Why is this taking so long? "top"

```
top - 19:26:25 up  5:43,  3 users,  load average: 9.84, 9.63, 7.32
Tasks: 172 total,   1 running, 169 sleeping,   2 stopped,   0 zombie
Cpu(s): 88.2%us, 11.4%sy,  0.0%ni,  0.4%id,  0.0%wa,  0.[illegible]%hi,  0.0%si,  0.0%st
Mem:  15407244k total, 14758868k used,   648376k free,   70120k buffers
Swap:        0k total,        0k used,        0k f[illegible]949444k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %ME[illegible]D
17588 yarn      20   0 1826m 151m  27m S 111.7  1.0   [illegible]4.53 java
17465 yarn      20   0 1849m 236m  27m S 96.1  1.6   0.[illegible].56 java
17423 yarn      20   0 1843m 274m  27m S 75.9  1.8   0:12.75 java
17350 yarn      20   0 1853m 493m  27m S 71.9  3.3   0:15.66 ja[illegible]
 4477 yarn      20   0 3422m 466m  28m S  8.0  3.1   3:02.65 [illegible]
 5363 hadoop    20   0 6333m 441m  14m S  6.0  2.9   2:28.49[illegible]
 2315 hadoop    20   0 3496m 171m  14m S  4.3  1.1   1:57.1[illegible]
 4906 yarn      20   0 2918m 256m  28m S  4.3  1.7   5:30[illegible]
 8980 hive      20   0 1653m 271m  28m S  4.0  1.8   1:[illegible]
10221 oozie     20   0 3511m 438m  17m S  3.3  2.9   8[illegible]
12759 hue       20   0 1088m 117m  12m S  1.3  0.8   0:2[illegible]
24364 yarn      20   0 3219m 604m  27m S  1.3  4.0   0:47.[illegible]
 3810 hdfs      20   0 2522m 638m  27m S  0.7  4.2   1:49.9[illegible]
 3938 hdfs      20   0 1507m 275m  27m S  0.7  1.8   1:33.83 [illegible]
 5711 yarn      20   0 3054m 242m  27m S  0.7  1.6   0:15.50 ja[illegible]
    3 root      20   0     0    0    0 S  0.3  0.0   0:00.95 kso[illegible]tirqd/0
 1580 root      20   0  110m 3084 2872 S  0.3  0.0   0:00.08 dump-instance-s
 3446 kms       20   0 5736m 164m  14m S  0.3  1.1   0:25.42 java
12730 root      20   0  357m  15m 4056 S  0.3  0.1   0:05.88 python2.6
17492 hadoop    20   0 15292 2392 2016 R  0.3  0.0   0:00.03 top
17579 yarn      20   0  110m 2820 2668 S  0.3  0.0   0:00.01 bash
17646 hadoop    20   0 15288 2192 1896 S  0.3  0.0   0:00.01 top
20964 hadoop    20   0 1611m 117m  27m S  0.3  0.8   0:02.83 java
    1 root      20   0 19776 2744 2264 S  0.0  0.0   0:01.44 init
```

Load of 9?

4 cores processing
15 million lines?

We eventually get the result…

```
INFO  : 2016-03-12 19:35:43,436 Stage-1 map = 100%,  reduce = 100%,
Cumulative CPU 2672.63 sec
INFO  : MapReduce Total cumulative CPU time: 44 minutes 32 seconds
630 msec
INFO  : Ended Job = job_1457790368658_0004
+-----------+--+
|    _c0    |
+-----------+--+
| 15949554  |
+-----------+--+
1 row selected (1700.456 seconds)
0: jdbc:hive2://localhost:10000>
```

There are many ways to load data into a table

CREATE EXTERNAL TABLE name LOCATION ;

—*optional: FIELDS TERMINATED BY ','*

—*optional: LINES TERMINATED BY '\n'*

CREATE EXTERNAL TABLE *table...*;

LOAD DATA INPATH '*hdfs_path*' INTO TABLE *table*;

CREATE EXTERNAL TABLE ...;

ALTER TABLE *name* SET LOCATION '*hdfs_or_s3_path_of_directory*'

CREATE [TEMPORARY|EXTERNAL] TABLE *name*

INSERT INTO TABLE *name* SELECT *expression1,expression2,..* FROM *othertable*

CREATE TABLE...

INSERT ... VALUES ...

Create Table DDL and Alter Table DDL

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS]
[db_name.]table_name    -- (Note: TEMPORARY available in Hive 0.14.0
and later)
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|
DESC], ...)] INTO num_buckets BUCKETS]
  [SKEWED BY (col_name, col_name, ...)                  -- (Note:
Available in Hive 0.10.0 and later)]
     ON ((col_value, col_value, ...), (col_value, col_value, ...), ...)
     [STORED AS DIRECTORIES]
  [
   [ROW FORMAT row_format]
   [STORED AS file_format]
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES
(...)]  -- (Note: Available in Hive 0.6.0 and later)
  ]
  [LOCATION hdfs_path]
  [TBLPROPERTIES (property_name=property_value, ...)]   -- (Note:
Available in Hive 0.6.0 and later)
  [AS select_statement];   -- (Note: Available in Hive 0.5.0 and later;
not supported for external tables)
```

—*https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL*

Partitions split a single table into different sections

For example, we can partition forensicswiki logfiles by YEAR and MONTH.

e.g.:

```
$ aws s3 ls s3://gu-anly502/ps05/
                           PRE forensicswiki/
$ aws s3 ls s3://gu-anly502/ps05/forensicswiki/
                           PRE 2012/
$ aws s3 ls s3://gu-anly502/ps05/forensicswiki/2012/
                           PRE 01/
                           PRE 02/
                           PRE 03/
                           PRE 04/
                           PRE 05/
                           PRE 06/
                           PRE 07/
                           PRE 08/
                           PRE 09/
                           PRE 10/
                           PRE 11/
                           PRE 12/
$ aws s3 ls s3://gu-anly502/ps05/forensicswiki/2012/01/
2016-03-12 14:23:30  402236086 access.log.2012-01
$
```

When a query is issued Hive will only consider the necessary partitions.

Creating partitions:
Create the table; create the partitions; add data to the partitions.

```
0: jdbc:hive2://localhost:10000> create table plogs (line string) partitioned by
(year int, month int);
No rows affected (0.032 seconds)
0: jdbc:hive2://localhost:10000> alter table plogs add partition (year=2012,month=1);
No rows affected (0.082 seconds)
0: jdbc:hive2://localhost:10000> alter table plogs partition (year=2012,month=1) set
location 's3://gu-anly502/ps05/forensicswiki/2012/01/';
No rows affected (0.386 seconds)
0: jdbc:hive2://localhost:10000> alter table plogs add partition (year=2012,month=2);
No rows affected (0.052 seconds)
0: jdbc:hive2://localhost:10000> alter table plogs partition (year=2012,month=2) set
location 's3://gu-anly502/ps05/forensicswiki/2012/02/';
No rows affected (0.36 seconds)
0: jdbc:hive2://localhost:10000> alter table plogs add partition (year=2012,month=3);
No rows affected (0.057 seconds)
0: jdbc:hive2://localhost:10000> alter table plogs partition (year=2012,month=3) set
location 's3://gu-anly502/ps05/forensicswiki/2012/03/';
No rows affected (0.356 seconds)
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> select year,month,substr(line,0,30) from plogs limit
3;
+-------+--------+-----------------------------------+--+
| year  | month  |                _c2                |
+-------+--------+-----------------------------------+--+
| 2012  | 1      | 77.21.0.59 - - [01/Jan/2012:00    |
| 2012  | 1      | 77.21.0.59 - - [01/Jan/2012:00    |
| 2012  | 1      | 77.21.0.59 - - [01/Jan/2012:00    |
+-------+--------+-----------------------------------+--+
3 rows selected (0.332 seconds)
0: jdbc:hive2://localhost:10000>
```

Parsing Apache logs in Hive
1. Create a table of extracted fields; 2. Reparse the fields as necessary

```
DROP TABLE IF EXISTS apache_common_log;
CREATE EXTERNAL TABLE apache_common_log (
  host STRING,
  identity STRING,
  user STRING,
  rawdatetime STRING,
  request STRING,
  status STRING,
  size STRING,
  refer STRING,
  agent STRING
  )
ROW FORMAT SERDE 'org.apache.hadoop.hive
WITH SERDEPROPERTIES (
  "input.regex" = "([^ ]*) ([^ ]*) ([^ ]                                  [^\"]*\") (-|[0-9]*) (-|
[0-9]*) \"([^\"]*)\" \"([^\"]*)\".*",
  "output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s"
)
STORED AS TEXTFILE
LOCATION 's3://gu-anly502/ps05/forensicswiki/2012/12/';

create temporary table clean_logs (
  host string,
  datetime string,
  url string
);

insert overwrite table clean_logs
  select host,
         from_unixtime(unix_timestamp(rawdatetime, "[dd/
         regexp_extract(request,".[^ ]* ([^ ]*)")

  from apache_common_log;


SELECT * FROM clean_logs limit 3;
```

Uses Hive's Regular Expression Serializer/ Deserializer

Creates a temporary table clean_logs with the desired fields

reformats the time, extracts the URL from the request.

Running the sample code:

```
$ beeline -u jdbc:hive2://localhost:10000 -n hadoop -p hadoop -f apache-demo.sql
scan complete in 8ms
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 1.0.0-amzn-2)
Driver: Hive JDBC (version 1.0.0-amzn-2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000> DROP TABLE IF EXISTS apache_common_log;
No rows affected (0.202 seconds)
0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE apache_common_log (
. . . . . .>   host STRING,
. . . . . .>   identity STRING,
. . . . . .>   user STRING,
...
. . . . . .>   from apache_common_log;
INFO  : Number of reduce tasks is set to 0 since there's no reduce operator
INFO  : number of splits:2
INFO  : Submitting tokens for job: job_1457790368658_0014
INFO  : The url to track the job: http://ip-172-31-44-166.ec2.internal:20888/proxy/application_1457790368658_0014/
INFO  : Starting Job = job_1457790368658_0014, Tracking URL = http://ip-172-31-44-166.ec2.internal:20888/proxy/
application_1457790368658_0014/
INFO  : Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1457790368658_0014
INFO  : Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 0
INFO  : 2016-03-13 04:12:22,934 Stage-1 map = 0%,  reduce = 0%
...
INFO  : Stage-5 is filtered out by condition resolver.
INFO  : Moving data to: hdfs://ip-172-31-44-166.ec2.internal:8020/tmp/hive/hadoop/5216aa41-8209-48fc-a9ab-f02c2329007c/
hive_2016-03-13_04-12-15_999_5818612435067683317-3/-ext-10000 from hdfs://ip-172-31-44-166.ec2.internal:8020/tmp/hive/
hadoop/5216aa41-8209-48fc-a9ab-f02c2329007c/hive_2016-03-13_04-12-15_999_5818612435067683317-3/-ext-10002
INFO  : Loading data to table default.clean_logs from hdfs://ip-172-31-44-166.ec2.internal:8020/tmp/hive/hadoop/
5216aa41-8209-48fc-a9ab-f02c2329007c/hive_2016-03-13_04-12-15_999_5818612435067683317-3/-ext-10000
INFO  : Table default.clean_logs stats: [numFiles=2, numRows=1397115, totalSize=149134460, rawDataSize=147737345]
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000>
0: jdbc:hive2://localhost:10000> -- SELECT * FROM apache_common_log LIMIT 5;
0: jdbc:hive2://localhost:10000> SELECT * FROM clean_logs limit 3;
+------------------+----------------------+----------------------------------+--+
| clean_logs.host  | clean_logs.datetime  |          clean_logs.url          |
+------------------+----------------------+----------------------------------+--+
| 72.47.85.98      | 2012-12-01 08:19:03  | /                                |
| 69.163.129.236   | 2012-12-01 08:19:04  | /w/extensions/BibTex/bibtex.css  |
| 69.163.129.236   | 2012-12-01 08:19:04  | /w/extensions/BibTex/bibtex.js   |
+------------------+----------------------+----------------------------------+--+
3 rows selected (0.09 seconds)
0: jdbc:hive2://localhost:10000>
```

Other goodies in Hive

Math Functions

Array functions

Map & Reduce functions

Date time functions

XML (xpath) & JSON

String Functions

Statistics, Aggragate

expode() — Takes an array and maps to separate rows

Parsing

MD5, SHA1, CRC, Encryption

—*https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF*

https://www.pexels.com/photo/people-apple-iphone-writing-154/

Homework and L08 Preview

Start thinking about your final projects!

Mon Mar 20 — L08: Hive & Pig (probably will change)

Tue Mar 22 — Final Project Individual Proposals

Mon Mar 27 — Scalable Machine Learning with Spark

Tue Mar 28 — Final Project Group Proposals

Mon April 3 — L10: Streaming Databases & Graph Databases

Mon April 10 — No class — Passover & Italy

Mon April 10 – Fri April 14 — Final Project Online Clinic (it's graded!)

Mon April 17 — No class Easter Break

Mon April 24 — L11: NoSQL

Mon May 1 — L12: Final Project Presentations (last class!!!)

Wed May 10 — Final Projects Due

Mon May 15 — Grades due for graduating students (anybody graduating?)

Reading!

Read this Apache Hive Documentation:

- https://cwiki.apache.org/confluence/display/Hive/Tutorial

Skim the API

- https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL
- https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients

The Hive documentation is on a wiki.

Recommended blog posts:

- https://blog.cloudera.com/blog/2014/02/migrating-from-hive-cli-to-beeline-a-primer/
- https://www.brentozar.com/archive/2013/03/introduction-to-hive-partitioning/
- http://hortonworks.com/hadoop-tutorial/how-to-process-data-with-apache-hive/
- https://www.qubole.com/blog/big-data/5-tips-for-efficient-hive-queries/

Readings

A brief history of databases, Stephen Fortune

- http://avant.org/project/history-of-databases/