L05: Databases at Scale

ANLY 502: Massive Data Fundamentals

Simson Garfinkel & Ghaleb Abdulla

February 22, 2016

Outline for today's lesson

Status Check:

- AWS EMR Tricks
- PS03 In Progress

Student Presentations:

Guest Presentation — Donald Miner

Pig

Future Deliverables:

- PS04 will be assigned on Friday! Now due March 18
- PS05 will is being revised — Now due April 1st (Something with text or image processing)
- Start thinking about final projects — Proposals due March 22nd

AWS EMR Tricks

Make your time more efficient

EMR Bootstrap code — s3://gu-anly502/bootstrap.sh

- Bootstrap specified in EMR creation sequence.

EMR startup code — create your own! Mine is at s3://gu-anly502/startup.sh

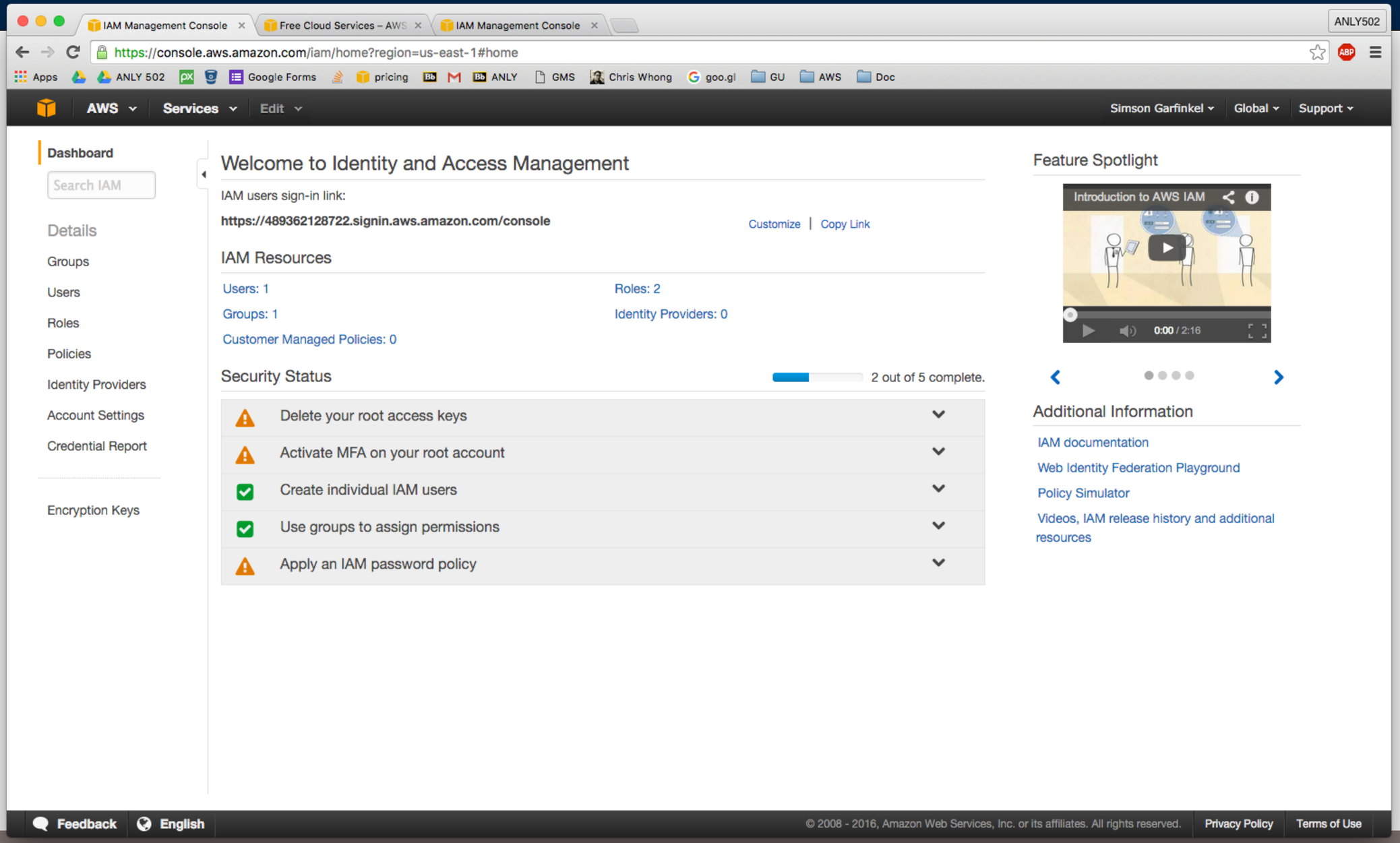
- I run it with:

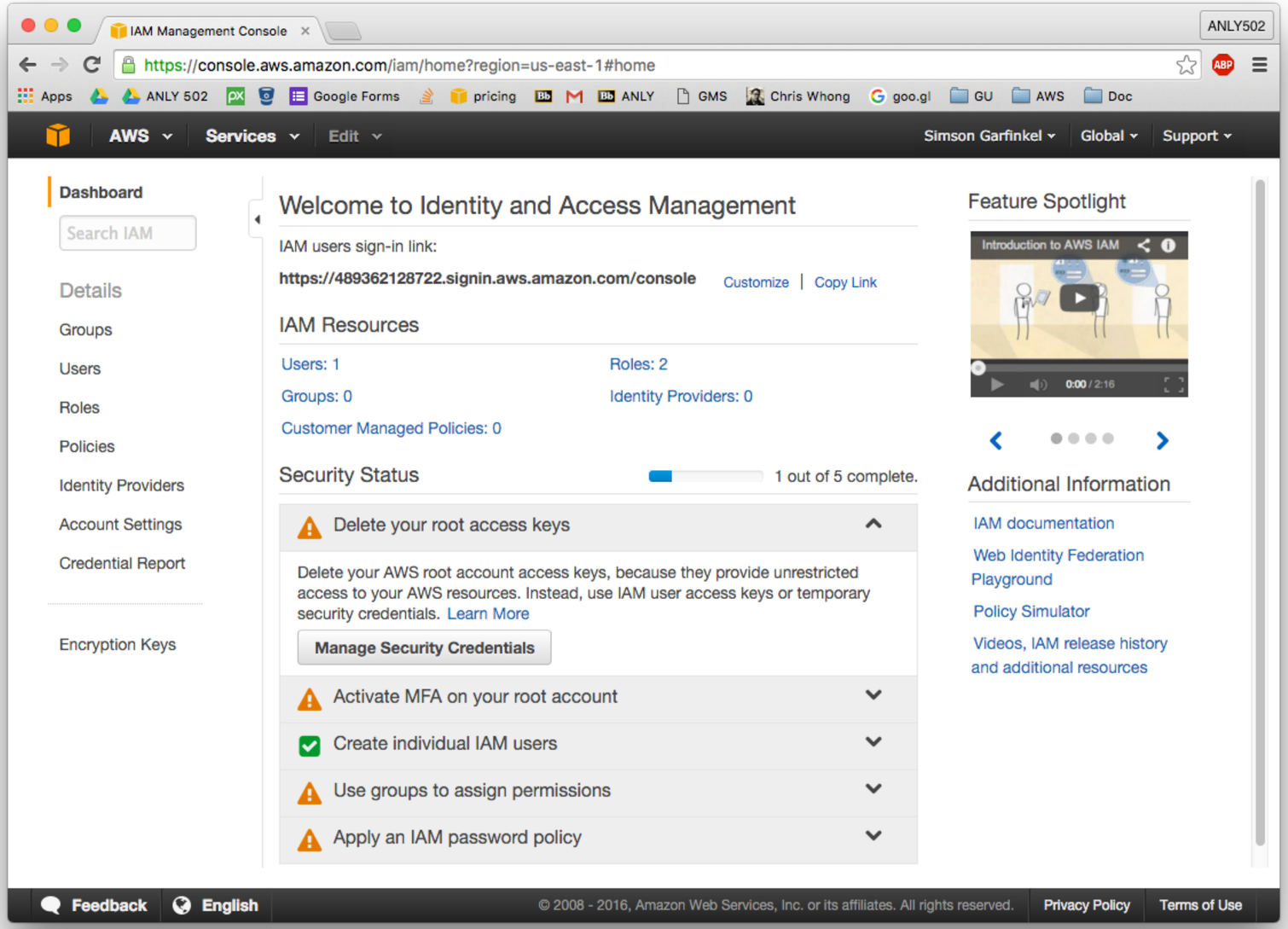
```
$ aws s3 cp s3://gu-anly502/startup.sh - | bash
```

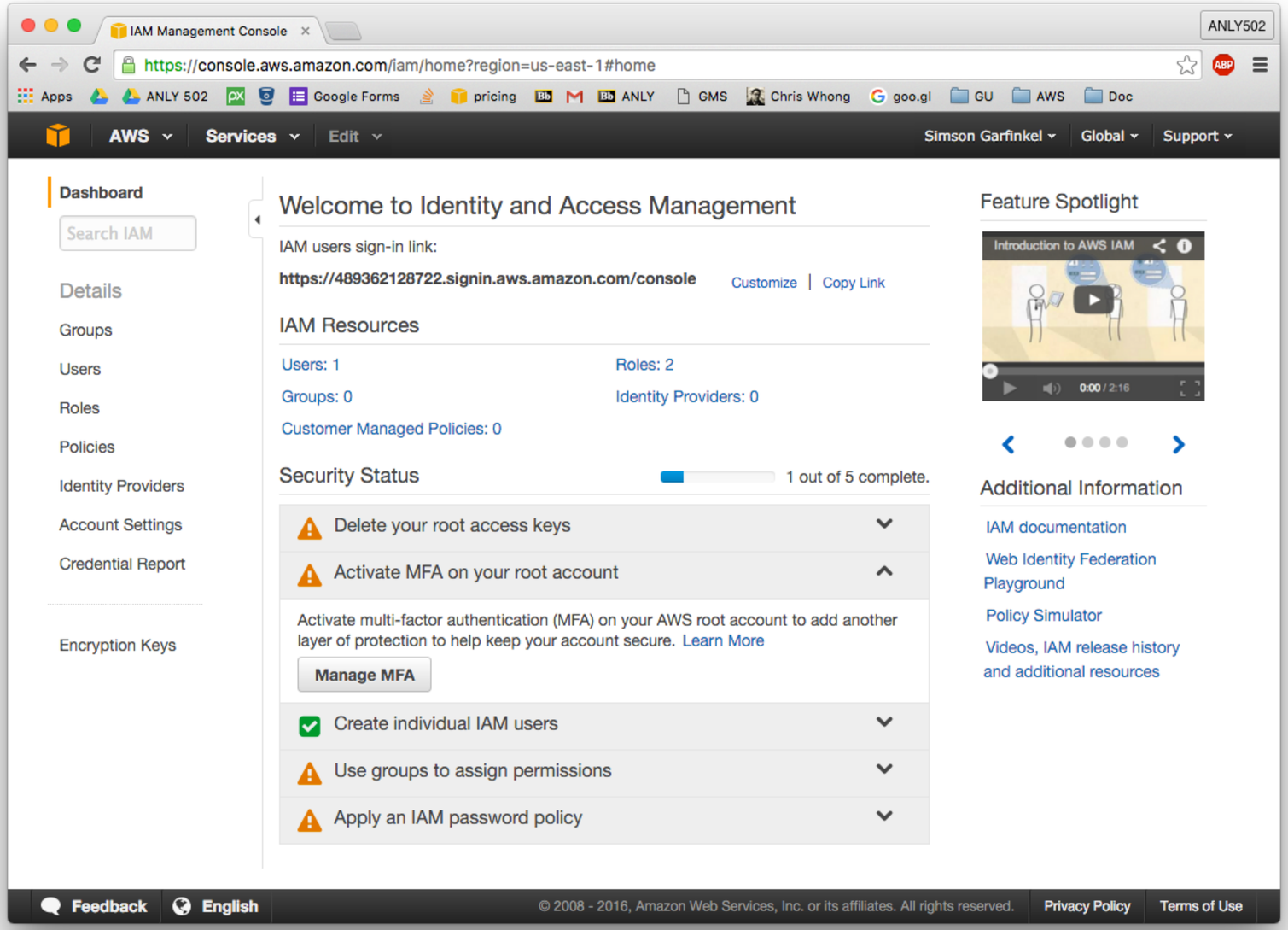
Tired of typing your AWS authorization credentials? Try IAM roles.

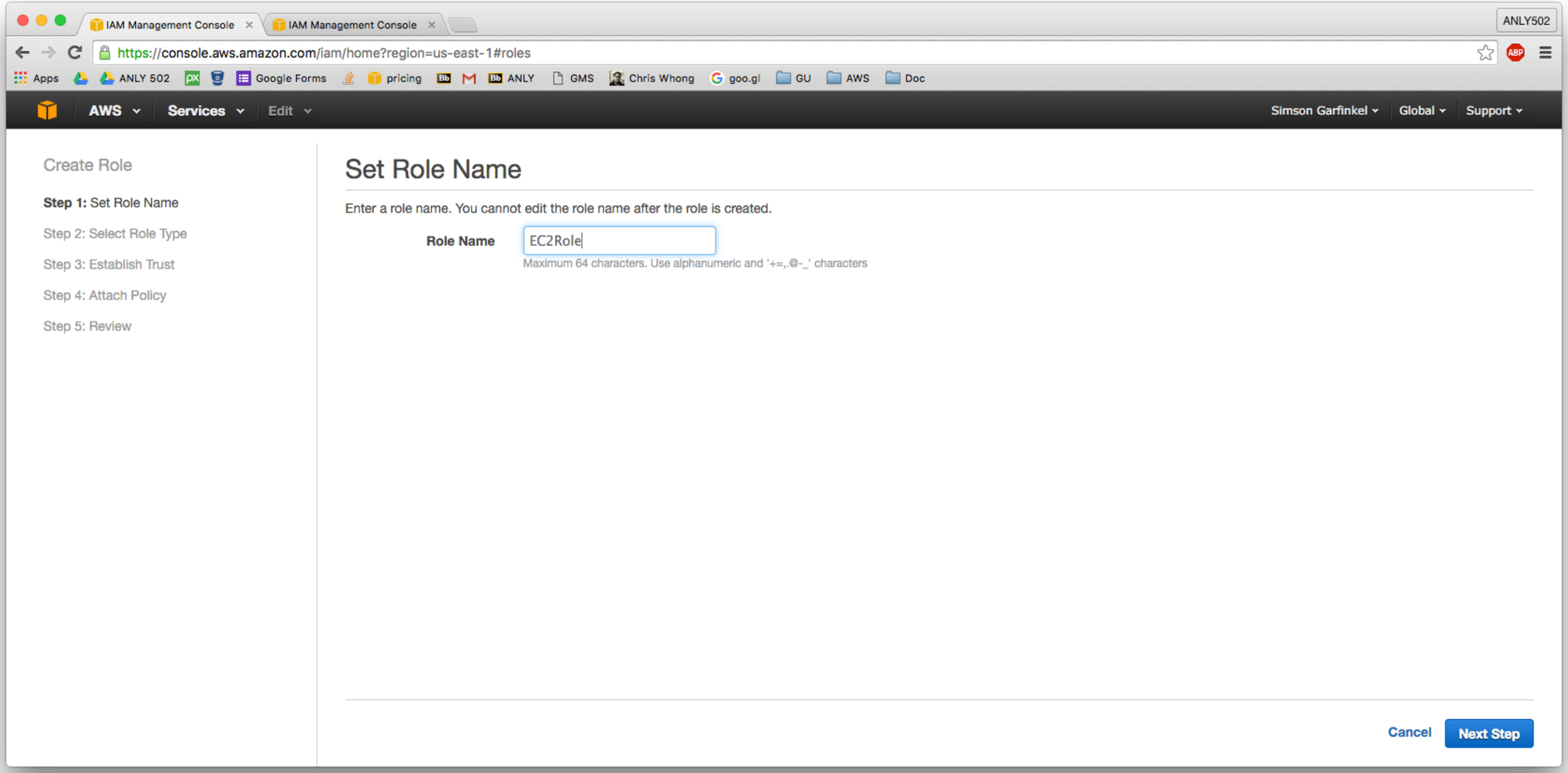
```
[ec2-user@ip-172-31-58-163 ~]$ aws s3 ls s3://gu-anly502/ps03/forensicswiki | head
Unable to locate credentials. You can configure credentials by running "aws configure".
```

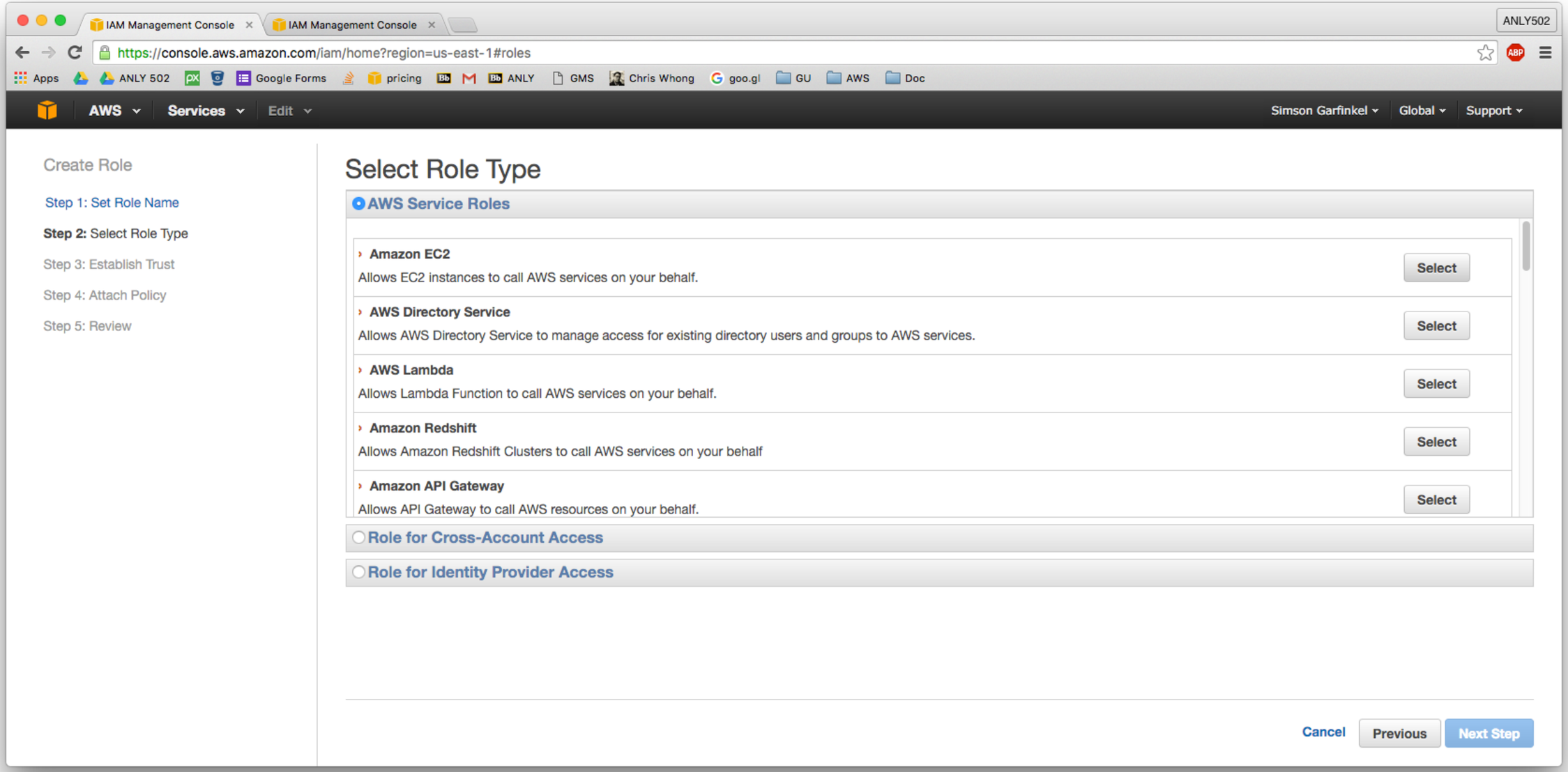
IAM roles let you "burn in" account authentication.

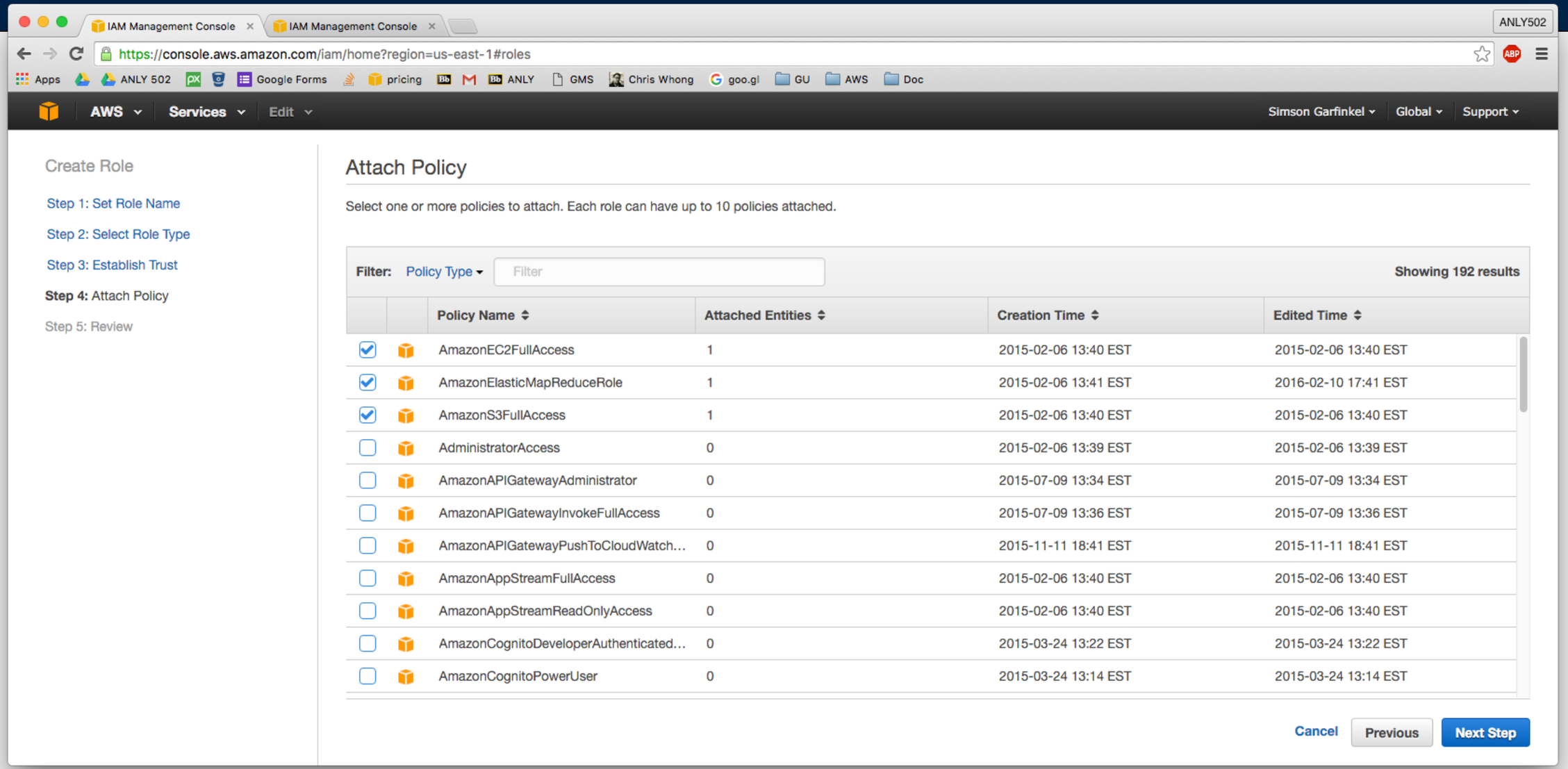

IAM Management Console
Free Cloud Services – AWS
IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#home
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
Welcome to Identity and Access Management
IAM users sign-in link:
https://489362128722.signin.aws.amazon.com/console
Customize | Copy Link
IAM Resources
Users: 1
Roles: 2
Groups: 1
Identity Providers: 0
Customer Managed Policies: 0
Security Status
2 out of 5 complete.
Delete your root access keys
Activate MFA on your root account
Create individual IAM users
Use groups to assign permissions
Apply an IAM password policy
Feature Spotlight
Introduction to AWS IAM
0:00 / 2:16
Additional Information
IAM documentation
Web Identity Federation Playground
Policy Simulator
Videos, IAM release history and additional resources
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

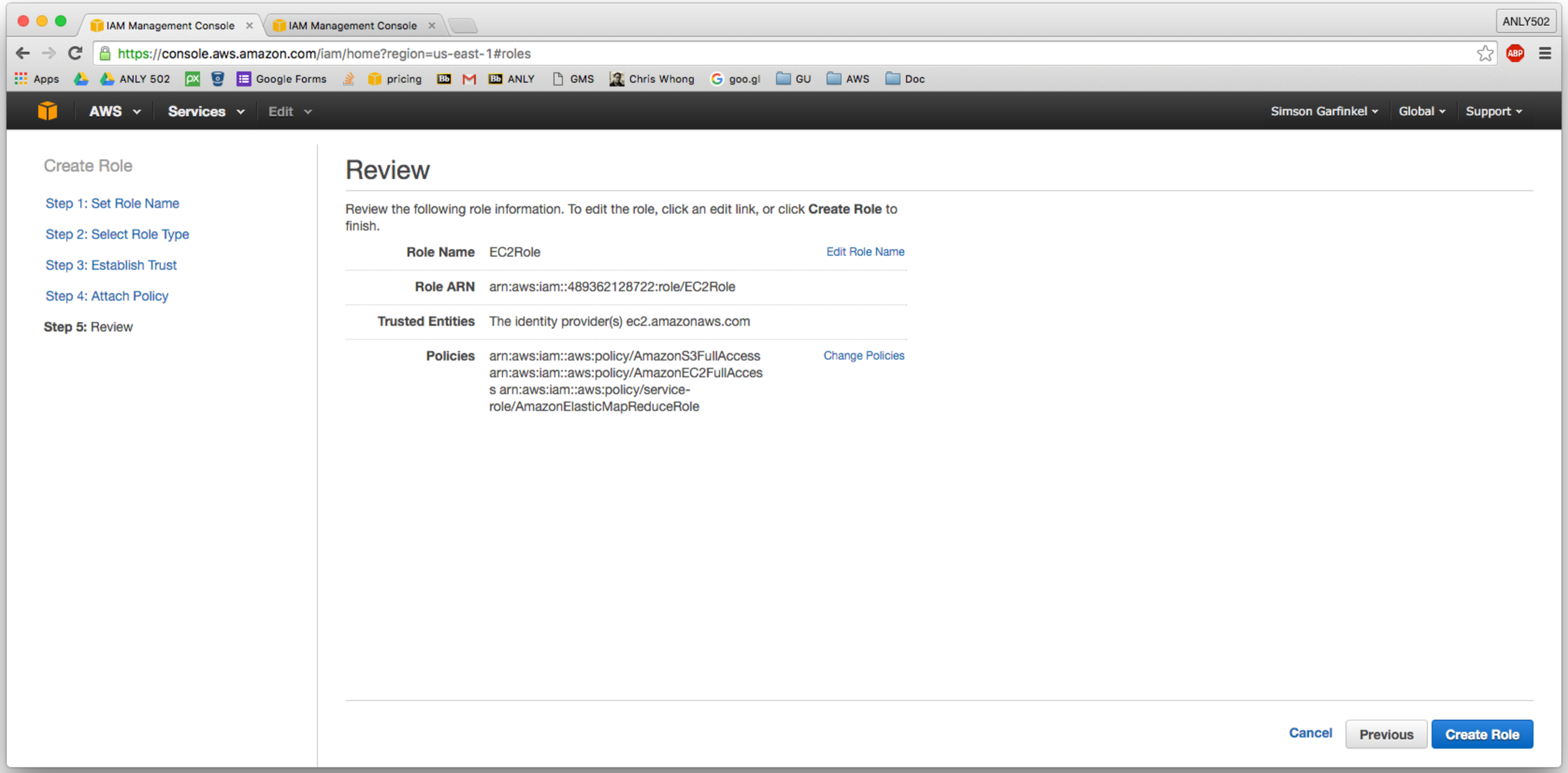

IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#home
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
Welcome to Identity and Access Management
IAM users sign-in link:
https://489362128722.signin.aws.amazon.com/console
Customize
Copy Link
IAM Resources
Users: 1
Roles: 2
Groups: 0
Identity Providers: 0
Customer Managed Policies: 0
Security Status
1 out of 5 complete.
Delete your root access keys
Delete your AWS root account access keys, because they provide unrestricted access to your AWS resources. Instead, use IAM user access keys or temporary security credentials. Learn More
Manage Security Credentials
Activate MFA on your root account
Create individual IAM users
Use groups to assign permissions
Apply an IAM password policy
Feature Spotlight
Introduction to AWS IAM
0:00 / 2:16
Additional Information
IAM documentation
Web Identity Federation Playground
Policy Simulator
Videos, IAM release history and additional resources
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#home
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Dashboard
Search IAM
Details
Groups
Users
Roles
Policies
Identity Providers
Account Settings
Credential Report
Encryption Keys
Welcome to Identity and Access Management
IAM users sign-in link:
https://489362128722.signin.aws.amazon.com/console
Customize | Copy Link
IAM Resources
Users: 1
Roles: 2
Groups: 0
Identity Providers: 0
Customer Managed Policies: 0
Security Status
1 out of 5 complete.
Delete your root access keys
Activate MFA on your root account
Activate multi-factor authentication (MFA) on your AWS root account to add another layer of protection to help keep your account secure. Learn More
Manage MFA
Create individual IAM users
Use groups to assign permissions
Apply an IAM password policy
Feature Spotlight
Introduction to AWS IAM
0:00 / 2:16
Additional Information
IAM documentation
Web Identity Federation Playground
Policy Simulator
Videos, IAM release history and additional resources
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

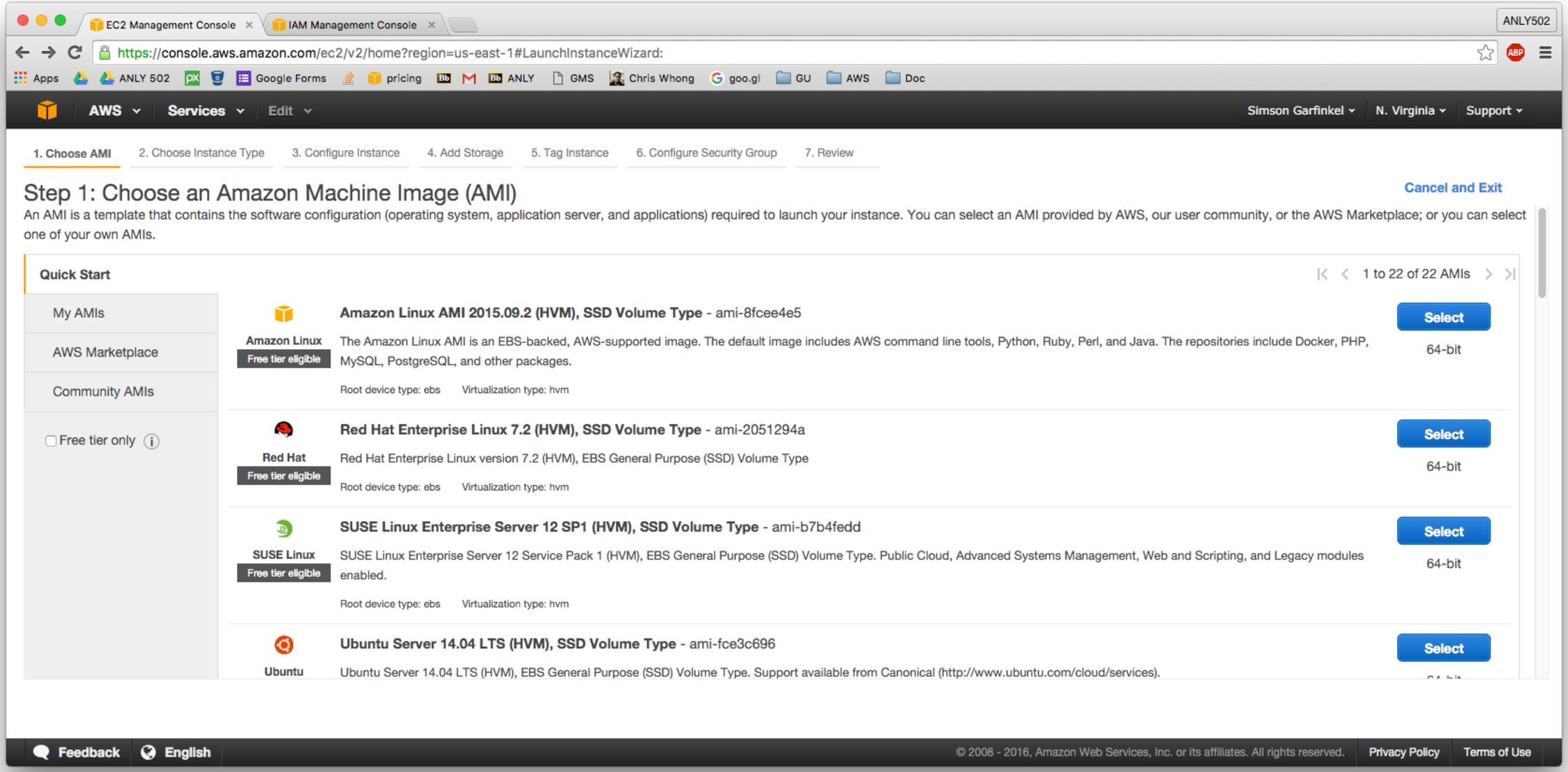

IAM Management Console
IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#roles
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Create Role
Step 1: Set Role Name
Step 2: Select Role Type
Step 3: Establish Trust
Step 4: Attach Policy
Step 5: Review
Set Role Name
Enter a role name. You cannot edit the role name after the role is created.
Role Name
EC2Role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters
Cancel
Next Step


IAM Management Console
IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#roles
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Create Role
Step 1: Set Role Name
Step 2: Select Role Type
Step 3: Establish Trust
Step 4: Attach Policy
Step 5: Review
Select Role Type
AWS Service Roles
Amazon EC2
Allows EC2 instances to call AWS services on your behalf.
Select
AWS Directory Service
Allows AWS Directory Service to manage access for existing directory users and groups to AWS services.
Select
AWS Lambda
Allows Lambda Function to call AWS services on your behalf.
Select
Amazon Redshift
Allows Amazon Redshift Clusters to call AWS services on your behalf
Select
Amazon API Gateway
Allows API Gateway to call AWS resources on your behalf.
Select
Role for Cross-Account Access
Role for Identity Provider Access
Cancel
Previous
Next Step

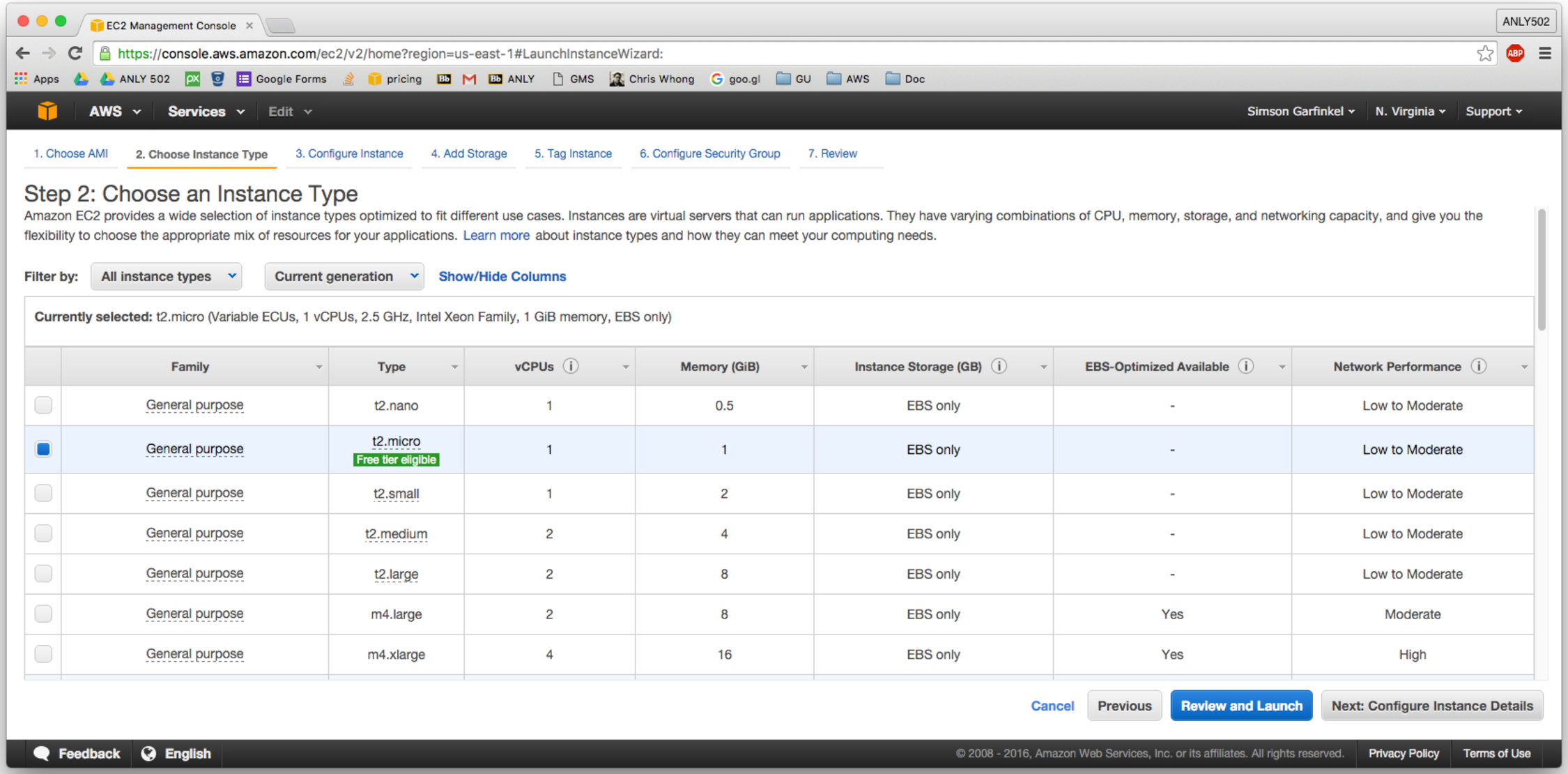

IAM Management Console
IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#roles
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Create Role
Step 1: Set Role Name
Step 2: Select Role Type
Step 3: Establish Trust
Step 4: Attach Policy
Step 5: Review
Attach Policy
Select one or more policies to attach. Each role can have up to 10 policies attached.
Filter:
Policy Type
Filter
Showing 192 results
Policy Name
Attached Entities
Creation Time
Edited Time
AmazonEC2FullAccess 1 2015-02-06 13:40 EST 2015-02-06 13:40 EST
AmazonElasticMapReduceRole 1 2015-02-06 13:41 EST 2016-02-10 17:41 EST
AmazonS3FullAccess 1 2015-02-06 13:40 EST 2015-02-06 13:40 EST
AdministratorAccess 0 2015-02-06 13:39 EST 2015-02-06 13:39 EST
AmazonAPIGatewayAdministrator 0 2015-07-09 13:34 EST 2015-07-09 13:34 EST
AmazonAPIGatewayInvokeFullAccess 0 2015-07-09 13:36 EST 2015-07-09 13:36 EST
AmazonAPIGatewayPushToCloudWatch... 0 2015-11-11 18:41 EST 2015-11-11 18:41 EST
AmazonAppStreamFullAccess 0 2015-02-06 13:40 EST 2015-02-06 13:40 EST
AmazonAppStreamReadOnlyAccess 0 2015-02-06 13:40 EST 2015-02-06 13:40 EST
AmazonCognitoDeveloperAuthenticated... 0 2015-03-24 13:22 EST 2015-03-24 13:22 EST
AmazonCognitoPowerUser 0 2015-03-24 13:14 EST 2015-03-24 13:14 EST
Cancel
Previous
Next Step

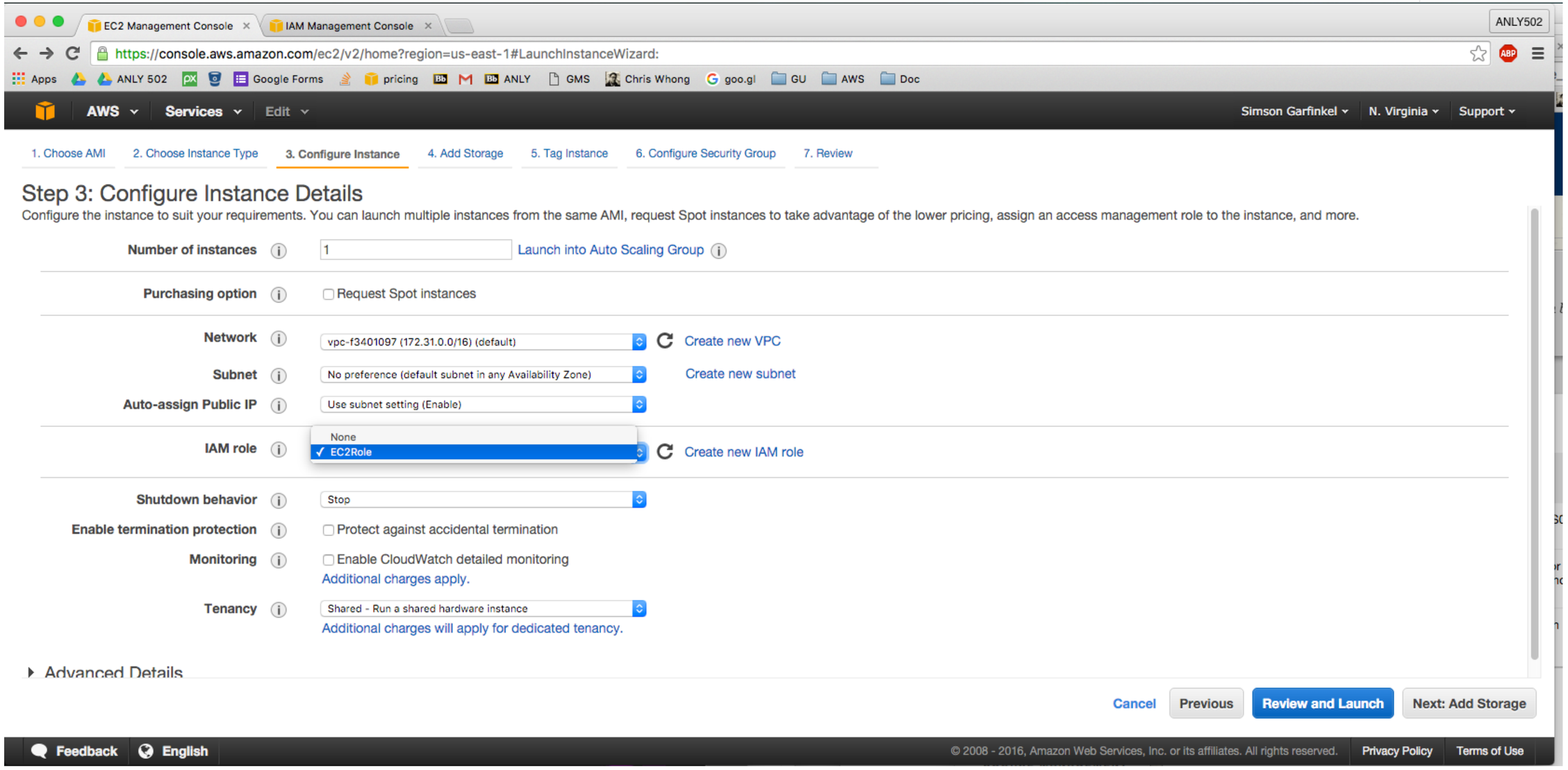

IAM Management Console
IAM Management Console
ANLY502
https://console.aws.amazon.com/iam/home?region=us-east-1#roles
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
Global
Support
Create Role
Step 1: Set Role Name
Step 2: Select Role Type
Step 3: Establish Trust
Step 4: Attach Policy
Step 5: Review
Review
Review the following role information. To edit the role, click an edit link, or click Create Role to finish.
Role Name
EC2Role
Edit Role Name
Role ARN
arn:aws:iam::489362128722:role/EC2Role
Trusted Entities
The identity provider(s) ec2.amazonaws.com
Policies
arn:aws:iam::aws:policy/AmazonS3FullAccess
arn:aws:iam::aws:policy/AmazonEC2FullAccess
arn:aws:iam::aws:policy/service-role/AmazonElasticMapReduceRole
Change Policies
Cancel
Previous
Create Role

No, when you create the new VM, specify the IAM Role



(by the way — you can leave a t2.micro running at the "free tier" without cost.)




EC2 Management Console
IAM Management Console
ANLY502
https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:
Apps
ANLY 502
Google Forms
pricing
ANLY
GMS
Chris Whong
goo.gl
GU
AWS
Doc
AWS
Services
Edit
Simson Garfinkel
N. Virginia
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-f3401097 (172.31.0.0/16) (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zone)
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
IAM role
None
EC2Role
Create new IAM role
Shutdown behavior
Stop
Enable termination protection
Protect against accidental termination
Monitoring
Enable CloudWatch detailed monitoring
Additional charges apply.
Tenancy
Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.
Advanced Details
Cancel
Previous
Review and Launch
Next: Add Storage
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

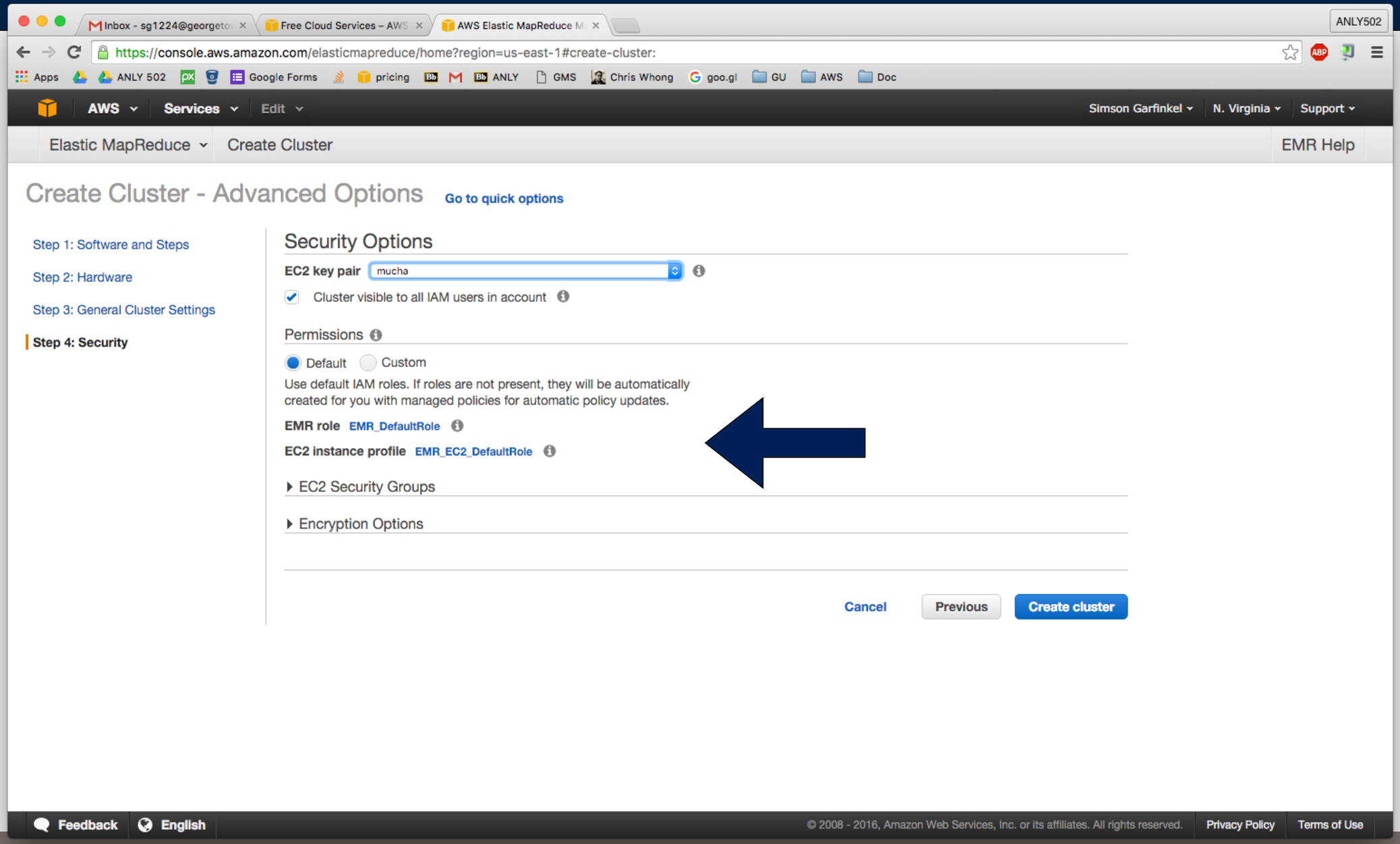
Now, when you log-in, you are pre-authenticated!

```
[Dance ~ 10:17:21]$ ssh -A ec2-user@52.87.205.16
The authenticity of host '52.87.205.16 (52.87.205.16)' can't be established.
ECDSA key fingerprint is SHA256:ddrORYwqYlMcvH9rwIjil6q4kx+2nSpJYrlljJC85fs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '52.87.205.16' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
[ec2-user@ip-172-31-61-18 ~]$ aws s3 ls s3://gu-anly502/ps03/forensicswiki
                           PRE forensicswiki/
[ec2-user@ip-172-31-61-18 ~]$ aws s3 ls s3://gu-anly502/ps03/forensicswiki/ | head
2016-02-14 20:55:54     507604 access.log.2012-01-01.gz
2016-02-14 20:55:54     652899 access.log.2012-01-02.gz
2016-02-14 20:55:54     823445 access.log.2012-01-03.gz
2016-02-14 20:55:54     813495 access.log.2012-01-04.gz
2016-02-14 20:55:54     867034 access.log.2012-01-05.gz
2016-02-14 20:55:54     748648 access.log.2012-01-06.gz
2016-02-14 20:55:54     565061 access.log.2012-01-07.gz
2016-02-14 20:55:54     639396 access.log.2012-01-08.gz
2016-02-14 20:55:54     956386 access.log.2012-01-09.gz
2016-02-14 20:55:54     862819 access.log.2012-01-10.gz
```

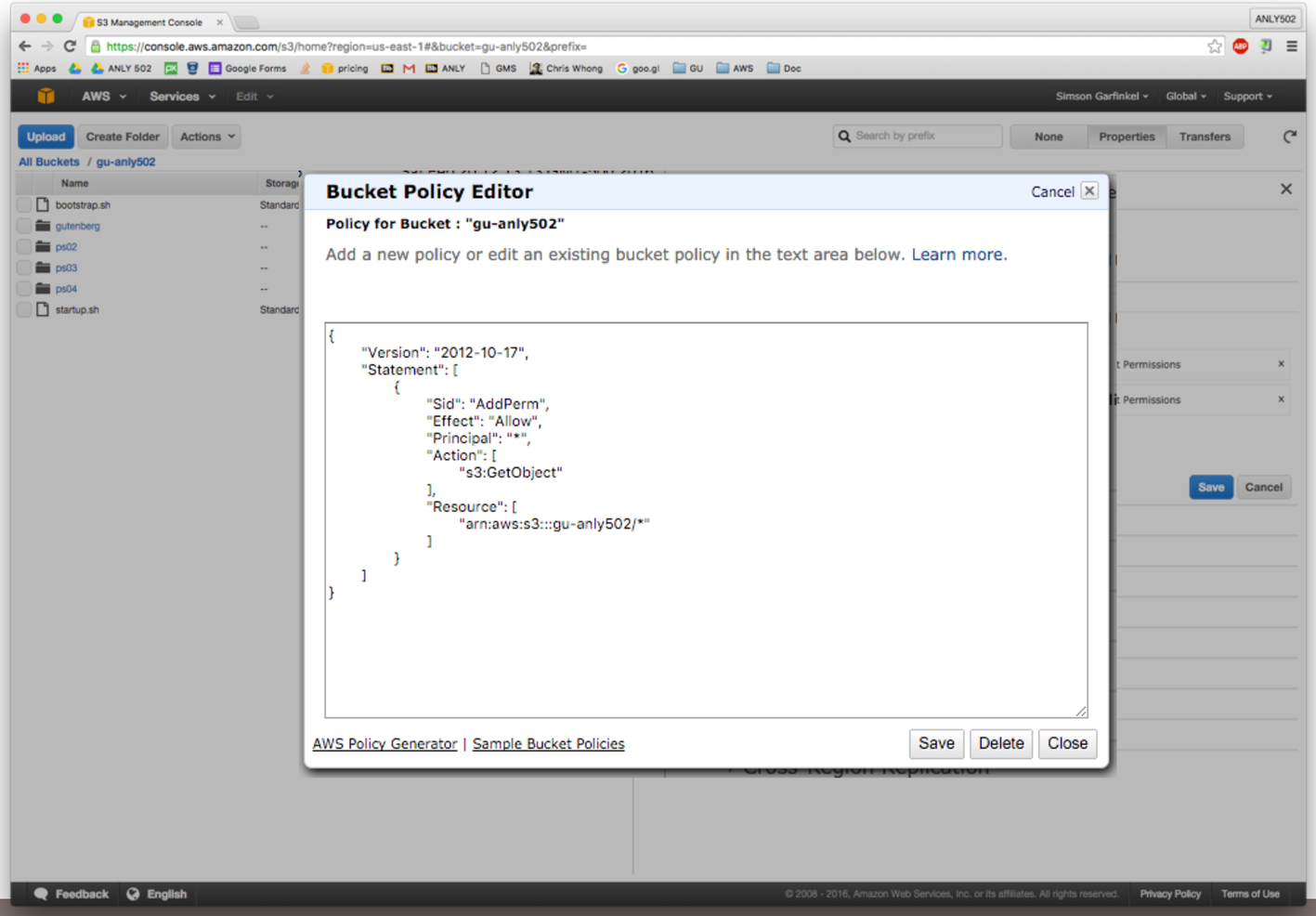
EMR_DefaultRole is how EMR reads & writes to S3



S3 Bucket Policies — All objects in the bucket get the same policy

Bucket policies are specified by JSON.

I have added a bucket policy to gu-anly502 so that all principals can perform s3:GetObject



AWS S3-Browser

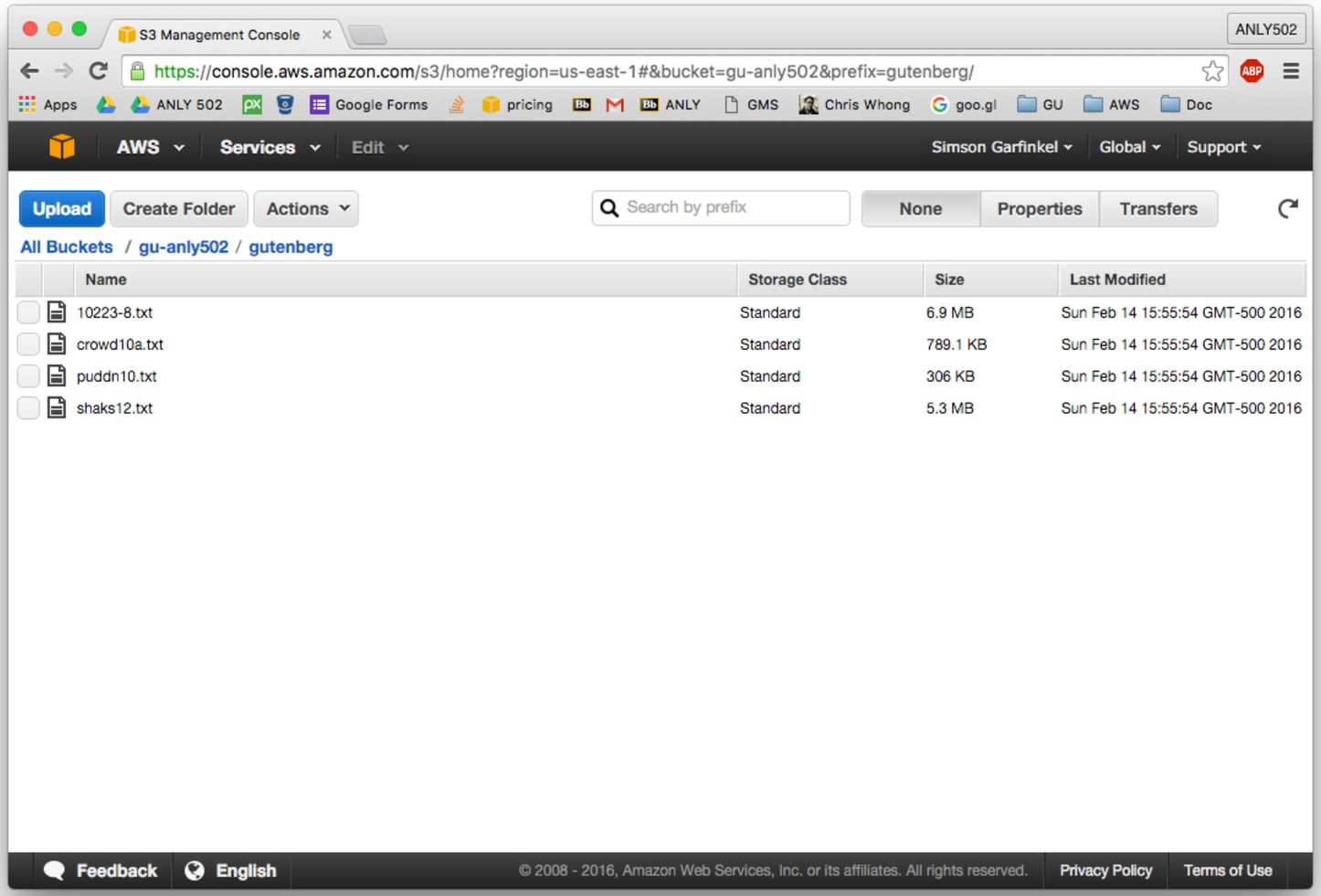
Built-in

Easy to download or upload a file

Modify permissions.

Set policies.

“Definitive.”

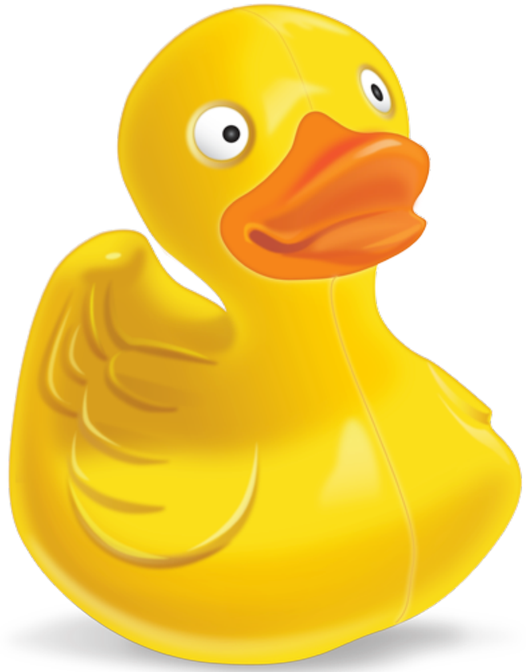


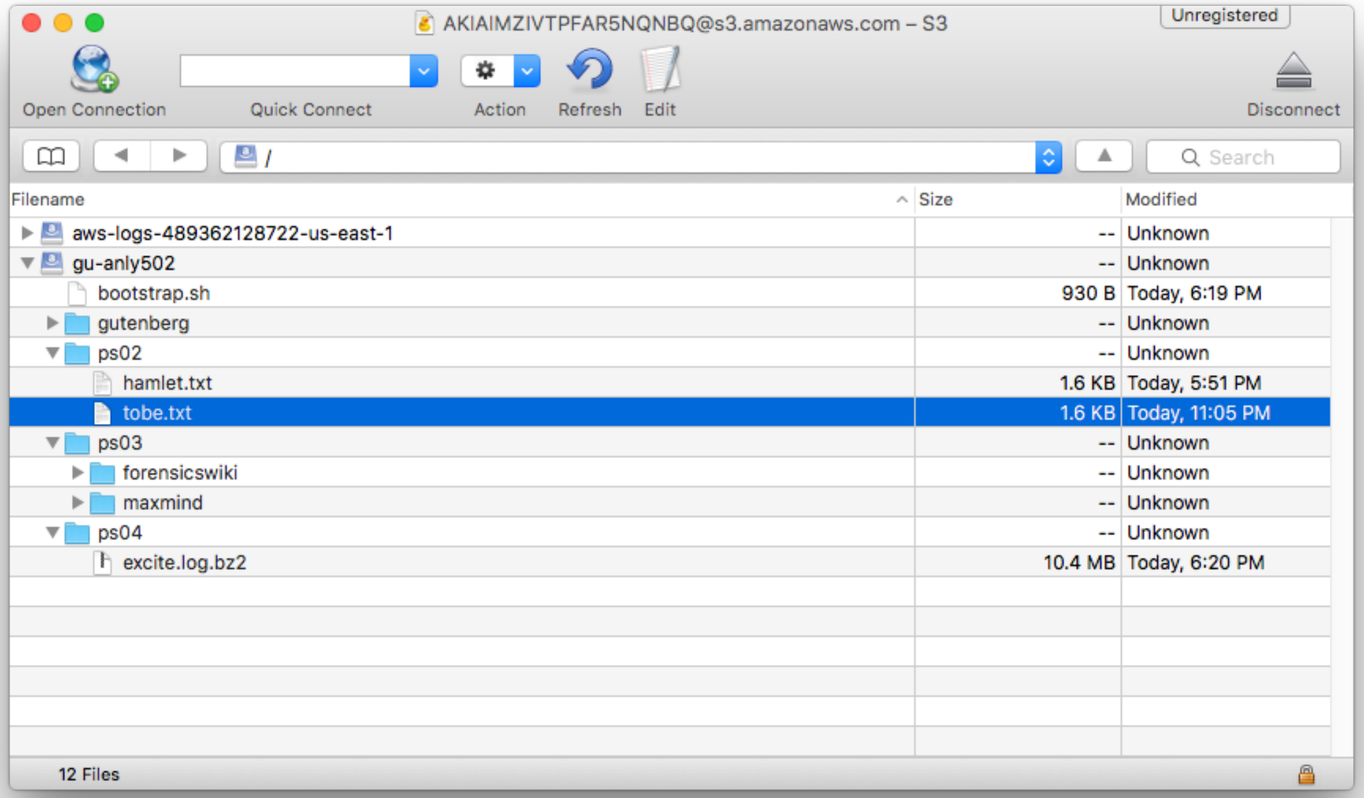
Cyberduck (Mac)

Drag & Drop Files

Select to edit permissions

Single-click editing





File Systems Compatible with Amazon EMR

http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-plan-file-systems.html

File System	Prefix	Description
HDFS	**`hdfs://`(or no prefix)**	HDFS is a distributed, scalable, and portable file system for Hadoop. An advantage of HDFS is data awareness between the Hadoop cluster nodes managing the clusters and the Hadoop cluster nodes managing the individual steps. For more information about how HDFS works, go to the Hadoop documentation. HDFS is used by the master and core nodes. One advantage is that it's fast; a disadvantage is that it's ephemeral storage which is reclaimed when the cluster ends. It's best used for caching the results produced by intermediate job-flow steps.
EMRFS	**`s3://`**	EMRFS is an implementation of HDFS used for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like Amazon S3 server-side encryption, read-after-write consistency, and list consistency. **Note:** Previously, Amazon EMR used the S3 Native FileSystem with the URI scheme, `s3n`. While this still works, we recommend that you use the `s3` URI scheme for the best performance, security, and reliability.
local file system		The local file system refers to a locally connected disk. When a Hadoop cluster is created, each node is created from an EC2 instance that comes with a preconfigured block of preattached disk storage called an instance store. Data on instance store volumes persists only during the life of its EC2 instance. Instance store volumes are ideal for storing temporary data that is continually changing, such as buffers, caches, scratch data, and other temporary content. For more information, see Amazon EC2 Instance Storage.
(Legacy) Amazon S3 block file system	**`s3bfs://`**	The Amazon S3 block file system is a legacy file storage system. We strongly discourage the use of this system. Important **IMPORTANT: We recommend that you do not use this file system because it can trigger a race condition that might cause your cluster to fail. However, it might be required by legacy applications.**

EMR times are highly variable.
However, it seems that there was a problem on Feb 13:

Sat Feb 13: Copy 2009-01-12-articles.tsv to HDFS:

```
[hadoop@ip-172-31-42-90 ~]$ time hdfs dfs -put /wikipedia/rawd/freebase-wex-2009-01-12-articles.tsv hdfs:///user/hadoop/infiles/


real 84m35.733s
user 1m12.176s
sys  0m59.660s
```

Sat Feb 13: Copy 2009-01-12-articles.tsv to S3:

```
[hadoop@ip-172-31-42-90 ~]$ time aws s3 cp /wikipedia/rawd/freebase-wex-2009-01-12-articles.tsv s3://anly502-slg/infiles/freebase-
wex-2009-01-12-articles.tsv
upload: ../../wikipedia/rawd/freebase-wex-2009-01-12-articles.tsv to s3://anly502-slg/infiles/freebase-wex-2009-01-12-articles.tsv

real 55m32.932s
user 4m43.408s
sys  2m6.236s
```

Sat Feb. 20 Copy to HDFS:

```
$ hdfs dfs -put /wikipedia/rawd/freebase-wex-2009-01-12-articles.tsv hdfs:///freebase-wex-2009-01-12-articles.tsv
681 seconds
```

Sat Feb. 20 Copy to S3:

```
$ aws s3 cp /wikipedia/rawd/freebase-wex-2009-01-12-articles.tsv s3://gu-anly502/
574 seconds
```

PS03 In Progress

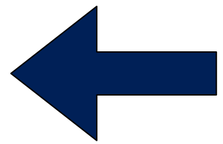
mrjob could not read all of the forensicswiki access.2012-??-??.gz files

I downloaded them to /mnt and ran them locally:

```
[last: 0s][~/ANLY502_SOLUTIONS/PS03]$ python join1.py /mnt/fwiki/
using configs in /home/hadoop/.mrjob.conf
creating tmp directory /tmp/join1.hadoop.20160220.203904.676391
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00000
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00001
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00002
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00003
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00004
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00005
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00006
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00007
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00008
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00009
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00010
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00011
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00012
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00013
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00014
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00015
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00016
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00017
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00018
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00019
...
```

Bad Data in PS03!

```
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00117
writing to /tmp/join1.hadoop.20160220.203904.676391/step-0-mapper_part-00118
Traceback (most recent call last):
  File "join1.py", line 52, in <module>
    FwikiMaxmindJoin.run()
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 461, in run
    mr_job.execute()
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 479, in execute
    super(MRJob, self).execute()
  File "/usr/local/lib/python2.7/site-packages/mrjob/launch.py", line 153, in execute
    self.run_job()
  File "/usr/local/lib/python2.7/site-packages/mrjob/launch.py", line 216, in run_job
    runner.run()
  File "/usr/local/lib/python2.7/site-packages/mrjob/runner.py", line 470, in run
    self._run()
  File "/usr/local/lib/python2.7/site-packages/mrjob/sim.py", line 173, in _run
    self._invoke_step(step_num, 'mapper')
  File "/usr/local/lib/python2.7/site-packages/mrjob/sim.py", line 260, in _invoke_step
    working_dir, env)
  File "/usr/local/lib/python2.7/site-packages/mrjob/inline.py", line 160, in _run_step
    child_instance.execute()
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 470, in execute
    self.run_mapper(self.options.step_num)
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 536, in run_mapper
    write_line(out_key, out_value)
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 707, in write_line
    print >> self.stdout, write(key, value)
  File "/usr/local/lib/python2.7/site-packages/mrjob/protocol.py", line 75, in write
    self._dumps(value))
  File "/usr/local/lib/python2.7/site-packages/mrjob/protocol.py", line 88, in _dumps
    return json.dumps(value)
  File "/usr/local/lib64/python2.7/site-packages/simplejson/__init__.py", line 261, in dumps
    return _default_encoder.encode(obj)
  File "/usr/local/lib64/python2.7/site-packages/simplejson/encoder.py", line 208, in encode
    return encode_basestring_ascii(o)
UnicodeDecodeError: 'utf8' codec can't decode byte 0xcf in position 119: invalid continuation byte
[last: 1342s][~/ANLY502_SOLUTIONS/PS03]$
```



I wrote a small program to scan for the bad data.

Scanning program:

```
#!/usr/bin/python2.7

from __future__ import print_function
import sys

for fname in sys.argv[1:]:
    print(fname)
    with open(fname,"r") as f:
        for line in f:
            line.decode('utf8')
```

1. **Open each file.**
2. **Try to decode each line as "utf"**

Run it:

```
$ python scan.py /mnt/fwiki/*
/mnt/fwiki/access.log.2012-01-01
/mnt/fwiki/access.log.2012-01-02
/mnt/fwiki/access.log.2012-01-03
/mnt/fwiki/access.log.2012-01-04
/mnt/fwiki/access.log.2012-01-05
...
/mnt/fwiki/access.log.2012-02-21
/mnt/fwiki/access.log.2012-02-22
Traceback (most recent call last):
  File "scan.py", line 10, in <module>
    line.decode('utf8')
  File "/usr/lib64/python2.7/encodings/utf_8.py", line 16, in decode
    return codecs.utf_8_decode(input, errors, True)
UnicodeDecodeError: 'utf8' codec can't decode byte 0xae in position 366: invalid start byte
```

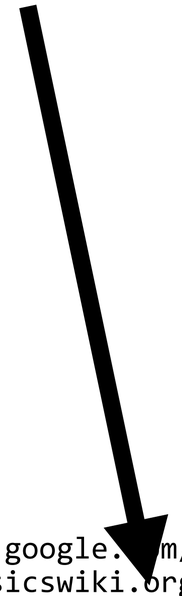
I modified program to print the line number of the bad data:

New version:

```
from __future__ import print_function
import sys

for fname in sys.argv[1:]:
    print(fname)
    with open(fname,"r") as f:
        number = 0
        for line in f:
            number += 1
            try:
                line.decode('utf8')
            except UnicodeDecodeError:
                print("bad line {}: {}".format(number,line))
```

Invalid UTF-8!
supplied by the browser!



Output:

```
$ python scan.py /mnt/fwiki/access.log.2012-02-22
/mnt/fwiki/access.log.2012-02-22
bad line 8606: 72.199.97.164 - - [22/Feb/2012:05:12:23 -0800] "GET /wiki/Tools HTTP/1.1" 200 15246 "http://www.google.com/url?
sa=t&rct=j&q=backtrack+5+bulk+extractor+on+pc+drive+c&source=web&cd=3&ved=0CDIQFjAC&url=http%3A%2F%2Fwww.forensicswiki.org%2Fwiki
%2FTools&ei=culET7-HPK6DsAKw5_DCDw&usg=AFQjCNF25Wt28dse6kfr2zWDOqOTRFgEZg&sig2=nw5yN2XYKAgw1Pab2RJ7-A" "Mozilla/5.0 (CaLiKiNgZ?; U;
CPU OS 4_3_3 like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko) Version/5.0.2 Mobile/8J3 Safari/6533.18.5"

bad line 8607: 72.199.97.164 - - [22/Feb/2012:05:12:23 -0800] "GET /w/load.php?
debug=false&lang=en&modules=site&only=styles&skin=monobook&* HTTP/1.1" 200 1547 "http://www.forensicswiki.org/wiki/Tools" "Mozilla/5.0
(CaLiKiNgZ?; U; CPU OS 4_3_3 like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko) Version/5.0.2 Mobile/8J3 Safari/6533.18.5"
```

Verify the bug by running join1.py with *just the suspicious file*

```
[last: 684s][~/ANLY502_SOLUTIONS/PS03]$ python join1.py /mnt/fwiki/access.log.2012-02-22
using configs in /home/hadoop/.mrjob.conf
creating tmp directory /tmp/join1.hadoop.20160220.213828.279090
writing to /tmp/join1.hadoop.20160220.213828.279090/step-0-mapper_part-00000
Traceback (most recent call last):
  File "join1.py", line 52, in <module>
    FwikiMaxmindJoin.run()
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 461, in run
    mr_job.execute()
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 479, in execute
    super(MRJob, self).execute()
  File "/usr/local/lib/python2.7/site-packages/mrjob/launch.py", line 153, in execute
    self.run_job()
  File "/usr/local/lib/python2.7/site-packages/mrjob/launch.py", line 216, in run_job
    runner.run()
  File "/usr/local/lib/python2.7/site-packages/mrjob/runner.py", line 470, in run
    self._run()
  File "/usr/local/lib/python2.7/site-packages/mrjob/sim.py", line 173, in _run
    self._invoke_step(step_num, 'mapper')
  File "/usr/local/lib/python2.7/site-packages/mrjob/sim.py", line 260, in _invoke_step
    working_dir, env)
  File "/usr/local/lib/python2.7/site-packages/mrjob/inline.py", line 160, in _run_step
    child_instance.execute()
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 470, in execute
    self.run_mapper(self.options.step_num)
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 536, in run_mapper
    write_line(out_key, out_value)
  File "/usr/local/lib/python2.7/site-packages/mrjob/job.py", line 707, in write_line
    print >> self.stdout, write(key, value)
  File "/usr/local/lib/python2.7/site-packages/mrjob/protocol.py", line 75, in write
    self._dumps(value))
  File "/usr/local/lib/python2.7/site-packages/mrjob/protocol.py", line 88, in _dumps
    return json.dumps(value)
  File "/usr/local/lib64/python2.7/site-packages/simplejson/__init__.py", line 261, in dumps
    return _default_encoder.encode(obj)
  File "/usr/local/lib64/python2.7/site-packages/simplejson/encoder.py", line 208, in encode
    return encode_basestring_ascii(o)
UnicodeDecodeError: 'utf8' codec can't decode byte 0xae in position 366: invalid start byte
[last: 2s][~/ANLY502_SOLUTIONS/PS03]$
```

Note: This is happening in mrjob's write_line() function inside run_mapper().

Our code never sees the bad data.

There were other errors — non HTTP access logs in the access.log file

```
[last: 5s][~]$ egrep -v 'GET|POST|HEAD|OPTIONS|PUT|PROPFIND|LOCK' /mnt/access.log-filtered.txt
205  1          1          180
14920           206        11         2          129715
14920           207        2          2          16366
14920           208        1          1          8488
14920           209        7          7          85566
14920           210        7          4          21788
14920           211        30         4          276107
14920           215        1          1          180
14920           216        3          3          22117
14920           218        7          7          65194
14920           219        2          2          10109
```

You need to handle bad data:

- Instead of this

```
o = Weblog(line)
```

- Use this:

```
try:
    o = Weblog(line)
except ValueError:
    return
```

Performance issues — the small file problem

```
$ aws s3 ls s3://gu-anly502/ps03/
                           PRE forensicswiki.2012-01.unzipped/
                           PRE forensicswiki/
                           PRE maxmind/
2016-02-20 18:13:20 4268793922 forensicswiki.2012.txt
2016-02-20 15:44:37 32788306263 freebase-wex-2009-01-12-articles.tsv
$
```

Time to read 2012 forensicswiki files with a 3-m3xlarge node cluster:

s3://gu-anly502/ps03/forensicswiki/	s3://gu-anly502/ps03/forensicswiki.2012.txt
1547 seconds	1026 seconds
365 .gz files ≈ 1MB each ≈ 330 MB	1 file, 4GB

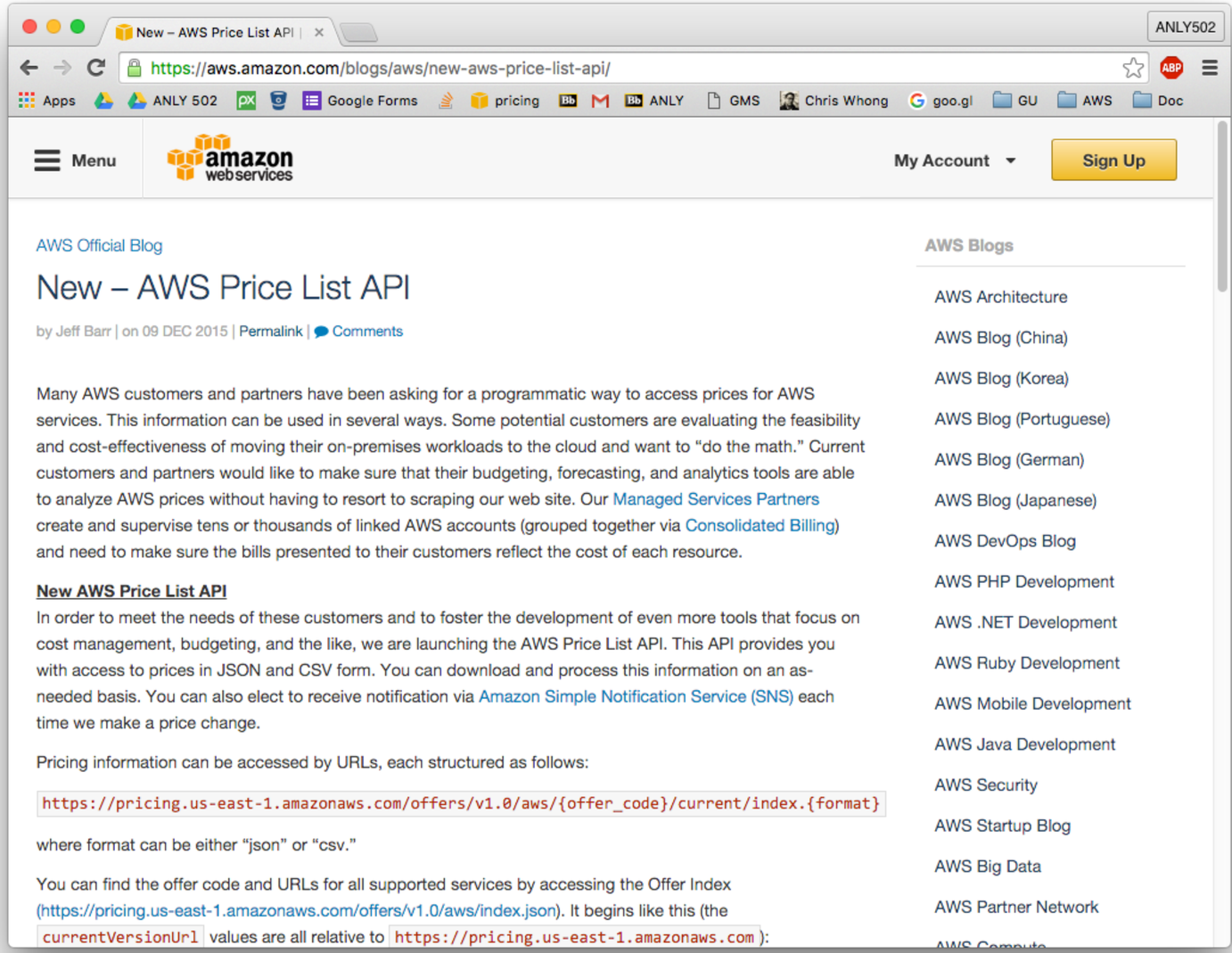
Using Amazon's API to find out about real-time pricing

AWS Price List API

- Offers — Services AWS is offering
- Products — e.g. VM instances
- Terms — e.g. OnDemand, Annual, etc.

Example:


```
"RDXNGJU5DRW4G5ZK" : {
  "sku" : "RDXNGJU5DRW4G5ZK",
  "productFamily" : "Compute Instance",
  "attributes" : {
    "servicecode" : "AmazonEC2",
    "location" : "South America (Sao Paulo)",
    "locationType" : "AWS Region",
    "instanceType" : "c3.large",
    "currentGeneration" : "Yes",
    "instanceFamily" : "Compute optimized",
    "vcpu" : "2",
    "physicalProcessor" : "Intel Xeon E5-2680 v2 (Ivy Bridge)",
    "clockSpeed" : "2.8 GHz",
    "memory" : "3.75 GiB",
    "storage" : "2 x 16 SSD",
    "networkPerformance" : "Moderate",
    "processorArchitecture" : "32-bit or 64-bit",
    "tenancy" : "Host",
    "operatingSystem" : "Linux",
    "licenseModel" : "No License required",
    "usagetype" : "SAE1-HostBoxUsage:c3.large",
    "operation" : "RunInstances",
    "enhancedNetworkingSupported" : "Yes",
    "preInstalledSw" : "NA",
    "processorFeatures" : "Intel AVX; Intel Turbo"
  }
},
```

Print the offers:

```
#!/usr/bin/env python3.5
offer_index_url = "https://pricing.us-east-1.amazonaws.com/offers/v1.0/aws/index.json"
import json,urllib.request
from tabulate import tabulate

if __name__=="__main__":
    offers = json.loads(urllib.request.urlopen(offer_index_url).read().decode('utf-8'))
    assert(offers['formatVersion']=='v1.0')

    table = [["Offer","offerCode","currentVersionUrl"]]

    for name in offers['offers']:
        od = offers['offers'][name]
        table.append([name,od['offerCode'],od['currentVersionUrl']])
    print("The following offers are available:")
    print(tabulate(table,headers="firstrow",tablefmt="simple"))
```

- Output:

```
$ ./aws_costing.py
The following offers are available:
Offer              offerCode          currentVersionUrl
-----------------  -----------------  ------------------------------------------------------
AmazonRedshift     AmazonRedshift     /offers/v1.0/aws/AmazonRedshift/current/index.json
AmazonSimpleDB     AmazonSimpleDB     /offers/v1.0/aws/AmazonSimpleDB/current/index.json
AmazonRDS          AmazonRDS          /offers/v1.0/aws/AmazonRDS/current/index.json
AmazonSES          AmazonSES          /offers/v1.0/aws/AmazonSES/current/index.json
AmazonRoute53      AmazonRoute53      /offers/v1.0/aws/AmazonRoute53/current/index.json
AmazonVPC          AmazonVPC          /offers/v1.0/aws/AmazonVPC/current/index.json
awskms             awskms             /offers/v1.0/aws/awskms/current/index.json
AmazonEC2          AmazonEC2          /offers/v1.0/aws/AmazonEC2/current/index.json
AmazonElastiCache  AmazonElastiCache  /offers/v1.0/aws/AmazonElastiCache/current/index.json
AmazonS3           AmazonS3           /offers/v1.0/aws/AmazonS3/current/index.json
AmazonCloudFront   AmazonCloudFront   /offers/v1.0/aws/AmazonCloudFront/current/index.json
AmazonDynamoDB     AmazonDynamoDB     /offers/v1.0/aws/AmazonDynamoDB/current/index.json
AmazonGlacier      AmazonGlacier      /offers/v1.0/aws/AmazonGlacier/current/index.json
```

Print the VMs:

```
# Get the EC2 offers
assert ec2_code in offers['offers']
ec2_url  = base+offers['offers'][ec2_code]['currentVersionUrl']
ec2_json = urllib.request.urlopen(ec2_url).read().decode('utf-8')

# Get all of the current ec2 offers
ec2_products = ec2_info['products']
print("Number of products available: {}".format(len(ec2_products)))
ec2_terms    = ec2_info['terms']
print("Terms available: {}".format(" ".join(ec2_terms)))

# Assemble an array of the instance types

instances = []
for (product,vals) in ec2_products.items():
    try:
           patts = vals['attributes']
           for(pk,pv) in  ec2_terms['OnDemand'][product].items():
                for(dk,dv) in pv['priceDimensions'].items():
                    gb   = float(patts['memory'].replace("GiB",""))
                    vcpu = float(patts['vcpu'])
                    row = (product, # row[0]
                           patts['instanceType'], # row[1]
                           vcpu,         # row[2]
                           gb, # row[3]
                           gb/vcpu,                 # row[4]
                           float(dv['pricePerUnit']['USD'])) # row[5]
                    instances.append(row)
    except KeyError:
        # Missing data
        pass
```

Convert text to float

9699 different product codes...

product	instanceType	vCPU	Memory	GiB/cpu	pricePerUnit
SZAG69AWYJF676BA	d2.4xlarge	16	122	7.625	0
DSG34N2933CDGRJJ	c4.xlarge	4	7.5	1.875	0
PYCJPPPYA7FXP2KM	m4.large	2	8	4	0
J28DJ6QCZ8VU7DZQ	d2.8xlarge	36	244	6.77778	6.198
62G57MU6KCQ2AQS8	t1.micro	1	0.613	0.613	0.08
7MVN3GT6EP25KDUJ	cc2.8xlarge	32	60.5	1.89062	2
232CDFDW89ENUXRB	d2.8xlarge	36	244	6.77778	0
H3H6PVAND793CJ85	c4.8xlarge	36	60	1.66667	0
86FEVXHAJVJ75D5R	c4.2xlarge	8	15	1.875	0.773
K4FQKJH96JE6DDW2	m3.xlarge	4	15	3.75	0
QZS65ZVZAUNM545N	hi1.4xlarge	16	60.5	3.78125	3.23
XBYJG3BUDTPN8NB9	cc2.8xlarge	32	60.5	1.89062	2.57
TB8JSDKA7MEGTRXV	m4.large	2	8	4	0.132
Q73NFXYCVJRVJD5P	i2.8xlarge	32	244	7.625	9.836
VN8JS6C4CHVEY8WD	i2.4xlarge	16	122	7.625	0.1
ERPWM7KEFVQABEK6	m3.xlarge	4	15	3.75	0
MHX8TSHV6Z45N5KU	d2.xlarge	4	30.5	7.625	0.759
GK3JQYYZHNZAHQ66	r3.2xlarge	8	61	7.625	0
FRR3BPV6Y433HGXY	d2.8xlarge	36	244	6.77778	6.198
PA99ECAE74DADX5J	c4.4xlarge	16	30	1.875	2.408

...

Restrict to N. Virginia, Shared, Linux, and price>0

```
# Get the EC2 offers
assert ec2_code in offers['offers']
ec2_url  = base+offers['offers'][ec2_code]['currentVersionUrl']
ec2_json = urllib.request.urlopen(ec2_url).read().decode('utf-8')

# Get all of the current ec2 offers
ec2_products = ec2_info['products']
print("Number of products available: {}".format(len(ec2_products)))
ec2_terms    = ec2_info['terms']
print("Terms available: {}".format(" ".join(ec2_terms)))

# Assemble an array of the instance types

instances = []
for (product,vals) in ec2_products.items():
    try:
        patts = vals['attributes']
        if patts['location']=='US East (N. Virginia)' and \
                patts['tenancy']=='Shared' and \
                patts['operatingSystem'] == 'Linux':
            for(pk,pv) in  ec2_terms['OnDemand'][product].items():
                for(dk,dv) in pv['priceDimensions'].items():
                    gb   = float(patts['memory'].replace("GiB",""))
                    vcpu = float(patts['vcpu'])
                    row = (product, # row[0]
                           patts['instanceType'], # row[1]
                           vcpu,         # row[2]
                           gb, # row[3]
                           gb/vcpu,                # row[4]
                           float(dv['pricePerUnit']['USD'])) # row[5]
                    # Convert values as necessary
                    if row[4]>0:
                        instances.append(row)
    except KeyError:
        # Missing data
        pass
```

Select N. VA / Shared / Linux

Convert text to float

54 total

product	instanceType	vCPU	Memory	GiB/cpu	pricePerUnit
RJZ63YZJGC58TPTS	hi1.4xlarge	16	60.5	3.78125	3.1
AGHHWVT6KDRBWTWP	t2.nano	1	0.5	0.5	0.0065
3DX9M63484ZSZFJV	cc2.8xlarge	32	60.5	1.89062	2
3UP33R2RXCADSPSX	m4.4xlarge	16	64	4	0.958
VHC3YWSZ6ZFZPJN4	m4.2xlarge	8	32	4	0.479
QY3YSEST3C6FQNQH	t2.medium	2	4	2	0.052
A67CJDV9B3YBP6N6	g2.8xlarge	32	60	1.875	2.6
2GCTBU78G22TGEXZ	m1.small	1	1.7	1.7	0.044
5KHB4S5E8M74C6ES	i2.xlarge	4	30.5	7.625	0.853
ZESHW7CZVERW2BN2	i2.4xlarge	16	122	7.625	3.41
6TEX73KEE94WMEED	c1.xlarge	8	7	0.875	0.52
P63NKZQXED5H7HUK	d2.2xlarge	8	61	7.625	1.38
RKCQDTMY5DZS4JWT	m2.4xlarge	8	68.4	8.55	0.98
X4RWGEB2DKQGCWC2	c1.medium	2	1.7	0.85	0.13
ASDZTDFMC5425T7P	m3.medium	1	3.75	3.75	0.067
QG5G45WKDWDDHTFV	t2.large	2	8	4	0.104
48VURD6MVAZ3M5JX	g2.2xlarge	8	15	1.875	0.65
639ZEB9D49ASFB26	t1.micro	1	0.613	0.613	0.02
U7343ZA6ABZUXFZ9	d2.xlarge	4	30.5	7.625	0.69
NARXYND9H74FTC7A	i2.8xlarge	32	244	7.625	6.82
ZJC9VZJF5NZNYSVK	d2.4xlarge	16	122	7.625	2.76
J4T9ZF4AJ2DXE7SA	m4.10xlarge	40	160	4	2.394
3RUU5T58T7XAFAAF	cr1.8xlarge	32	244	7.625	3.5
YGU2QZY8VPP94FSR	m3.large	2	7.5	3.75	0.133
4TCUDNKW7PMPSUT2	r3.8xlarge	32	244	7.625	2.66
MU4QGTJYWR6T73MZ	i2.2xlarge	8	61	7.625	1.705
HZC9FAP4F9Y8JW67	t2.micro	1	1	1	0.013
...					

Remember:

Always develop with a small data set.

- If you can, develop with -r local or -r inline (data must be in local file system)

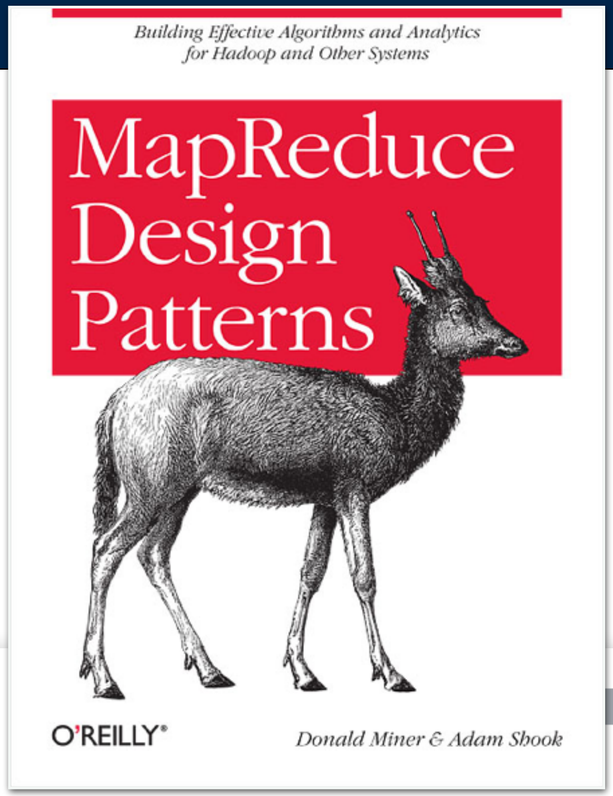
Student Presentations

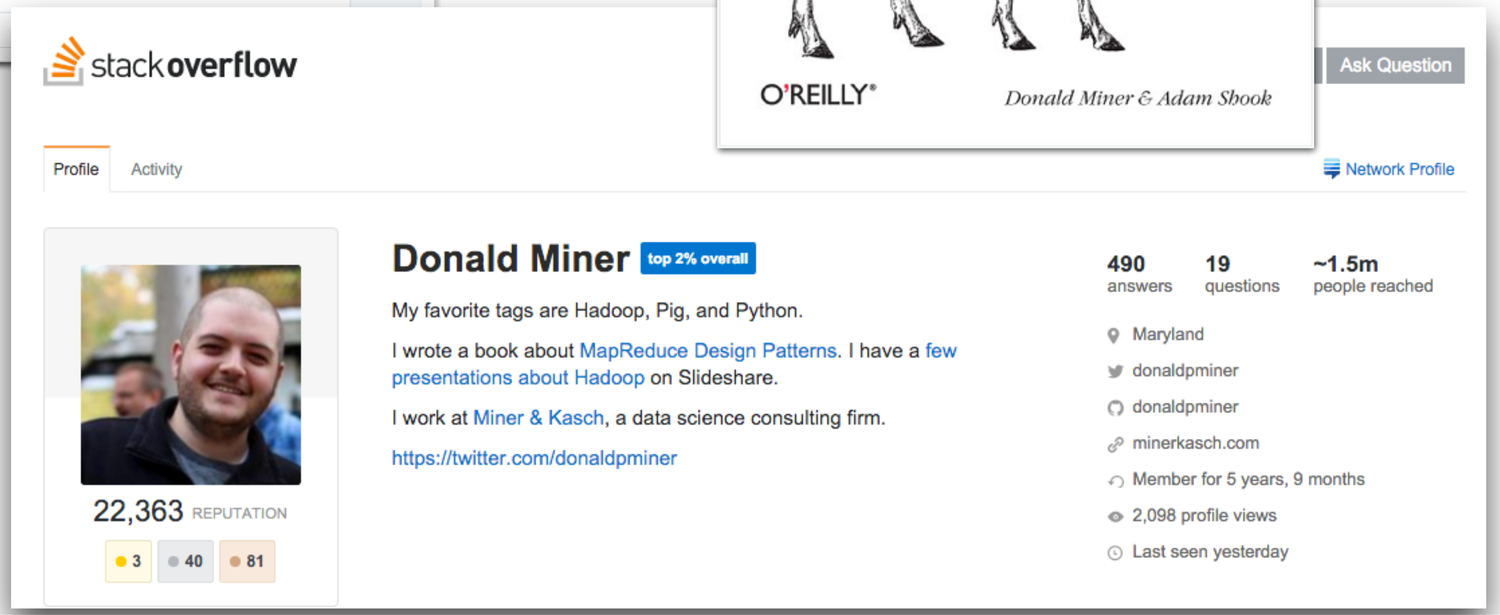
4 presentations in 20 minutes!

Xiuli Wang	Paper	YFCC100M: the new data in multimedia research
Jianze Zhou	Paper	The Beckman report on database research
Daodao Wang	Program	Apache Mahout Clustering
Ron Graf	Paper	GraphLab: A New Framework For Parallel Machine Learning

Don Miner

Guest Speaker: Donald Miner





Pig

Apache Pig

Started at Yahoo! Research

- Easier approach for MapReduce
- Procedural language
- PigLatin scripts *interpreted and run as* MapReduce jobs.

Pig Advantages:

- Easier to program than MapReduce.
- Declarative statements directly describe data transformations.
- Optimizer makes efficient decisions.
- Debugging operators:
  - *DESCRIBE, EXPLAIN, ILLUSTRATE*
- Can run “locally” or on Hadoop.

Pig Disadvantages:

- Simple statements may generate many MapReduce jobs.
- Can be hard to debug.
- Keywords are case insensitive
  - *LOAD, USING, AS, GROUP, BY, ...*
- Functions, relations, fields are case sensitive:
  - *PigStorage, COUNT,*

Pig reference materials in Readings/L05 Databases

Pig Latin: A Not-So-Foreign Language for Data Processing

Christopher Olston* Yahoo! Research
Benjamin Reed† Yahoo! Research
Utkarsh Srivastava‡ Yahoo! Research
Ravi Kumar§ Yahoo! Research
Andrew Tomkins¶ Yahoo! Research

ABSTRACT

There is a growing need for ad-hoc analysis of extremely large data sets, especially at internet companies where innovation critically depends on being able to analyze terabytes of data collected every day. Parallel database products, e.g., Teradata, offer a solution, but are usually prohibitively expensive at this scale. Besides, many of the people who analyze this data are entrenched procedural programmers, who find the declarative, SQL style to be unnatural. The success of the more procedural *map-reduce* programming model, and its associated scalable implementations on commodity hardware, is evidence of the above. However, the map-reduce paradigm is too low-level and rigid, and leads to a great deal of custom user code that is hard to maintain, and reuse.

We describe a new language called *Pig Latin* that we have designed to fit in a sweet spot between the declarative style of SQL, and the low-level, procedural style of map-reduce. The accompanying system, Pig, is fully implemented, and compiles Pig Latin into physical plans that are executed over *Hadoop*, an open-source, map-reduce implementation. We give a few examples of how engineers at Yahoo! are using Pig to dramatically reduce the time required for the development and execution of their data analysis tasks, compared to using Hadoop directly. We also report on a novel debugging environment that comes integrated with Pig, that can lead to even higher productivity gains. Pig is an open-source, Apache-incubator project, and available for general use.

Categories and Subject Descriptors:
H.2.3 Database Management: Languages

General Terms: Languages.

*olston@yahoo-inc.com
†breed@yahoo-inc.com
‡utkarsh@yahoo-inc.com
§ravikuma@yahoo-inc.com
¶atomkins@yahoo-inc.com

SIGMOD'08, June 9–12, 2008, Vancouver, BC, Canada.

1. INTRODUCTION

At a growing number of organizations, innovation revolves around the collection and analysis of enormous data sets such as web crawls, search logs, and click streams. Internet companies such as Amazon, Google, Microsoft, and Yahoo! are prime examples. Analysis of this data constitutes the innermost loop of the product improvement cycle. For example, the engineers who develop search engine ranking algorithms spend much of their time analyzing search logs looking for exploitable trends.

The sheer size of these data sets dictates that it be stored and processed on highly parallel systems, such as shared-nothing clusters. Parallel database products, e.g., Teradata, Oracle RAC, Netezza, offer a solution by providing a simple SQL query interface and hiding the complexity of the physical cluster. These products however, can be prohibitively expensive at web scale. Besides, they wrench programmers away from their preferred method of analyzing data, namely writing imperative scripts or code, toward writing declarative queries in SQL, which they often find unnatural, and overly restrictive.

As evidence of the above, programmers have been flocking to the more procedural *map-reduce* [4] programming model. A map-reduce program essentially performs a group-by-aggregation in parallel over a cluster of machines. The programmer provides a map function that dictates how the grouping is performed, and a reduce function that performs the aggregation. What is appealing to programmers about this model is that there are only two high-level declarative primitives (map and reduce) to enable parallel processing, but the rest of the code, i.e., the map and reduce functions, can be written in any programming language of choice, and without worrying about parallelism.

Unfortunately, the map-reduce model has its own set of limitations. Its one-input, two-stage data flow is extremely rigid. To perform tasks having a different data flow, e.g., joins or n stages, inelegant workarounds have to be devised. Also, custom code has to be written for even the most common operations, e.g., projection and filtering. These factors lead to code that is difficult to reuse and maintain, and in which the semantics of the analysis task are obscured. Moreover, the opaque nature of the map and reduce functions impedes the ability of the system to perform optimizations.

We have developed a new language called *Pig Latin* that combines the best of both worlds: high-level declarative querying in the spirit of SQL, and low-level, procedural programming à la map-reduce.

Olston 2008 Pig Latin

Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience

Alan F. Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, Utkarsh Srivastava
Yahoo!, Inc.*

ABSTRACT

Increasingly, organizations capture, transform and analyze enormous data sets. Prominent examples include internet companies and e-science. The *Map-Reduce* scalable dataflow paradigm has become popular for these applications. Its simple, explicit dataflow programming model is favored by some over the traditional high-level declarative approach: SQL. On the other hand, the extreme simplicity of Map-Reduce leads to much low-level hacking to deal with the many-step, branching dataflows that arise in practice. Moreover, users must repeatedly code standard operations such as *join* by hand. These practices waste time, introduce bugs, harm readability, and impede optimizations.

Pig is a high-level dataflow system that aims at a sweet spot between SQL and Map-Reduce. Pig offers SQL-style high-level data manipulation constructs, which can be assembled in an explicit dataflow and interleaved with custom Map- and Reduce-style functions or executables. Pig programs are compiled into sequences of Map-Reduce jobs, and executed in the *Hadoop* Map-Reduce environment. Both Pig and Hadoop are open-source projects administered by the Apache Software Foundation.

This paper describes the challenges we faced in developing Pig, and reports performance comparisons between Pig execution and raw Map-Reduce execution.

1. INTRODUCTION

Organizations increasingly rely on ultra-large-scale data processing in their day-to-day operations. For example, modern internet companies routinely process petabytes of web content and usage logs to populate search indexes and perform ad-hoc mining tasks for research purposes. The data includes unstructured elements (e.g., web page text; images) as well as structured elements (e.g., web page click records; extracted entity-relationship models). The processing combines generic relational-style operations (e.g., filter; join; count) with specialized domain-specific operations (e.g., part-of-speech tagging; face detection). A similar situation arises in e-science, national intelligence, and other domains.

The popular *Map-Reduce* [8] scalable data processing framework, and its open-source realization *Hadoop* [1], cater to these workloads and offer a simple dataflow programming model that appeals to many users. However, in practice, the extreme simplicity of the Map-Reduce programming model leads to several problems. First, it does not directly support complex N-step dataflows, which often arise in practice. Map-Reduce also lacks explicit support for combined processing of multiple data sets (e.g., joins and other data matching operations), a crucial aspect of knowledge discovery. Lastly, frequently-needed data manipulation primitives like filtering, aggregation and top-k thresholding must be coded by hand.

Consequently, users end up stitching together Map-Reduce dataflows by hand, hacking multi-input flows, and repeatedly implementing standard operations inside black-box functions. These practices slow down data analysis, introduce mistakes, make data processing programs difficult to read, and impede automated optimization.

Our *Pig* system [4] offers composable high-level data manipulation constructs in the spirit of SQL, while at the same time retaining the properties of Map-Reduce systems that make them attractive for certain users, data types, and workloads. In particular, as with Map-Reduce, Pig programs encode explicit dataflow graphs, as opposed to implicit dataflow as in SQL. As one user from Adobe put it:

> "Pig seems to give the necessary parallel programming constructs (FOREACH, FLATTEN, COGROUP .. etc) and also give sufficient control back to the programmer (which a purely declarative approach like [SQL on top of Map-Reduce][1] doesn't)."

Pig dataflows can interleave built-in relational-style operations like filter and join, with user-provided executables (scripts or pre-compiled binaries) that perform custom processing. Schemas for the relational-style operations can be supplied at the last minute, which is convenient when working with temporary data for which system-managed metadata is more of a burden than a benefit. For data used

*Author email addresses: {gates, olgan, shubhamc, pradeepk, shravanm, olston, breed, sms, utkarsh}@yahoo-inc.com.

VLDB '09, August 24-28, 2009, Lyon, France

[1]Reference to specific software project removed.

Gates 2009 The Pig Experience

Pig Latin Reference Manual 2

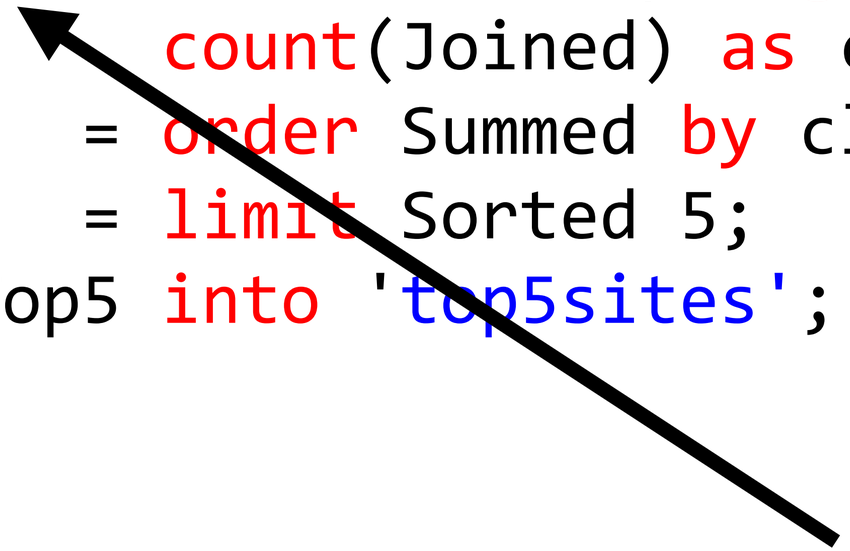
by

Table of contents

PigLatin Reference Manual V2

Famous example:
Pig program to find top 5 websites for Twitter users age 18-25

```
Users    = load 'users' as (name, age);
Filtered = filter Users by age >= 18 and age <= 25;
Pages    = load 'pages' as (user, url);
Joined   = join Filtered by name, Pages by user;
Grouped  = group Joined by url;
Summed   = foreach Grouped generate group,
           count(Joined) as clicks;
Sorted   = order Summed by clicks desc;
Top5     = limit Sorted 5;
store Top5 into 'top5sites';
```



Equivalent MapReduce program (in Java)

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobControl;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String key = line.substring(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(',');
            String value = line.substring(firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it's from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
                String value = t.toString();
                if (value.charAt(0) == '1')
first.add(value.substring(1));
                else second.add(value.substring(1));
                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputCollector<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(',');
            int secondComma = line.indexOf(',', firstComma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don't need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter) throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
    public static void main(String[] args) throws IOException {
        JobConf lp = new JobConf(MRExample.class);
        lp.setJobName("Load Pages");
        lp.setInputFormat(TextInputFormat.class);
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
            new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.addInputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
            new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf(MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMapper.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRExample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFileOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputFormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
        jc.addJob(limit);
        jc.run();
    }
}
```

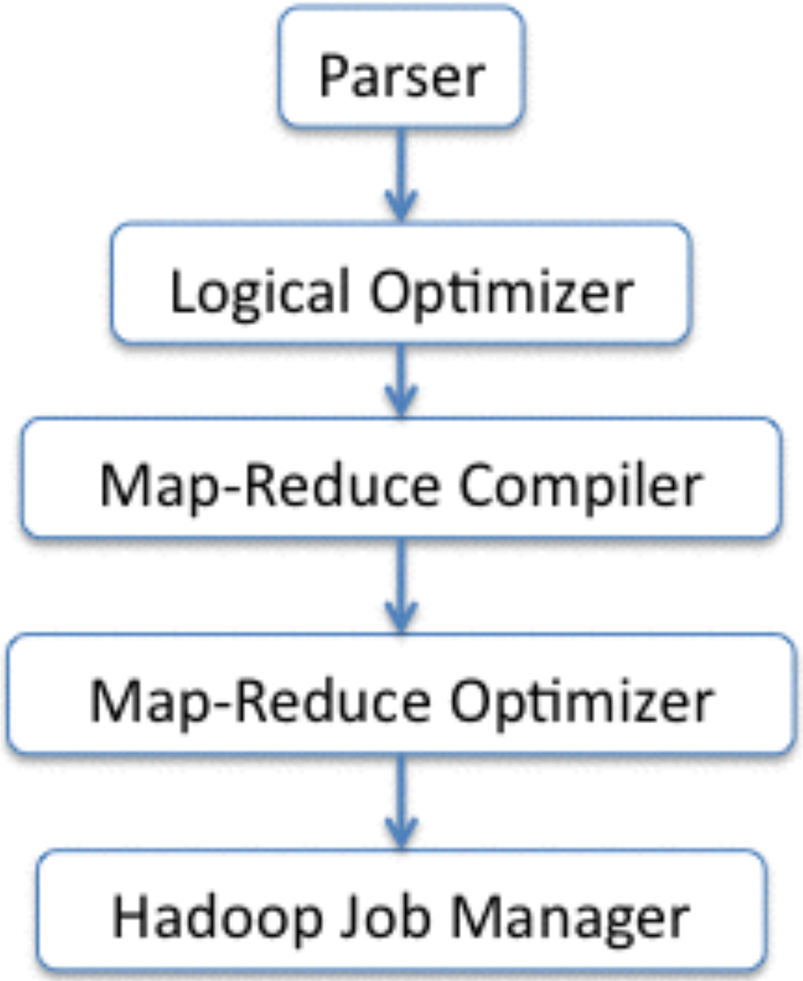
Pig takes your program and compiles it into a Hadoop job.

Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience

Alan F. Gates, Olga Natkovich, Shubham Chopra, Pradeep Kamath, Shravan M. Narayanamurthy, Christopher Olston, Benjamin Reed, Santhosh Srinivasan, Utkarsh Srivastava

Yahoo!, Inc.*

```
urls = LOAD ‘dataset’ AS (url, category, pagerank);
groups = GROUP urls BY category;
bigGroups = FILTER groups BY COUNT(urls)>1000000;
result = FOREACH bigGroups GENERATE
             group, top10(urls);
STORE result INTO ‘myOutput’;
```



Pig builds a "data flow" model from your program.

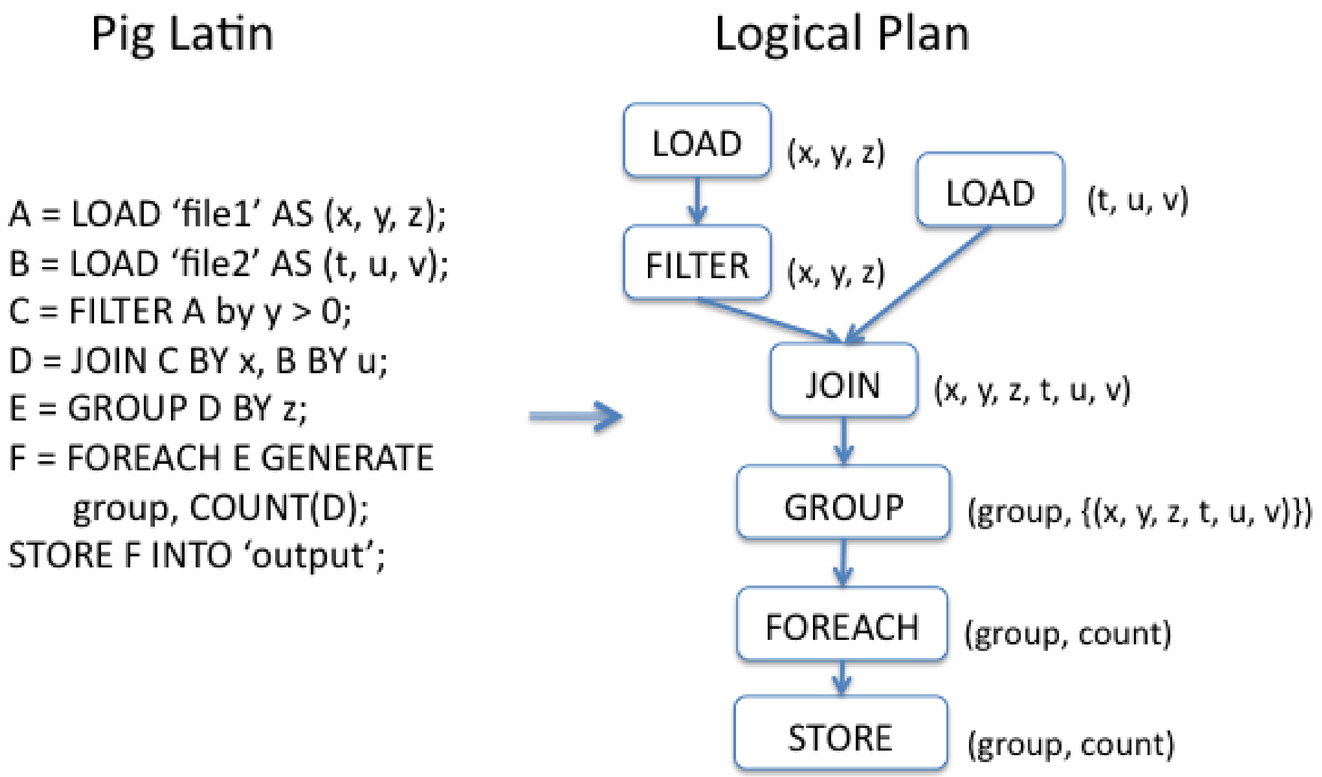
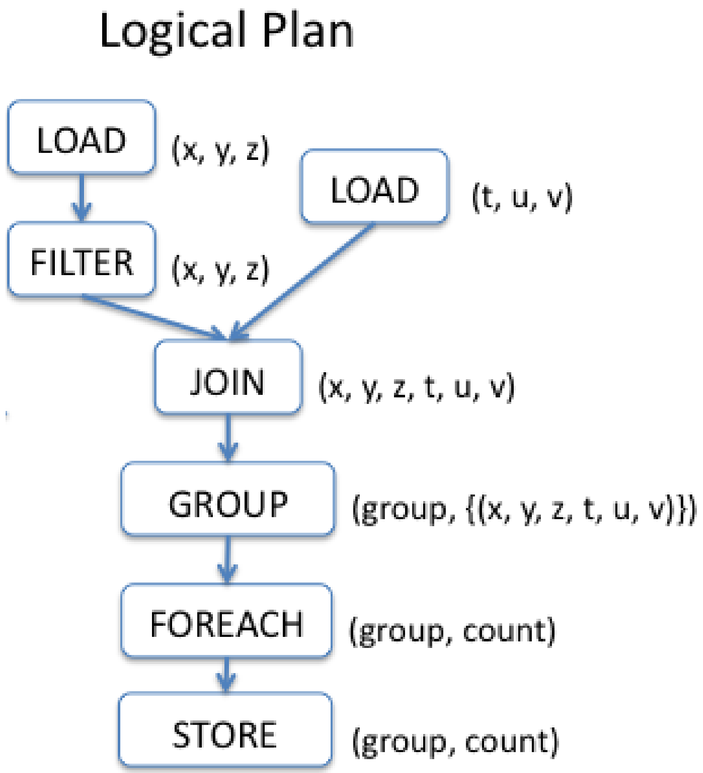
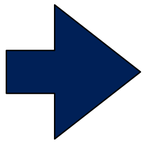
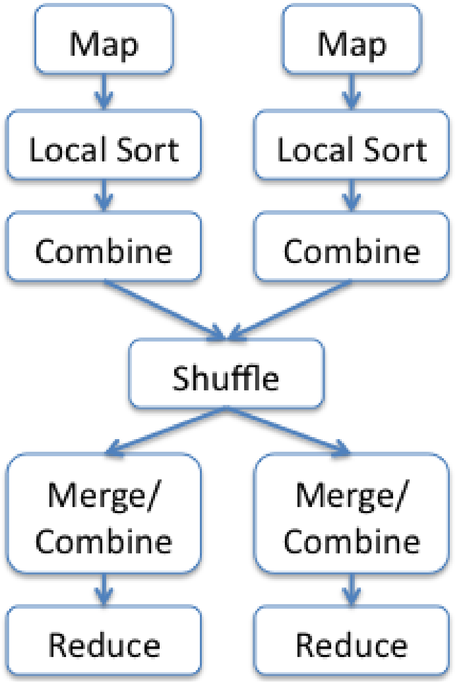


Figure 2: Pig Latin to logical plan translation.

The data flow is translated into a series of MapReduce steps.

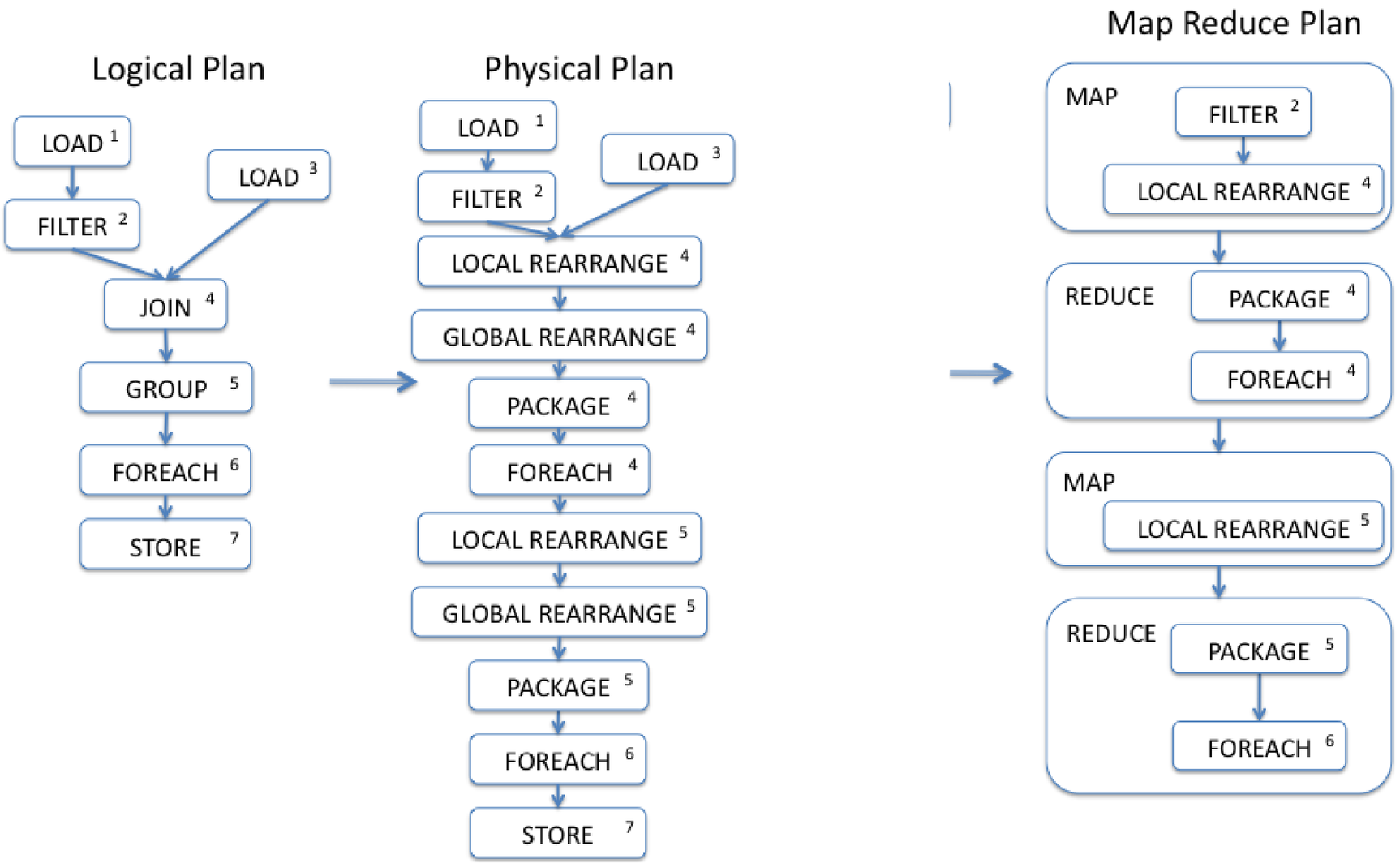






Pig Latin to logical plan translation.

Which are translated to an efficient Map Reduce Plan.



Pig Latin Program — Basic Program Design

Basic Pig Latin program:

- LOAD data from a file system (HDFS or S3)
- Transform the data.
- STORE to file system or DUMP to output.

Pig Data Loading Functions:

- A = LOAD *filename* [USING function] [AS schema];

e.g.:

- A = LOAD 'file';
- A = LOAD *filename* USING BinStorage();
- A = LOAD *filename* USING PigStorage(*field_delimiter)*;
- A = LOAD *filename* USING PigStorage() AS (*field_desc);*

Pig Latin Program — Basic Program Design

Basic Pig Latin program:

- LOAD data from a file system (HDFS or S3)
- Transform the data.
- STORE to file system or DUMP to output.

Pig transformation examples:

- FILTER

```
B = FILTER A BY $1 == 1;
B = FILTER A BY date == “1980-01-01”;
B = FILTER A BY $1 > 50;
```

- ORDER BY

```
C = ORDER B BY $0;
C = ORDER B BY date;
```

- LIMIT

```
D = LIMIT B 30;
```

- JOIN

```
D = JOIN C BY $1, B BY $1;
D = JOIN C BY ipaddress, D BY ipaddress;
```

Pig Latin Program — Basic Program Design

Basic Pig Latin program:

- LOAD data from a file system (HDFS or S3)
- Transform the data.
- STORE to file system or DUMP to output.

Pig Storage examples:

- STORE

```
STORE A INTO 'outputfile';
STORE A INTO 'outputfile.gz';


-- Store UTF-8:
STORE A INTO 'output' USING PigDump();

-- Store in Binary
STORE A INTO 'output' USING BinStorage();

--- Store with delimiters:
STORE A INTO 'output' USING PigStorage('*');
```

Pig can run locally or on MapReduce

Which version am I running?

```
$ pig -help
```

Pig modes of operation:

Warning: EMR has problems with pig -x local

	Local Mode	MapReduce Mode
Interactive	`$ pig -x local`	`$ pig -x mapreduce`
Batch	`$ pig -x local` *`filename.pig`*	`$ pig -x mapreduce` *`filename.pig`*

Pig Latin statements work with relations.
A = LOAD 'foo.txt' A is a *relation*.

A relation is a "bag."

- A **bag** is a collection of **tuples**.
- A **tuple** is an ordered set of **fields**
- A **field** is a piece of **data**.

Pig Data Types:

- Scalar types: **int, long, double, chararray**
- map — An “associative array” (like a python dictionary)

```
chararray : anytype
```

  —*e.g.*

```
“first” : “George”
“last”  : “Washington”
“born”  : 1732
```

- tuple

```
(v0, v1, v2, ...)
```

- bag — a collection of tuples

```
((a, b, c),
 (d, e, f),
 ...
)
```

Example (from reference guide)

```
A = LOAD 'student' USING PigStorage()
        AS (name:chararray, age:int, gpa:float);
DUMP A;
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
(Joe,18,3.8F)
```

	First Field	SecondField	Third Field
Data Type	chararray	int	float
Positional notation	$0	$1	$2
Possible name	name	age	gpa
Field value	John	18	4.0

It's best to use names!

Pig Latin FOREACH ... GENERATE

FOREACH ... GENERATE creates new relations from old ones.

Example (from reference guide):

```
A = LOAD 'student' USING PigStorage()
AS (name:chararray, age:int, gpa:float);
DUMP A;
(John,18,4.0F)
(Mary,19,3.8F)
(Bill,20,3.9F)
(Joe,18,3.8F)


X = FOREACH A GENERATE name,$2;
DUMP X;
(John,4.0F)
(Mary,3.8F)
(Bill,3.9F)
(Joe,3.8F)
```

Simple data types:

Simple Data Types	Description	Example
Scalars		
int	Signed 32-bit integer	10
long	Signed 64-bit integer	Data: 10L or 10l Display: 10L
float	32-bit floating point	Data: 10.5F or 10.5f or 10.5e2f or 10.5E2F Display: 10.5F or 1050.0F
double	64-bit floating point	Data: 10.5 or 10.5e2 or 10.5E2 Display: 10.5 or 1050.0
Arrays		
chararray	Character array (string) in Unicode UTF-8 format	hello world
bytearray	Byte array (blob)	

Pig is a complete data flow programming language

Functions:

- +, -, *, /, %,

NULL:

- Operations can return NULL; NULL is ignored by AVG(), MIN(), MAX(), SUM(), COUNT()

Conditions:

- ==, !=, >, <, >=, <=

Conditionals:

- NO IF STATEMENT!
- *conditional* ? *if-true* : *if-false*

Example from Pig Latin Reference Manual:

```
A = LOAD 'data' AS (f1:int, f2:int, :bag{T:tuple(t1:int,t2:int)});
DUMP A;
(10,1,{(2,3),(4,6)})
(10,3,{(2,3),(4,6)})
(10,6,{(2,3),(4,6),(5,7)})

X = FOREACH A GENERATE f1, f2, f1%f2;
DUMP X;
(10,1,0)
(10,3,1)
(10,6,4)

X = FOREACH A GENERATE f2, (f2==1?1:COUNT(B));
DUMP X;
(1,1L)
(3,2L)
(6,3L)
```

Word Count with Pig

```
lines     = LOAD 's3://gu-anly502/ps02/tobe.txt' as (line:chararray);
words     = FOREACH lines generate flatten(TOKENIZE(line)) as word;
grouped   = GROUP words by word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
dump wordcount;
```

LOAD — Loads the data

FOREACH — TOKENIZEs each line. Creates a "words" alias where each tuple is a "word"

GROUP — combines words that have the same word

FOREACH — counts the number of words in each group.

DUMP — sends to standard output.

Note:

- Put spaces around the equals sign (=) !
- Most Pig words are case-sensitive. (Exception: built-in statements like LOAD, FOREACH, GROUP and GENERATE).

grunt> — the Pig command line

```
grunt> help
Commands:
<pig latin statement>; - See the PigLatin manual for details: http://hadoop.apache.org/pig
File system commands:
    fs <fs arguments> - Equivalent to Hadoop dfs command: http://hadoop.apache.org/common/docs/current/hdfs_shell.html
Diagnostic commands:
    describe <alias>[::<alias] - Show the schema for the alias. Inner aliases can be described as A::B.
    explain [-script <pigscript>] [-out <path>] [-brief] [-dot|-xml] [-param <param_name>=<param_value>]
        [-param_file <file_name>] [<alias>] - Show the execution plan to compute the alias or for entire script.
        -script - Explain the entire script.
        -out - Store the output into directory rather than print to stdout.
        -brief - Don't expand nested plans (presenting a smaller graph for overview).
        -dot - Generate the output in .dot format. Default is text format.
        -xml - Generate the output in .xml format. Default is text format.
        -param <param_name - See parameter substitution for details.
        -param_file <file_name> - See parameter substitution for details.
        alias - Alias to explain.
    dump <alias> - Compute the alias and writes the results to stdout.
Utility Commands:
    exec [-param <param_name>=param_value] [-param_file <file_name>] <script> -
        Execute the script with access to grunt environment including aliases.
        -param <param_name - See parameter substitution for details.
        -param_file <file_name> - See parameter substitution for details.
        script - Script to be executed.
    run [-param <param_name>=param_value] [-param_file <file_name>] <script> -
        Execute the script with access to grunt environment.
        -param <param_name - See parameter substitution for details.
        -param_file <file_name> - See parameter substitution for details.
        script - Script to be executed.
    sh  <shell command> - Invoke a shell command.
    kill <job_id> - Kill the hadoop job specified by the hadoop job id.
    set <key> <value> - Provide execution parameters to Pig. Keys and values are case sensitive.
        The following keys are supported:
        default_parallel - Script-level reduce parallelism. Basic input size heuristics used by default.
        debug - Set debug on or off. Default is off.
        job.name - Single-quoted name for jobs. Default is PigLatin:<script name>
        job.priority - Priority for jobs. Values: very_low, low, normal, high, very_high. Default is normal
        stream.skippath - String that contains the path. This is used by streaming.
        any hadoop property.
    help - Display this message.
    history [-n] - Display the list statements in cache.
        -n Hide line numbers.
    quit - Quit the grunt shell.
grunt>
```

Always ask for "help"

Always read the documentation

Grunt supports many Unix commands: ls, cat,

```
grunt> ls s3://gu-anly502/
16/02/15 15:48:52 INFO s3n.S3NativeFileSystem: listStatus s3://gu-anly502/ with recursive false
s3://gu-anly502/bootstrap.sh<r 1>  936
s3://gu-anly502/gutenberg  <dir>
s3://gu-anly502/ps02       <dir>
s3://gu-anly502/ps03       <dir>
s3://gu-anly502/ps04       <dir>
grunt>

grunt> ls s3://gu-anly502/ps02/
16/02/15 15:49:01 INFO s3n.S3NativeFileSystem: listStatus s3://gu-anly502/ps02 with recursive
false
s3://gu-anly502/ps02/hamlet.txt<r 1>       1644
s3://gu-anly502/ps02/tobe.txt<r 1> 43
grunt>

grunt> cat s3://gu-anly502/ps02/tobe.txt
16/02/15 15:49:05 INFO s3n.S3NativeFileSystem: Opening 's3://gu-anly502/ps02/tobe.txt' for
reading
To be, or not to be- that is the question:
grunt>
```

To minimize Pig output — lower the warning level

Pig uses log4j to log. Make a copy of the existing log4j.properties file and edit it:

```
$ cp /etc/pig/conf.dist/log4j.properties log4j_WARN
```

—*set these lines:*

```
# ***** Set root logger level to DEBUG and its only appender to A.
log4j.rootLogger=ERROR, A
log4j.logger.org.apache.pig=warn,A
log4j.logger.org.apache.hadoop=warn,A
```

When you run pig, type:

```
$ pig -4 log4j_WARN
```

Hadoop Word Count in Pig

```
$ pig -4 log4j_WARN
grunt> lines = load 's3://gu-anly502/ps02/tobe.txt' as (line:chararray);
...
grunt> dump lines;
...
(To be, or not to be- )
(that is the question:)
grunt>
...
grunt> words = FOREACH lines generate flatten(TOKENIZE(line)) as word;
grunt> grouped = GROUP words by word;
grunt> wordcount = FOREACH grouped GENERATE group, COUNT(words);
grunt> dump wordcount;
68560 [JobControl] WARN  org.apache.hadoop.mapreduce.JobResourceUploader  - No job jar file set.  User classes may not be found.
See Job or Job#setJar(String).
68560 [JobControl] WARN  org.apache.hadoop.mapreduce.JobResourceUploader  - No job jar file set.  User classes may not be found.
See Job or Job#setJar(String).
68934 [DataStreamer for file /tmp/hadoop-yarn/staging/hadoop/.staging/job_1455488005182_0020/job.xml block
BP-1229375385-172.31.42.104-1455487984302:blk_1073742532_7091] INFO  amazon.emr.metrics.MetricsSaver  - 1 aggregated HDFSWriteDelay
113 raw values into 1 aggregated values, total 1
(To,1)
(be,1)
(is,1)
(or,1)
(to,1)
(be-,1)
(not,1)
(the,1)
(that,1)
(question:,1)
grunt>
```

Sorting the output…

```
grunt> dump wordcount;
68560 [JobControl] WARN  org.apache.hadoop.mapreduce.JobResourceUploader  - No job jar file set.  User classes may not be found.
See Job or Job#setJar(String).
68560 [JobControl] WARN  org.apache.hadoop.mapreduce.JobResourceUploader  - No job jar file set.  User classes may not be found.
See Job or Job#setJar(String).
68934 [DataStreamer for file /tmp/hadoop-yarn/staging/hadoop/.staging/job_1455488005182_0020/job.xml block
BP-1229375385-172.31.42.104-1455487984302:blk_1073742532_7091] INFO  amazon.emr.metrics.MetricsSaver  - 1 aggregated HDFSWriteDelay
113 raw values into 1 aggregated values, total 1
(To,1)
(be,1)
(is,1)
(or,1)
(to,1)
(be-,1)
(not,1)
(the,1)
(that,1)
(question:,1)

grunt> sorted_wordcount = ORDER wordcount by $0;
grunt> dump sorted_wordcount;
(To,1)
(be,1)
(be-,1)
(is,1)
(not,1)
(or,1)
(question:,1)
(that,1)
(the,1)
(to,1)
```

Working with a larger data set — use LIMIT to limit output.

```
grunt> hamlet = LOAD 's3://gu-anly502/ps02/hamlet.txt' AS (line:chararray);
grunt> words = foreach hamlet generate flatten(TOKENIZE(line)) as word;
grunt> grouped = GROUP words by word;
grunt> wordcount = FOREACH grouped GENERATE group, COUNT(words);
grunt> sorted_words = ORDER wordcount BY $1 DESC;
grunt> sorted_words20 = limit sorted_words 20;
grunt> dump sorted_words20;
(of,14)
(the,14)
(to,9)
(and,7)
(The,6)
(a,5)
(To,5)
(And,5)
(that,4)
(we,4)
(bear,3)
(That,3)
(us,3)
(in,3)
(make,2)
(end,2)
(makes,2)
(all,2)
(For,2)
(have,2)
grunt>
```

Pig Latin scripts can be put in files and run from the command line (like mrjob).

```
$ cat top20.pig
hamlet = LOAD 's3://gu-anly502/ps02/hamlet.txt' AS (line:chararray);
words = foreach hamlet generate flatten(TOKENIZE(line)) as word;
grouped = GROUP words by word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
sorted_words = ORDER wordcount BY $1 DESC;
sorted_words20 = limit sorted_words 20;
dump sorted_words20;
quit;
$ pig top20.pig -stop-on-failure
...
(of,14)
(the,14)
(to,9)
(and,7)
(The,6)
(a,5)
(To,5)
(And,5)
(that,4)
(we,4)
(bear,3)
(That,3)
(us,3)
(in,3)
(make,2)
(end,2)
(makes,2)
(all,2)
(For,2)
(have,2)
$
```

-stop-on-failure is recommended

Pig Status — don't just ignore it.
Use *store lines into 'outputfile';* to write output to a file.

```
4064342 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.SimplePigStats  - Script Statistics:

HadoopVersion  PigVersion UserId     StartedAt  FinishedAt Features
2.7.1-amzn-0   0.14.0-amzn-0         hadoop     2016-02-15 17:10:13  2016-02-15 17:10:34  UNKNOWN

Success!

Job Stats (time in seconds):
JobId           Maps       Reduces    MaxMapTime MinMapTime AvgMapTime MedianMapTime         MaxReduceTime         MinReduceTime
     AvgReduceTime         MedianReducetime     Alias      Feature    Outputs
job_1455488005182_0036    1          0          6          6          6          6          0          0          0          0
     lines      MAP_ONLY   hdfs://ip-172-31-42-104.ec2.internal:8020/user/hadoop/outputfile,

Input(s):
Successfully read 2 records (356 bytes) from: "s3://gu-anly502/ps02/tobe.txt"

Output(s):
Successfully stored 2 records (44 bytes) in: "hdfs://ip-172-31-42-104.ec2.internal:8020/user/hadoop/outputfile"

Counters:
Total records written : 2
Total bytes written : 44
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1455488005182_0036


...
```

```
16/02/15 17:10:34 INFO mapreduce.SimplePigStats: Script Statistics:

HadoopVersion  PigVersion UserId     StartedAt  FinishedAt Features
2.7.1-amzn-0   0.14.0-amzn-0         hadoop     2016-02-15 17:10:13  2016-02-15 17:10:34  UNKNOWN

Success!

Job Stats (time in seconds):
JobId           Maps       Reduces    MaxMapTime MinMapTime AvgMapTime MedianMapTime         MaxReduceTime         MinReduceTime
     AvgReduceTime         MedianReducetime     Alias      Feature    Outputs
job_1455488005182_0036    1          0          6          6          6          6          0          0          0          0
     lines      MAP_ONLY   hdfs://ip-172-31-42-104.ec2.internal:8020/user/hadoop/outputfile,

Input(s):
Successfully read 2 records (356 bytes) from: "s3://gu-anly502/ps02/tobe.txt"

Output(s):
Successfully stored 2 records (44 bytes) in: "hdfs://ip-172-31-42-104.ec2.internal:8020/user/hadoop/outputfile"

Counters:
Total records written : 2
Total bytes written : 44
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1455488005182_0036
```

```
grunt> cat hdfs:///user/hadoop/outputfile
cat hdfs:///user/hadoop/outputfile
To be, or not to be–
that is the question:
grunt>
```

Grunt built-in commands:

```
Was expecting one of:
    <EOF>
    "cat" ...
    "clear" ...
    "fs" ...
    "sh" ...
    "cd" ...
    "cp" ...
    "copyFromLocal" ...
    "copyToLocal" ...
    "dump" ...
    "\\d" ...
    "describe" ...
    "\\de" ...
    "aliases" ...
    "explain" ...
    "\\e" ...
    "help" ...
    "history" ...
    "kill" ...
    "ls" ...
    "mv" ...
    "mkdir" ...
    "pwd" ...
    "quit" ...
    "\\q" ...
    "register" ...
    "rm" ...
    "rmf" ...
    "set" ...
    "illustrate" ...
    "\\i" ...
    "run" ...
    "exec" ...
    "scriptDone" ...
    "" ...
    "" ...
    <EOL> ...
    ";" ...
```

Describe and Illustrate show the structure of relations.

```
grunt> describe lines
describe lines
16/02/15 17:14:10 INFO Configuration.deprecation:
fs.default.name is deprecated. Instead, use fs.defaultFS
lines: {line: chararray}

grunt> illustrate lines;
-------------------------------------
| lines     | line:chararray        |
-------------------------------------
|           | that is the question: |
-------------------------------------
grunt>
```

Pig User Defined Functions (UDFs)

UDFs expand Pig's functionality.

- Parse input lines
- Perform complex operations.
- Example — a UDF could search the MaxMind IP address geolocation database
  —*provided that the database is on each node.*

Coding Options:

- Write in Java — import as registered jar files.
- Write in jython — (Python that generates jar files) — import as registered jar files.
- Write in python — Access with "pig streaming API" (similar to Hadoop streaming)

Pig can process any tab-delimited data.
How do you process data that aren't tab-delimited? (e.g. Apache log files)

Piggybank — a collection of algorithms for pig.

- CommonLogLoader —
  - https://pig.apache.org/docs/r0.14.0/api/org/apache/pig/piggybank/storage/apachelog/CommonLogLoader.html
- CombinedLogLoader:
  - https://pig.apache.org/docs/r0.14.0/api/org/apache/pig/piggybank/storage/apachelog/CombinedLogLoader.html

```
raw = LOAD 'combined_log' USING org.apache.pig.piggybank.storage.apachelog.CombinedLogLoader AS (remoteAddr, remoteLogname, user,
time, method, uri, proto, status, bytes, referer, userAgent);
```

- Note: I was not able to get CombinedLogLoader to work with the ForensicsWiki logs!

I used REGEX_EXTRACT to extract the log file entries:

```
logs_base =
  FOREACH
   raw_logs
  GENERATE
   FLATTEN ( EXTRACT( line,
     '^(\\S+) (\\S+) (\\S+) \\[([\\w/]+):(\\d{2}:\\d{2}:\\d{2}) [+\\-]\\d{4}\\] "(\\S+) (\\S+) \\S+" (\\S+) (\\S+) "([^"]*)"
"([^"]*)"'
     ) ) AS (
     host: chararray, identity: chararray, user: chararray, date: chararray, time: chararray, verb: chararray, url: chararray,
request: chararray, status: int,
     size: chararray, referrer: chararray, agent: chararray
     );
```

Pig program to produce hits-by-day

```
DEFINE EXTRACT       org.apache.pig.piggybank.evaluation.string.EXTRACT();

raw_logs = load 's3://gu-anly502/ps03/forensicswiki.2012.txt' as (line:chararray);

logs_base =
  FOREACH raw_logs GENERATE FLATTEN (
     EXTRACT( line,
     '^(\\S+) (\\S+) (\\S+) \\[([\\w/]+):(\\d{2}:\\d{2}:\\d{2}) [+\\-]\\d{4}\\] "(\\S+) (\\S+) \\S+" (\\S+) (\\S+) "([^"]*)"
"([^"]*)"'
     ) ) AS (
     host: chararray, identity: chararray, user: chararray, date: chararray, time: chararray, verb: chararray, url: chararray,
request: chararray, status: int,
     size: chararray, referrer: chararray, agent: chararray
     );

by_date = GROUP logs_base BY (date);

date_counts = FOREACH by_date GENERATE
    group as date,      -- the key you grouped on
    COUNT(logs_base);   -- the number of log lines wiht this date

dump date_counts;
```

Pig output

```
$ pig parse_apache.pig
16/02/21 20:18:47 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
16/02/21 20:18:47 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
16/02/21 20:18:47 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
45   [main] INFO  org.apache.pig.Main  - Apache Pig version 0.14.0-amzn-0 (r: unknown) compiled Jan 14 2016, 02:55:53
16/02/21 20:18:47 INFO pig.Main: Apache Pig version 0.14.0-amzn-0 (r: unknown) compiled Jan 14 2016, 02:55:53
...
16/02/21 20:23:09 INFO util.MapRedUtil: Total input paths to process : 5
(01/Jul/2012,35039)
(01/Sep/2012,33272)
(02/Jul/2012,46445)
(02/Sep/2012,36225)
(03/Jul/2012,43922)
(03/Sep/2012,40703)
(04/Jul/2012,38576)
...
(30/Jul/2012,45488)
(30/Sep/2012,37817)
(31/Jul/2012,48353)
263298 [main] INFO  org.apache.pig.Main  - Pig script completed in 4 minutes, 23 seconds and 386 milliseconds (263386 ms)
16/02/21 20:23:10 INFO pig.Main: Pig script completed in 4 minutes, 23 seconds and 386 milliseconds (263386 ms)

[20:23:11 last: 266s][~/ANLY502/L05]
$
```

266 seconds to process 4GB file!

Parse the date as a "datetime" and create a new relation with just the desired fields.

Old regular expression:

```
logs_base =
  FOREACH raw_logs GENERATE FLATTEN (
     EXTRACT( line,
     '^(\\S+) (\\S+) (\\S+) \\[([\\w/]+):(\\d{2}:\\d{2}:\\d{2}) [+\\-]\\d{4}\\] "(\\S+) (\\S+) \\S+" (\\S+) (\\S+) "([^"]*)"
"([^"]*)"'
     ) ) AS (
     host: chararray, identity: chararray, user: chararray, date: chararray, time: chararray, verb: chararray, url: chararray,
request: chararray, status: int,
     size: chararray, referrer: chararray, agent: chararray);
```

New:

```
logs_base =
  FOREACH
   raw_logs
  GENERATE
   FLATTEN ( EXTRACT( line,
     '^(\\S+) (\\S+) (\\S+) \\[([^\\]]+)\\] "(\\S+) (\\S+) \\S+" (\\S+) (\\S+) "([^"]*)" "([^"]*)"'
     ) ) AS (
     host: chararray, identity: chararray, user: chararray, datetime_str: chararray, verb: chararray, url: chararray, request:
chararray, status: int,
     size: int, referrer: chararray, agent: chararray
     );

logs = FOREACH logs_base GENERATE ToDate(datetime_str,'dd/MMM/yyyy:HH:mm:ss Z') AS date, host, url, size;
```

"schema"

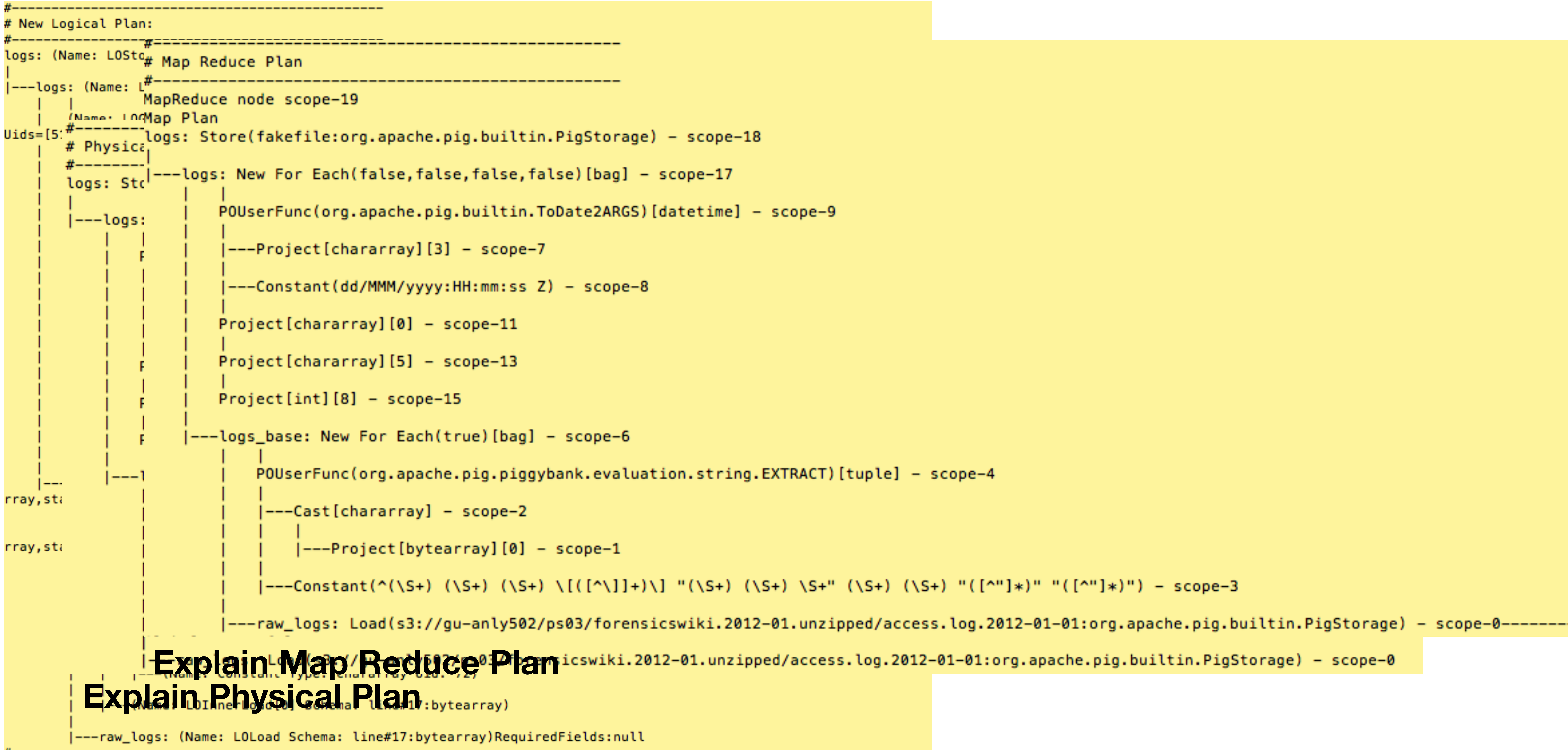
"describe" and "explain"

```
logs = FOREACH logs_base GENERATE ToDate(datetime_str,'dd/MMM/yyyy:HH:mm:ss Z') AS date, host, url, size;
```

Describe logs:

```
logs: {date:
```

Explain logs:



Explain "Logical Plan"

Final demo: list of forensicswiki hits by date:

Program:

```
raw_logs = load 's3://gu-anly502/ps03/forensicswiki.2012.txt' as (line:chararray);
logs_base =
  FOREACH
   raw_logs
  GENERATE
   FLATTEN ( EXTRACT( line,
     '^(\\S+) (\\S+) (\\S+) \\[([^\\]]+)\\] "(\\S+) (\\S+) \\S+" (\\S+) (\\S+) "([^"]*)" "([^"]*)"'
     ) ) AS (
     host: chararray, identity: chararray, user: chararray, datetime_str: chararray, verb: chararray, url: chararray, request: chararray,
status: int,
     size: int, referrer: chararray, agent: chararray
     );

by_date = GROUP logs BY (date);
date_counts = FOREACH by_date GENERATE
    group as date,      -- the key you grouped on
    COUNT(logs_base);   -- the number of log lines wiht this date
dump date_counts;
```

Output:

```
(,0)
(2012-01-01T00:00:00.000Z,29116)
(2012-01-02T00:00:00.000Z,38188)
...
(2012-12-31T00:00:00.000Z,36631)
(2013-01-01T00:00:00.000Z,1283)
329255 [main] INFO  org.apache.pig.Main  - Pig script completed in 5 minutes, 29 seconds and 337 milliseconds (329337 ms)
16/02/22 00:43:57 INFO pig.Main: Pig script completed in 5 minutes, 29 seconds and 337 milliseconds (329337 ms)

[00:43:58 last: 331s][~/ANLY502/L05]
$
```

331 seconds! (4x faster that mrjob)

A little cleaner — (2012-12-31T00:00:00.000Z,36631) -> (2012-12-31,36631)

Add a second GENERATE:

```
logs  = FOREACH logs_base GENERATE ToDate(SUBSTRING(datetime_str,0,11),'dd/MMM/yyyy') AS date, host, url, size;
logs2 = FOREACH logs      GENERATE SUBSTRING(ToString(date),0,10) AS date, host, url, size;

by_date = GROUP logs2 BY (date);
date_counts = FOREACH by_date GENERATE
    group AS date,    -- the key you grouped on
    COUNT(logs2);      -- the number of log lines wiht this date

date_counts_sorted = ORDER date_counts BY date;
dump date_counts_sorted;
```

And run...

```
(2012-12-28,39090)
(2012-12-29,54360)
(2012-12-30,40828)
(2012-12-31,36631)
(2013-01-01,1283)
368896 [main] INFO  org.apache.pig.Main  - Pig script completed in 6 minutes, 8 seconds and 977 milliseconds (368977 ms)
16/02/22 01:21:35 INFO pig.Main: Pig script completed in 6 minutes, 8 seconds and 977 milliseconds (368977 ms)
[hadoop@ip-172-31-37-188 L05]$ %
```

368 seconds (up from 331)

MaxMind Join with the Forensicswiki Data

```
DEFINE EXTRACT       org.apache.pig.piggybank.evaluation.string.EXTRACT();

raw_logs = load 's3://gu-anly502/ps03/forensicswiki.2012.txt' as (line:chararray);

maxmind  = load 's3://gu-anly502/ps03/maxmind' as (ipaddr:chararray, country:chararray);


logs_base =
  FOREACH
   raw_logs
  GENERATE
   FLATTEN ( EXTRACT( line,
     '^(\\S+) (\\S+) (\\S+) \\[([^\\]]+)\\] "(\\S+) (\\S+) \\S+" (\\S+) (\\S+) "([^"]*)" "([^"]*)"'
     ) ) AS (
     host: chararray, identity: chararray, user: chararray, datetime_str: chararray, verb: chararray, url: chararray, request:
chararray, status: int,
     size: int, referrer: chararray, agent: chararray
     );

geolocated_logs = JOIN logs_base BY host, maxmind BY ipaddr;
geolocated_50 = LIMIT geolocated_logs 50;
dump geolocated_50;
...
(180.76.5.67,-,-,01/Jan/2012:13:02:39 -0800,GET,/wiki/Special:WhatLinksHere/User_talk:Marc_Yu,200,3799,-,Mozilla/5.0 (compatible;
Baiduspider/2.0; +http://www.baidu.com/search/spider.html),180.76.5.67,China)
(180.76.5.89,-,-,01/Jan/2012:02:27:53 -0800,GET,/wiki/Special:RecentChangesLinked/Libvshadow,200,4391,-,Mozilla/5.0 (compatible;
Baiduspider/2.0; +http://www.baidu.com/search/spider.html),180.76.5.89,China)
(180.76.5.89,-,-,01/Jan/2012:21:47:55 -0800,GET,/images/7/79/?C=S;O=D,200,553,-,Mozilla/5.0 (compatible; Baiduspider/2.0; +http://
www.baidu.com/search/spider.html),180.76.5.89,China)
```

PS04 will involve doing the full join with the original maxmind data!

Next Week

Schedule for next two weeks

Mon Feb 22 — Today!

- L05 — Pig

Fri Feb 26th

- PS03a Due
  - *—let me know if you are having problems!*
  - *—Check the new source code at ANLY502/PS03*
  - *—Be careful about error checking. Massive data is always messy data.*

Mon Feb 29 — Next week

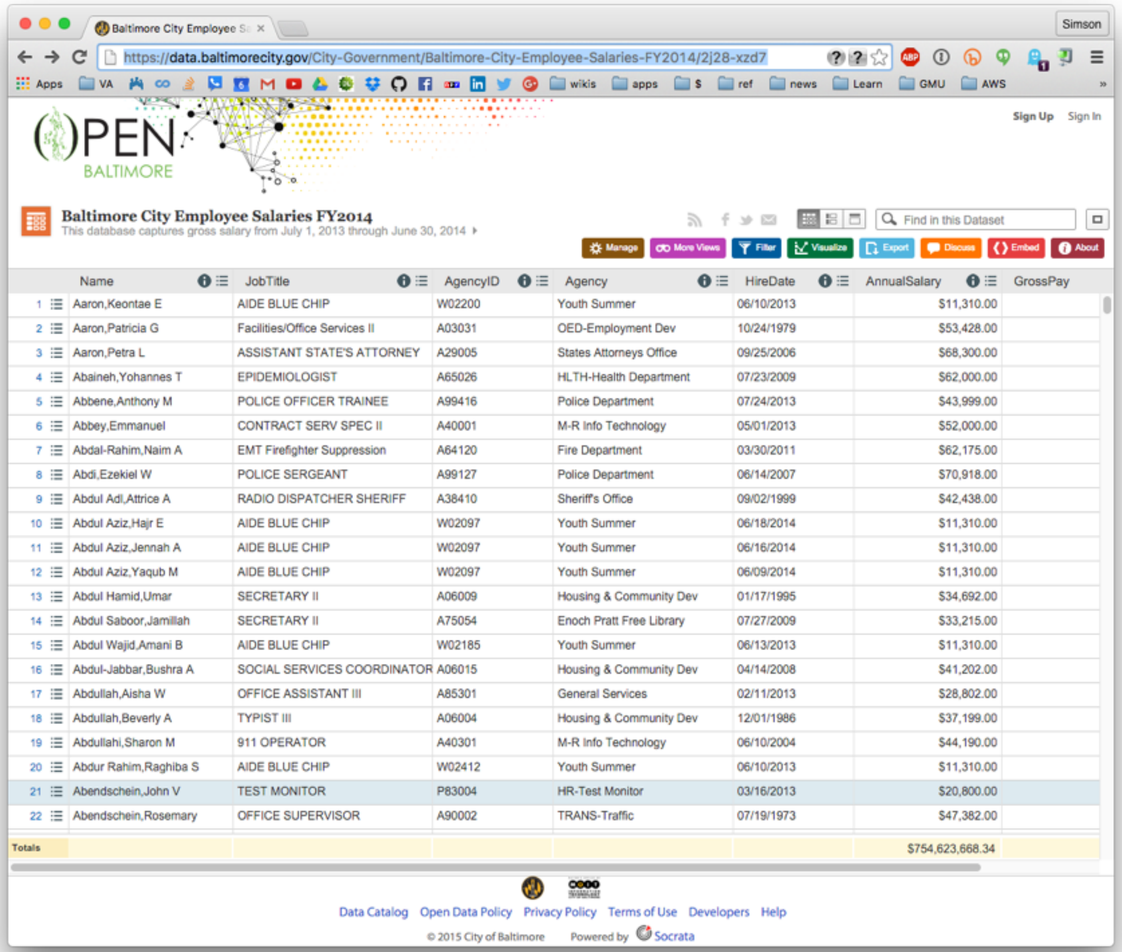
- L06 — Spark
- PS04 — Released — Pig & Spark

Fair Information Practice

Remember this data set from L02?

Source data: Baltimore City Employee Salaries FY2014

- https://data.baltimorecity.gov/City-Government/Baltimore-City-Employee-Salaries-FY2014/2j28-xzd7



Example from Donald Miner PyCon 2015 https://www.youtube.com/watch?v=b8HLYUp_fA8

This is a pretty creepy data set.

For every Baltimore employee, it has:

	Name	JobTitle	AgencyID	Agency	HireDate	AnnualSalary	GrossPay
1	Aaron,Keontae E	AIDE BLUE CHIP	W02200	Youth Summer	06/10/2013	$11,310.00	
2	Aaron,Patricia G	Facilities/Office Services II	A03031	OED-Employment Dev	10/24/1979	$53,428.00	
3	Aaron,Petra L	ASSISTANT STATE'S ATTORNEY	A29005	States Attorneys Office	09/25/2006	$68,300.00	

This is public information because these people are *city employees*.

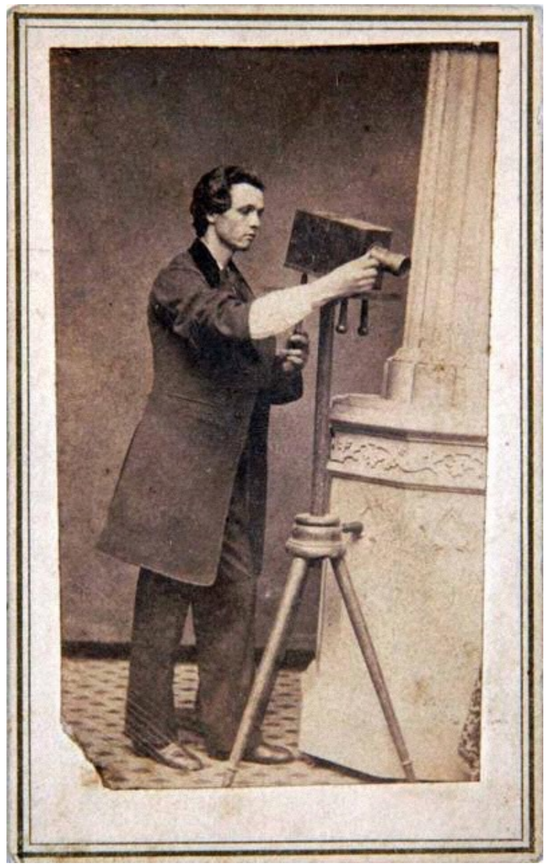
Issues:

- What's missing?
- What happens if these data are incorrect?

Example from Donald Miner PyCon 2015 https://www.youtube.com/watch?v=b8HLYUp_fA8

Modern concerns with privacy date back to the 1880s

Photography as we know it was invented in the 1860s.



http://www.antiquecameras.net/photographers18601900.html

In 1888 George Eastman's Kodak company invented the box camera with roll film

Suddenly cameras were portable, affordable, and easy-to-use.

- "You press the button — we do the rest."

1888 box camera

- https://en.wikipedia.org/wiki/Kodak

In 1890 Samuel Warren and Louis Brandeis identified technology as a primary threat to privacy.

"The Right to Privacy," Samuel D. Warren and Louis D. Brandeis,

- Harvard Law Review, Vol. IV, No. 5, December 15, 1890
- http://groups.csail.mit.edu/mac/classes/6.805/articles/privacy/Privacy_brand_warr2.html

Famously called privacy "the right to be let alone."*

- Said technology threatened to take "what is whispered in the closet" and have it "proclaimed from the house-tops."
- Key technologies of concern: photography & low-cost newspapers.
- Argued that Tort Law should be used to protect the right of privacy.

Today we see the article articulating four different kinds of privacy:

- Appropriation of a person's name or likeness.
- Intrusion into a person's seclusion or private affairs
- Disclosure of embarrassing private facts
- Publicity that places a person in false light

***Others had used the phrase before. See http://www.rbs2.com/privacy.htm**

1902: Roberson v. Rochester Folding Box Co.

Abigail Roberson posed for a drawing.

Franklin Mills Flour hired Rochester Folding Box Co. to make flour boxes.

- A lithograph of Roberson was put by Folding Box Co. on the container.
- "Flour of the Family"

Roberson sued for privacy invasion. Roberson lost!

- "It will be observed that there is no complaint made that plaintiff was libeled by this publication is said to be a very good one, and one that her friends and acquaintances were able to recogniz that a good portrait of her, and, [***10] therefore, one easily recognized, has been used to attra the paper upon which defendant mill company's advertisements appear."

1903: New York Legislature enacts a law giving people the right to sue for c their image without permission.

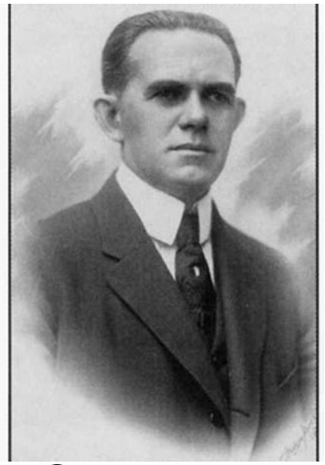
- http://faculty.uml.edu/sgallagher/Roberson.htm

Databanks also date to the 1890s

1899 — Retail Credit founded in Atlanta by Cator and Guy Woolford

- Created "Merchant Guide" for Atlanta Grocers
  - —*List of customers who paid and who didn't pay*
  - —*Sold to grocers for $25/year*
- Eagerly adopted computers in the 1960s
- Changed name to Equifax in 1979

Guy Woolford



Cator Woolford

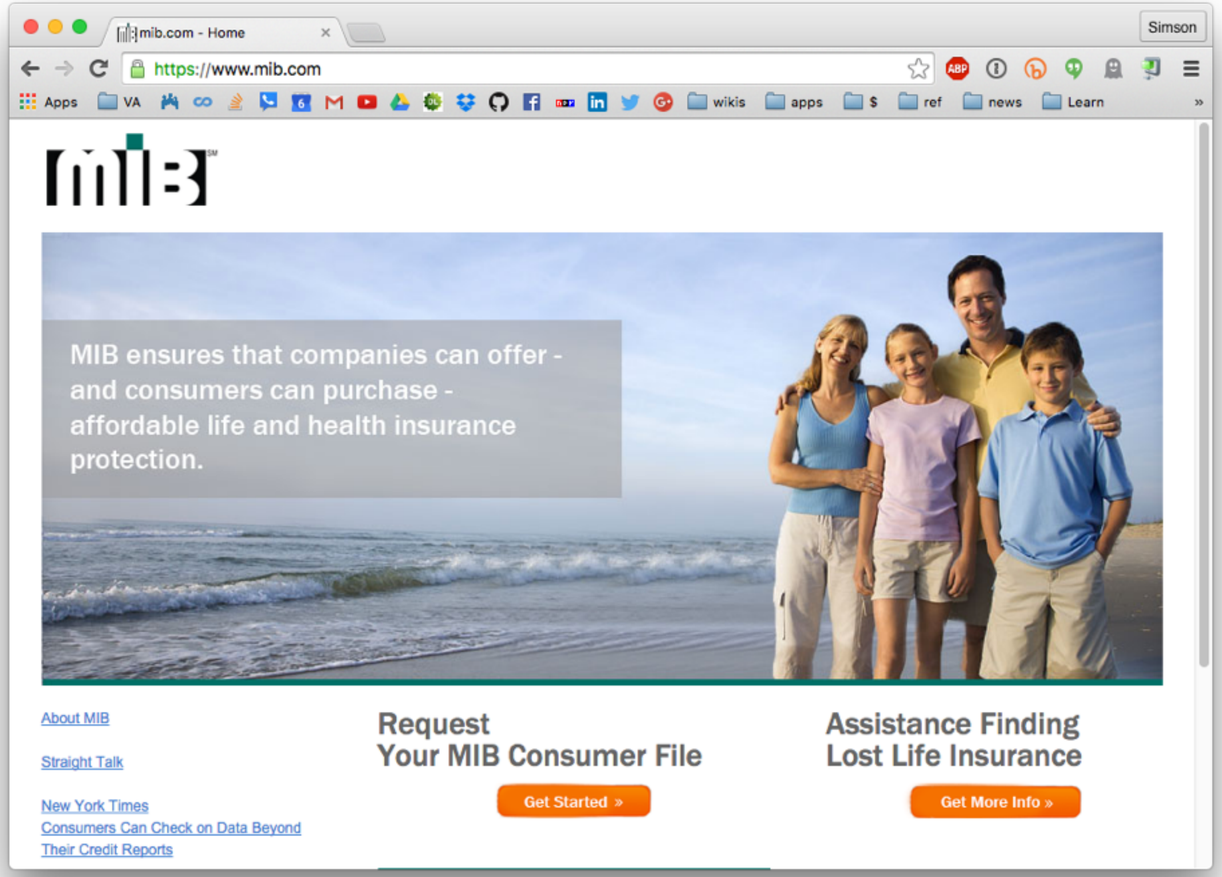
1902 — The Medical Information Burea is created in Boston
A cooperative organization for life insurance firms to share information.

Designed to prevent adverse selection:

- *—John is diagnosed with heart condition.*
- *—Buys a lot of life insurance from company X.*
- *—John dies.*
- *—Company X has big payout to John's family.*

Designed to prevent insurance shopping:

- *—John applies to company X for life insurance*
- *—John tells company X: I have a heart condition!*
- *—Company X denies life insurance.*
- *—John applies to Company Y, doesn't mention heart or X.*
- *—Company Y writes a policy for John.*
- *—John has a heart attack, dies.*



Private databanks were created for tracking reputation.

This is "The Bad People" problem.

- The world is filled with bad people.
- You can't put them all in prison.

Databases let businesses learn from the mistakes of others.

Retail Credit

- List of people "known" not to reply their debts

Medical Information Bureau (est. 1902)

- List of people with "known" medical problems

Chicago-area merchants (1950s)

- List of "known" shoplifters

Goals in tracking bad people.

Blacklisting — the original goal.

- Make a list of the bad people.
- Don’t do business with anybody on the list.

Reform

- Track ill-deeds and gave them a chance to make amends.
- "Penitentiary" as a place to "make penance."
- This was never the goal of private databanks.

By the 1960s, Credit report files were a mess.

Contained both “factual” and “investigative” information.

- Contained information that was hearsay or just plain wrong.

Records confused between individuals.

No “statute of limitations” on the information.

People frequently prohibited from seeing their own records.

1968: Congress held hearings on the database industry

Typical "Investigative" Credit Report, circa 1965:

- “Retired Army Lieutenant Colonel”
- “A rather wild-tempered, unreasonably, and uncouth person….
  “who abused his rank and wasn’t considered a well-adjusted person.
  “He was known to roam the reservation at Ft. Hood and shoot cattle belonging to ranchers who had leased the grazing land from the Army.”

  —*Retail Credit Co. of Atlanta, Ga : hearing before a subcommittee of the Committee on Government Operations, House of Representatives, Ninetieth Congress, second session. May 16, 1968 Hearings on the Retail Credit Company,*
  - http://lccn.loc.gov/71602454
  - KF27 .G665 1968a

"Supposedly confidential reports done by private investigative agencies for the FHA [Federal Housing Administration], on marital stability of FHA applicants, can be bought by private mortgage lenders for $1.50 each"

- Privacy and Freedom, Westin, (1970 edition), p. 160

Fair Credit Reporting Act, 1970

FCRA created five rights:

- Right to see your credit report.
- Right to challenge incorrect information.
- Right to have information expire.
  - —*Most information automatically removed from report after 7 years*
  - —*Bankruptcy information remains for 10 years*
- Right to know who accesses your report.
- Right to a free credit report if you are denied credit.

1973: Code of Fair Information Practice Practice.

Developed by the Department of Health, Education and Welfare following the passage of the FCRA

1. There must be no personal data record-keeping systems whose very existence is secret.
2. There must be a way for a person to find out what information about the person is in a record and how it is used.
3. There must be a way for a person to prevent information about the person that was obtained for one purpose from being used or made available for other purposes without the person's consent.
4. There must be a way for a person to correct or amend a record of identifiable information about the person.
5. Any organization creating, maintaining, using, or disseminating records of identifiable personal data must assure the reliability of the data for their intended use and must take precautions to prevent misuses of the data.

The Code of Fair Information Practice was never passed, but inspired many other laws.

US:

- Right to Financial Privacy Act (1970)
- Privacy Act of 1974 (5 USC §552a)
- Family Education Rights and Privacy Act (1974)
- Cable Communications Policy Act of 1984
- Video Privacy Protection Act of 1988 (18 USC 2710)
- Computer Matching and Privacy Protection Act of 1988 (PL 100-503)
- Telephone Consumer Protection Act of 1991
- Driver's Privacy Protection Act of 1994
- Health Insurance Portability and Accountability Act (1996)
- Children's Online Privacy Protection Act (1998)
- Gramm-Leach-Bliley (Final rule, May 24, 2000)
- Do-Not-Call Implementation Act of 2003

  - http://www.cdt.org/privacy/guide/protect/laws.php
  - http://epic.org/privacy/

Video Privacy Protection Act of 1988 (18 USC 2710)

Judge Robert Bork was nominated to serve on the Supreme Court by Ronald Regan

Washington DC's *City Paper* obtained Judge Bork's rental records from a local video store.

Congressmen realized that if Bork's records could be obtained, *anybody's records* could be obtained!

- Hearings were held in which many testified that rental records had been used in many divorce cases.
- The Cable Act had provided protection to pay-per-view records

Congress passed the Video Privacy Protection Act of 1988.

Driver's Privacy Protection Act of 1994

Actress Rebecca Schaeffer was murdered in 1989 by a crazed fan.

- The fan had been stalking her for three years.
- Schaeffer obtained a PO Box to hide her home address.
- The fan had hired a private detective to find Schaeffer's home address.
- The detective had bought the DMV records from State of California for $250.
  —*California required Schaeffer to provide her physical address to the DMV.*
- Five years later, Congress passed VPAA

Rebecca Schaeffer
1967 - 1989

Driver's Privacy Protection Act of 1994

DPPA limits what states can do with data:

- Motor vehicle or driver safety and theft; motor vehicle emissions; motor vehicle product alterations, recalls, or advisories; performance monitoring of motor vehicles and details; removal of non-owner records from original owner records.
- For use by any government agency, including any court or law enforcement agency, in carrying out its functions.
- For use in the normal course of business by a legitimate business, but only:
  - *To verify the accuracy of personal information submitted.*
  - *To correct information that's been submitted.*
- Research, provided that personal information is not published.
- Insurance; Providing notice to owners of impounded
- For use by any licensed private investigative agency or license security service "for any purpose permitted under this subsection."
- For use by employers to veirfy information relating to a commercial driver's license.
- Tolls; Surveys; Any other use authorized by state law.

In 2003, New Hampshire Supreme Court held investigation firms liable for the harm they cause for divulging personal information.

HIPAA:
Health Insurance Portability and Accountability Act of 1996*

Key Provisions:

- Largely about health insurance portability, not about privacy
- Privacy mandates are largely about security:
  - —*Firewalls, anti-virus, etc.*
  - —*Designate a privacy officer*
  - —*Post privacy policy*
  - —*Require outsourcing companies to protect information.*
  - —*Access to health information; procedures for correcting errors.*
- Enforced by the States (unfunded mandate); HHS enforces in “extreme cases.”

*privacy rule passed 2002

COPPA:
Children's Online Privacy Protection Act (1998).

Key Provisions:

- Applies to online collection information on children under 13
- Requires “verifiable parental consent”
  - —*Very hard in most cases; letter, fax or phone call*
  - —*Some exceptions — one time response to “homework help”*
- Privacy notice must be posted on website

http://www.ftc.gov/opa/1999/9910/childfinal.htm

Approaches to Privacy Enforcement

Unregulated Market

- Industry Standards (Voluntary)
- “Codes of conduct” — Limited enforcement through licensing
- Enforcement through “market forces;" limited enforcement from government

State and Federal Government

- Forcing companies to comply with their privacy policies.
  —*Federal Trade Commission Act of 1914 prohibits "unfair and deceptive trade practices."*
- Enforcement of privacy laws by regulatory agencies, states, etc.

Private Action

- Enforcement through private suit. (It's hard to prove damages.)