Case Study: Amazon AWS

CSE 40822 – Cloud Computing

Prof. Douglas Thain

University of Notre Dame

Caution to the Reader:

Herein are examples of prices consulted in fall 2018, to give a sense of the magnitude of costs. Do your own research before spending your own money!

Several Historical Trends

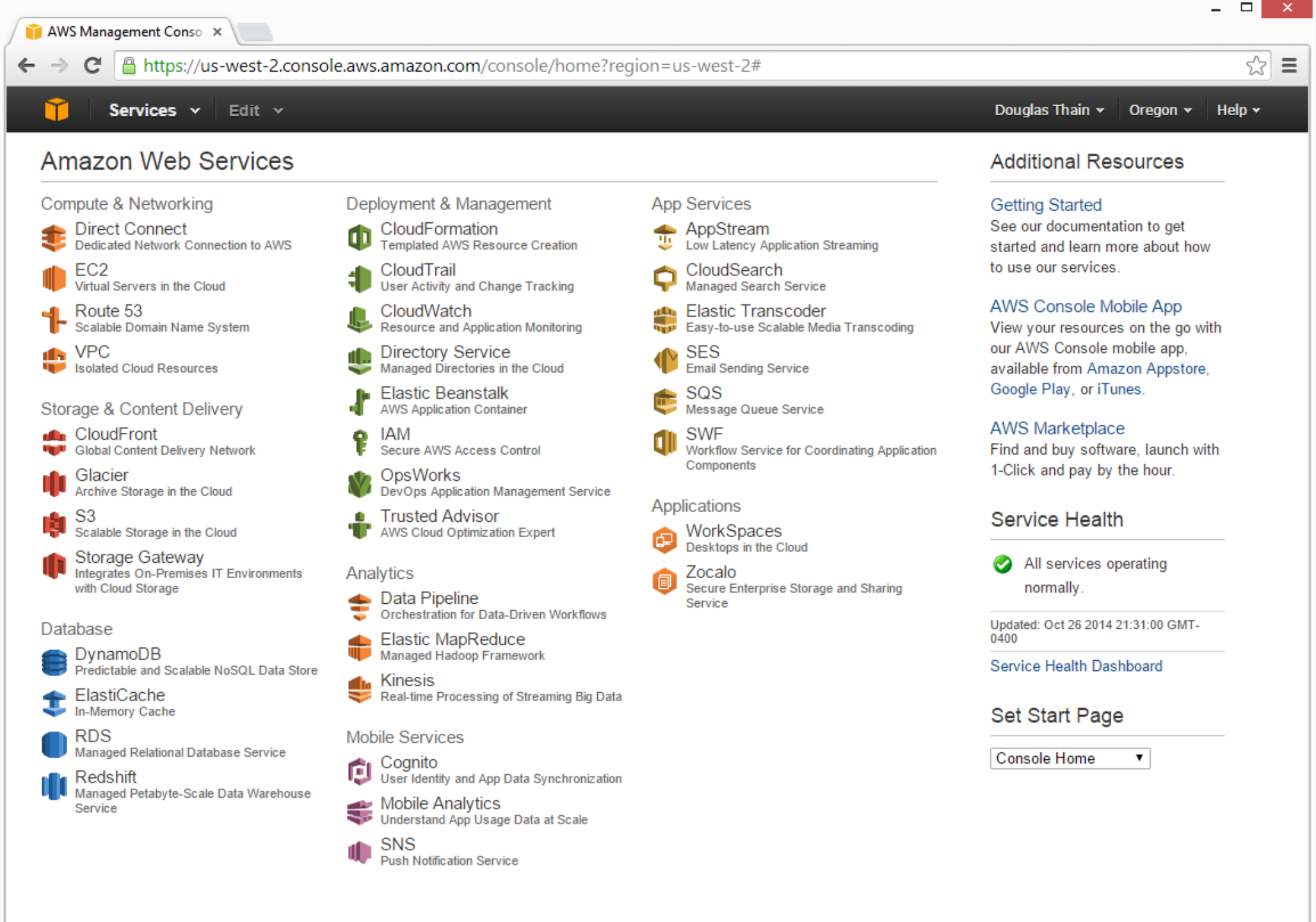
- Shared Utility Computing
  - 1960s – MULTICS – Concept of a Shared Computing Utility
  - 1970s – IBM Mainframes – rent by the CPU-hour. (Fast/slow switch.)
- Data Center Co-location
  - 1990s-2000s – Rent machines for months/years, keep them close to the network access point and pay a flat rate. Avoid running your own building with utilities!
- Pay as You Go
  - Early 2000s - Submit jobs to a remote service provider where they run on the raw hardware. Sun Cloud ($1/CPU-hour, Solaris +SGE) IBM Deep Capacity Computing on Demand (50 cents/hour)
- Virtualization
  - 1960s – OS-VM, VM-360 – Used to split mainframes into logical partitions.
  - 1998 – VMWare – First practical implementation on X86, but at significant performance hit.
  - 2003 – Xen paravirtualization provides much perf, but kernel must assist.
  - Late 2000s – Intel and AMD add hardware support for virtualization.

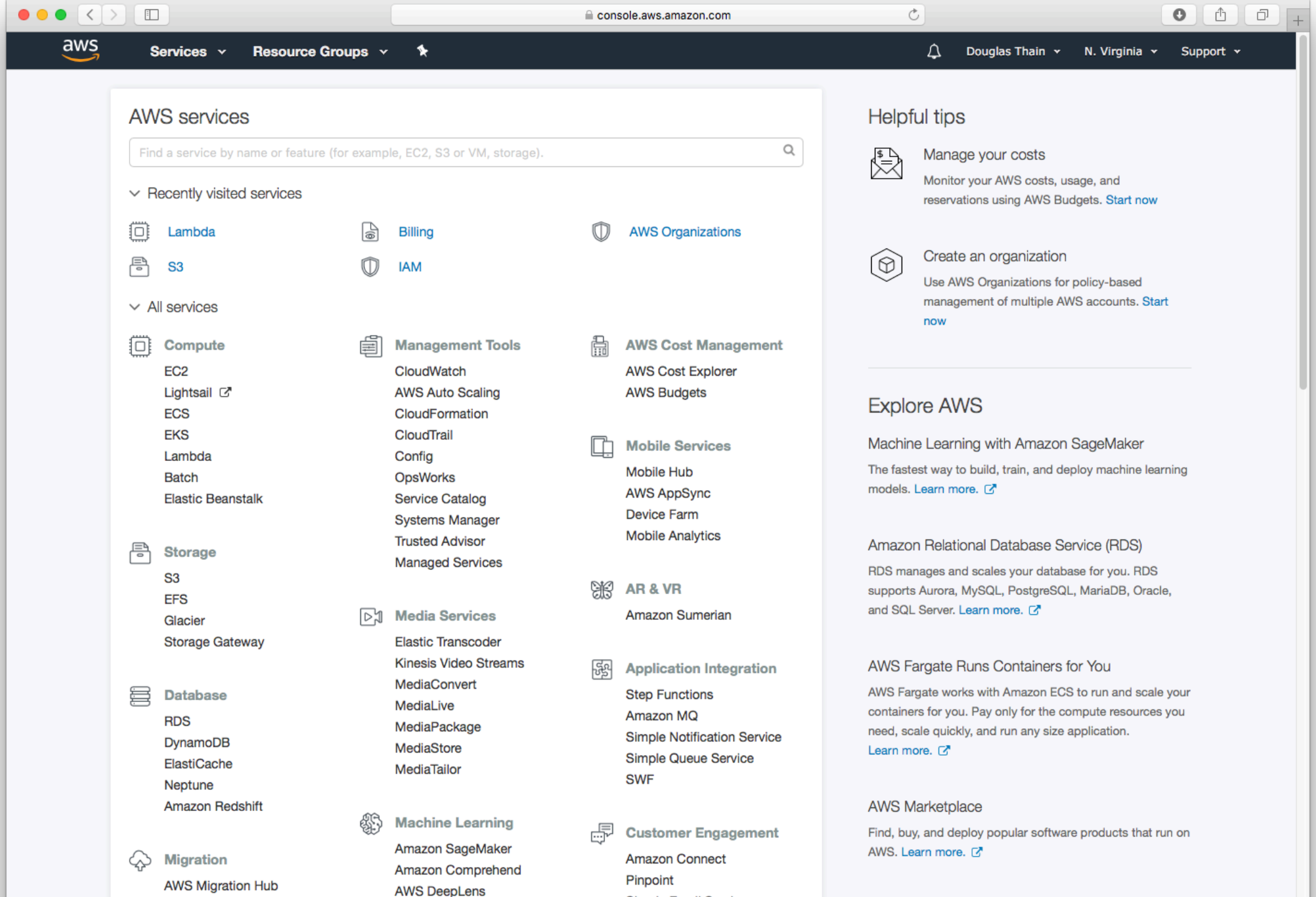
Virtual-* Allows for the Scale of Abstraction to Increase Over Time

- Run one process within certain resource limits.
  Op Sys has virtual memory, virtual CPU, and virtual storage (file system).
- Run multiple processes within certain resource limits.
  Resource containers (Solaris), virtual servers (Linux), virtual images (Docker)
- Run an entire operating system within certain limits.
  Virtual machine technology: VMWare, Xen, KVM, etc.
- Run a set of virtual machines connected via a private network.
  Virtual networks (SDNs) provision bandwidth between virtual machines.
- Run a private virtual architecture for every customer.
  Automated tools replicate virtual infrastructure as needed.

Amazon AWS

- Grew out of Amazon's need to rapidly provision and configure machines of standard configurations for its own business.
- Early 2000s – Both private and shared data centers began using virtualization to perform "server consolidation"
- 2003 – Internal memo by Chris Pinkham describing an "infrastructure service for the world."
- 2006 – S3 first deployed in the spring, EC2 in the fall
- 2008 – Elastic Block Store available.
- 2009 – Relational Database Service
- 2012 – DynamoDB
- 2014 − Lambda ("Serverless")
- 2016 (?) − Elastic Container Service
- **Does it turn a profit?**


AWS Management Conso
https://us-west-2.console.aws.amazon.com/console/home?region=us-west-2#
Services
Edit
Douglas Thain
Oregon
Help
Amazon Web Services
Compute & Networking
Direct Connect
Dedicated Network Connection to AWS
EC2
Virtual Servers in the Cloud
Route 53
Scalable Domain Name System
VPC
Isolated Cloud Resources
Storage & Content Delivery
CloudFront
Global Content Delivery Network
Glacier
Archive Storage in the Cloud
S3
Scalable Storage in the Cloud
Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage
Database
DynamoDB
Predictable and Scalable NoSQL Data Store
ElastiCache
In-Memory Cache
RDS
Managed Relational Database Service
Redshift
Managed Petabyte-Scale Data Warehouse Service
Deployment & Management
CloudFormation
Templated AWS Resource Creation
CloudTrail
User Activity and Change Tracking
CloudWatch
Resource and Application Monitoring
Directory Service
Managed Directories in the Cloud
Elastic Beanstalk
AWS Application Container
IAM
Secure AWS Access Control
OpsWorks
DevOps Application Management Service
Trusted Advisor
AWS Cloud Optimization Expert
Analytics
Data Pipeline
Orchestration for Data-Driven Workflows
Elastic MapReduce
Managed Hadoop Framework
Kinesis
Real-time Processing of Streaming Big Data
Mobile Services
Cognito
User Identity and App Data Synchronization
Mobile Analytics
Understand App Usage Data at Scale
SNS
Push Notification Service
App Services
AppStream
Low Latency Application Streaming
CloudSearch
Managed Search Service
Elastic Transcoder
Easy-to-use Scalable Media Transcoding
SES
Email Sending Service
SQS
Message Queue Service
SWF
Workflow Service for Coordinating Application Components
Applications
WorkSpaces
Desktops in the Cloud
Zocalo
Secure Enterprise Storage and Sharing Service
Additional Resources
Getting Started
See our documentation to get started and learn more about how to use our services.
AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.
AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.
Service Health
All services operating normally.
Updated: Oct 26 2014 21:31:00 GMT-0400
Service Health Dashboard
Set Start Page
Console Home


console.aws.amazon.com
aws
Services
Resource Groups
Douglas Thain
N. Virginia
Support
AWS services
Find a service by name or feature (for example, EC2, S3 or VM, storage).
Recently visited services
Lambda
S3
Billing
IAM
AWS Organizations
All services
Compute
EC2
Lightsail
ECS
EKS
Lambda
Batch
Elastic Beanstalk
Storage
S3
EFS
Glacier
Storage Gateway
Database
RDS
DynamoDB
ElastiCache
Neptune
Amazon Redshift
Migration
AWS Migration Hub
Management Tools
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services
Media Services
Elastic Transcoder
Kinesis Video Streams
MediaConvert
MediaLive
MediaPackage
MediaStore
MediaTailor
Machine Learning
Amazon SageMaker
Amazon Comprehend
AWS DeepLens
AWS Cost Management
AWS Cost Explorer
AWS Budgets
Mobile Services
Mobile Hub
AWS AppSync
Device Farm
Mobile Analytics
AR & VR
Amazon Sumerian
Application Integration
Step Functions
Amazon MQ
Simple Notification Service
Simple Queue Service
SWF
Customer Engagement
Amazon Connect
Pinpoint
Helpful tips
Manage your costs
Monitor your AWS costs, usage, and reservations using AWS Budgets. Start now
Create an organization
Use AWS Organizations for policy-based management of multiple AWS accounts. Start now
Explore AWS
Machine Learning with Amazon SageMaker
The fastest way to build, train, and deploy machine learning models. Learn more.
Amazon Relational Database Service (RDS)
RDS manages and scales your database for you. RDS supports Aurora, MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server. Learn more.
AWS Fargate Runs Containers for You
AWS Fargate works with Amazon ECS to run and scale your containers for you. Pay only for the compute resources you need, scale quickly, and run any size application.
Learn more.
AWS Marketplace
Find, buy, and deploy popular software products that run on AWS. Learn more.

Terminology

- Instance = One running virtual machine.
- Instance Type = hardware configuration: cores, memory, disk.
- Instance Store Volume = Temporary disk associated with instance.
- Image (AMI) = Stored bits which can be turned into instances.
- Key Pair = Credentials used to access VM from command line.
- Region = Geographic location, price, laws, network locality.
- Availability Zone = Subdivision of region that is fault-independent.

Model	vCPU	CPU Credits / hour	Mem (GiB)	Storage (GB)
t2.micro	1	6	1	EBS Only
t2.small	1	12	2	EBS Only
t2.medium	2	24	4	EBS Only

Model	vCPU	Mem (GiB)	SSD Storage (GB)
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Use Cases

High performance front-end fleets, web-servers, on-demand batch processing, distributed analytics, high performance science and engineering applications, ad serving, batch processing, MMO gaming, video encoding, and distributed analytics.

Model	vCPU	Mem (GiB)	SSD Storage (GB)
m3.medium	1	3.75	1 x 4
m3.large	2	7.5	1 x 32
m3.xlarge	4	15	2 x 40
m3.2xlarge	8	30	2 x 80

Model	vCPU	Mem (GiB)	SSD Storage (GB)
r3.large	2	15.25	1 x 32
r3.xlarge	4	30.5	1 x 80
r3.2xlarge	8	61	1 x 160
r3.4xlarge	16	122	1 x 320
r3.8xlarge	32	244	2 x 320

Use Cases

We recommend memory-optimized instances for high performance databases, distributed memory caches, in-memory analytics, genome assembly and analysis, larger deployments of SAP, Microsoft SharePoint, and other enterprise applications.

EC2 Pricing Model

- Free Usage Tier
- On-Demand Instances
  - Start and stop instances whenever you like, costs are rounded up to the nearest hour. (Worst price)
- Reserved Instances
  - Pay up front for one/three years in advance. (Best price)
  - Unused instances can be sold on a secondary market.
- Spot Instances
  - Specify the price you are willing to pay, and instances get started and stopped without any warning as the marked changes. (Kind of like Condor!)

http://aws.amazon.com/ec2/pricing/

Free Usage Tier

- 750 hours of EC2 running Linux, RHEL, or SLES t2.micro instance usage
- 750 hours of EC2 running Microsoft Windows Server t2.micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

On-Demand Instances

Linux | RHEL | SLES | Windows | Windows with SQL Standard | Windows with SQL Web

Windows with SQL Enterprise | Linux with SQL Standard | Linux with SQL Web | Linux with SQL Enterprise

Region: US East (Ohio)

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0052 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0104 per Hour
t3.small	2	Variable	2 GiB	EBS Only	$0.0208 per Hour
t3.medium	2	Variable	4 GiB	EBS Only	$0.0416 per Hour
t3.large	2	Variable	8 GiB	EBS Only	$0.0832 per Hour
t3.xlarge	4	Variable	16 GiB	EBS Only	$0.1664 per Hour
t3.2xlarge	8	Variable	32 GiB	EBS Only	$0.3328 per Hour

Reserved Instances

t3.large

STANDARD 1-YEAR TERM					
Payment Option	**Upfront**	**Monthly***	**Effective Hourly****	**Savings over On-Demand**	**On-Demand Hourly**
No Upfront	$0.00	$38.11	$0.052	37%	$0.0832
Partial Upfront	$218.00	$18.10	$0.05	40%	
All Upfront	$426.00	$0.00	$0.049	42%	

CONVERTIBLE 1-YEAR TERM					
Payment Option	**Upfront**	**Monthly***	**Effective Hourly****	**Savings over On-Demand**	**On-Demand Hourly**
No Upfront	$0.00	$43.80	$0.06	28%	$0.0832
Partial Upfront	$250.00	$20.88	$0.057	31%	
All Upfront	$490.00	$0.00	$0.056	33%	

STANDARD 3-YEAR TERM					
Payment Option	**Upfront**	**Monthly***	**Effective Hourly****	**Savings over On-Demand**	**On-Demand Hourly**
No Upfront	$0.00	$26.28	$0.036	57%	$0.0832
Partial Upfront	$438.00	$12.19	$0.033	60%	
All Upfront	$823.00	$0.00	$0.031	62%	

Spot Instances

Spot Instance Pricing History

Product : **Linux/UNIX** Instance type: **c3.4xlarge** Date range : **1 month** Availability zone: **All zones**

Availability zone	Price
us-west-2a	$0.1389
us-west-2b	$0.1390
us-west-2c	$0.1322
Date	September 28, 2014 9:08:53 AM UTC-4

Surprisingly, you can’t scale up that large.

Q: How many instances can I run in Amazon EC2?

You are limited to running up to 20 On-Demand Instances, purchasing 20 Reserved Instances, and requesting 5 Spot Instances per region. New AWS accounts may start with limits that are lower than the limits described here. Certain instance types are further limited per region as follows:

Instance Type	On-Demand Limit	Reserved Limit	Spot Limit
cg1.4xlarge	2	20	5
hi1.4xlarge	2	20	5
hs1.8xlarge	2	20	Not offered
cr1.8xlarge	2	20	5
g2.2xlarge	5	20	5

Simple Storage Service (S3)

- A **bucket** is a container for objects and describes location, logging, accounting, and access control. A bucket can hold any number of **objects**, which are files of up to 5TB. A bucket has a name that must be **globally unique**.
- Fundamental operations corresponding to HTTP actions:
  - http://**bucket**.s3.amazonaws.com/**object**
  - POST a new object or update an existing object.
  - GET an existing object from a bucket.
  - DELETE an object from the bucket
  - LIST keys present in a bucket, with a filter.
- A bucket has a **flat directory structure** (despite the appearance given by the interactive web interface.)

Easily Integrated into Web Applications

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post" enctype="multipart/form-data">

<input type="input" name="key" value="user/user1/" />

<input type="hidden" name="acl" value="public-read" />
<input type="hidden" name="success_action_redirect"
        value="http://examplebucket.s3.amazonaws.com/successful_upload.html" />
. . .
<input type="text" name="X-Amz-Credential”
        value="AKIAIOSFODNN7EXAMPLE/20130806/us-east-1/s3/aws4_request" />
. . .
<input type="submit" name="submit" value="Upload to Amazon S3" /> </form>
```

http://docs.aws.amazon.com/AmazonS3/latest/API/sigv4-post-example.html

Bucket Properties

- Versioning – If enabled, POST/DELETE result in the creation of new versions without destroying the old.
- Lifecycle – Delete or archive objects in a bucket a certain time after creation or last access or number of versions.
- Access Policy – Control **when and where** objects can be accessed.
- Access Control – Control who **may** access objects in this bucket.
- Logging – Keep track of how objects are accessed.
- Notification – Be notified when failures occur.

S3 Weak Consistency Model

Direct quote from the Amazon developer API:

“Updates to a single key are **atomic**….”

“Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. If a PUT request is successful, your data is safely stored. However, information about the changes must replicate across Amazon S3, which can take some time, and so you might observe the following behaviors:

- A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might report "key does not exist."
- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object might not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 might return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 might return the deleted data.”

Storage pricing (varies by region)

Region: US East (Ohio)

	Pricing
S3 Standard Storage	
First 50 TB / Month	$0.023 per GB
Next 450 TB / Month	$0.022 per GB
Over 500 TB / Month	$0.021 per GB
S3 Standard-Infrequent Access (S3 Standard-IA) Storage	
All storage	$0.0125 per GB
S3 One Zone-Infrequent Access (S3 One Zone-IA) Storage	
All storage	$0.01 per GB
Amazon Glacier Storage	
All storage	$0.004 per GB

Request pricing (varies by region)

For requests not otherwise specified below

Region: US East (Ohio)

	Pricing
Data Returned by S3 Select	$0.0007 per GB
Data Scanned by S3 Select	$0.002 per GB
PUT, COPY, POST, or LIST Requests	$0.005 per 1,000 requests
GET, SELECT and all other Requests	$0.0004 per 1,000 requests
Lifecycle Transition Requests into Standard – Infrequent Access or One Zone - Infrequent Access	$0.01 per 1,000 requests

DELETE requests are free. †

Amazon S3 request costs are based on the request type, and are charged on the quantity of requests or the volume of data retrieved as listed in the table below.

Data Transfer Pricing

The pricing below is based on data transferred "in" to and "out" of Amazon S3 (over the public Internet). AWS Direct Connect pricing can be found here. Transfers between S3 buckets or from S3 to any service(s) within the same region are free.

Region: US East (Ohio)

	Price
Data Transfer IN To Amazon S3 From Internet	
All data transfer in	$0.00 per GB
Data Transfer OUT From Amazon S3 To Internet	
Up to 1 GB / Month	$0.00 per GB
Next 9.999 TB / Month	$0.09 per GB
Next 40 TB / Month	$0.085 per GB
Next 100 TB / Month	$0.07 per GB
Greater than 150 TB / Month	$0.05 per GB
Data Transfer OUT From Amazon S3 To	
CloudFront	$0.00 per GB
US East (N. Virginia)	$0.01 per GB
Asia Pacific (Singapore)	$0.02 per GB

Elastic Block Store

- An EBS volume is a **virtual disk** of a fixed size with a block read/write interface. It can be **mounted** as a filesystem on a running EC2 instance where it can be **updated incrementally.** Unlike an instance store, an EBS volume is **persistent.**
- (Compare to an S3 object, which is essentially a file that must be accessed in its entirety.)
- Fundamental operations:
  - CREATE a new volume (1GB-1TB)
  - COPY a volume from an existing EBS volume or S3 object.
  - MOUNT on one instance at a time.
  - SNAPSHOT current state to an S3 object.

Amazon EBS Pricing

Cr

With Amazon EBS, you only pay for what you use. The pricing for Amazon EBS volumes is listed below.

Region: US East (N. Virginia)

Amazon EBS General Purpose (SSD) volumes

- $0.10 per GB-month of provisioned storage

Amazon EBS Provisioned IOPS (SSD) volumes

- $0.125 per GB-month of provisioned storage
- $0.065 per provisioned IOPS-month

Amazon EBS Magnetic volumes

- $0.05 per GB-month of provisioned storage
- $0.05 per 1 million I/O requests

Amazon EBS Snapshots to Amazon S3

- $0.095 per GB-month of data stored

EBS is approx. 3x more expensive by volume and 10x more expensive by IOPS than S3.

Use Glacier for Cold Data

- Glacier is structured like S3: a **vault** is a container for an arbitrary number of **archives**. Policies, accounting, and access control are associated with vaults, while an archive is a single object.
- However:
  - All operations are asynchronous and notified via SNS.
  - Vault listings are updated once per day.
  - Archive downloads may take up to four hours.
  - Only 5% of total data can be accessed in a given month.
- Pricing:
  - Storage: $0.01 per GB-month
  - Operations: $0.05 per 1000 requests
  - Data Transfer: Like S3, free within AWS.
- S3 Policies can be set up to automatically move data into Glacier.

Durability

- Amazon claims about S3:
  - Amazon S3 is designed to sustain the concurrent loss of data in two facilities, e.g. 3+ copies across multiple available domains.
  - 99.999999999% durability of objects over a given year.
- Amazon claims about EBS:
  - Amazon EBS volume data is replicated across multiple servers in an Availability Zone to prevent the loss of data from the failure of any single component.
  - Volumes <20GB modified data since last snapshot have an annual failure rate of 0.1% - 0.5%, resulting in complete loss of the volume.
  - Commodity hard disks have an AFR of about 4%.
- Amazon claims about Glacier is the same as S3:
  - Amazon S3 is designed to sustain the concurrent loss of data in two facilities, e.g. 3+ copies across multiple available domains PLUS periodic internal integrity checks.
  - 99.999999999% durability of objects over a given year.

- **Beware of oversimplified arguments about low-probability events!**

Amazon Elastic File Services (EFS)

- EFS is a standalone file service designed to shared among VMs.
  - File System Instance − Files, directories, storage allocation, multiple AZs
  - Mount Target − DNS name, IP address, NFS target, single AZ:

    file-system.id.efs.aws-region.amazonaws.com

  - Inside of VM, use normal NFS connection to the mount target:

```
mount –t nfs –o rsize=1M,wsize=1M,…
      file-system-dns-name /mnt/data
```

https://docs.aws.amazon.com/efs

Amazon ElasticFile Services (EFS)

- Data Consistency Model
  - "close to open" consistency semantics.
  - In (normal) asynchronous, sequential I/O mode:
    - Data is durable after an fsync or a close.
    - Data is visible to other processes that open after you close.
    - Indeterminate case: Process B opens before Process A closes.
  - In (explicit) synchronous I/O mode (O_DIRECT)
    - Non-appending writes are immediately visible.
    - (This implies that appending writes are not. Why?)
    - Hint: Size of file is a property of metadata.

Everything in AWS is Carefully Limited!

- 50KB/s per GB of data in free throughput
- 250MB/s Max EFS throughput per EC2 Instance
- 128 user IDs can have open files
- 32K open files on a single instance
- 4080B max symbolic link length
- 1000 maximum directory depth
- 47.9TiB max bytes per file
- 87 max locks per file (?)
- 7000 file operations per second

https://docs.aws.amazon.com/efs/latest/ug/limits.html

Amazon EFS Pricing

Pricing Table

Region	Storage (GB-Month)	Provisioned Throughput (MB/s-Month)	EFS File Sync (Per-GB into EFS)
US East (N. Virginia)	$0.30	$6.00	$0.01
US East (Ohio)	$0.30	$6.00	$0.01
US West (N. California)	$0.33	$6.60	$0.01
US West (Oregon)	$0.30	$6.00	$0.01
Asia Pacific (Seoul)	$0.33	$6.60	$0.01
Asia Pacific (Singapore)	$0.36	$7.20	$0.01
Asia Pacific (Sydney)	$0.36	$7.20	$0.01
Asia Pacific (Tokyo)	$0.36	$7.20	$0.01
EU (Frankfurt)	$0.36	$7.20	$0.01
EU (Ireland)	$0.33	$6.60	$0.01

Within your first 12 months on AWS, you can use up to 5 GB/month for free.

	EBS	EFS
Latency	"lowest" (local hardware)	"low" (network)
Max Throughput	2 GB/s	10 GB/s
Clients (VMs)	Single	O(1000)
Storage Cost Per Month	$0.05 / GB (Magnetic Disk)	$0.30 / GB
Access Cost Per Month	$0.05 / 1M IOPS (Magnetic Disk	$6.00 / MB/s Xput

Serverless Computing

- Aside: Worst. Name. Ever. (Except for deviceless computing.)
- "Serverless takes it a step further, where you don't even think about the infrastructure. You think of functions that your code needs to perform." – Prof. Paul Brenner, Notre Dame CRC *
- Key Idea:
  - Define a fixed function (and associated state) once.
  - Invoke that function on small amounts of data many times.
  - Tear down the function (and state).
- Cloud provider takes the responsibility of scaling the system up and down in order to meet the offered load.

*** https://cacm.acm.org/magazines/2018/2/224625-going-serverless/fulltext**

Amazon Lambda

- **With AWS Lambda, you can run code without provisioning or managing servers. You pay only for the compute time that you consume—there's no charge when your code isn't running. You can run code for virtually any type of application or backend service—all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.**

https://docs.aws.amazon.com/lambda/index.html#lang/en_us

Amazon Lambda

- Supported Environments:
  - Node.js, Python, C#, Go, Java, PowerShell
- Encouraged mode of use:
  - Define a self-contained work function in the native language.
  - Write a handler function that packs/unpacks JSON and invokes the work func.
- Possible, but less encouraged and can cause trouble:
  - Handler function uses system/exec/shell to invoke external processes in order to compute result.
- The lambda runtime is defined relative to a language runtime (e.g. Python2.7, Python3.0) and not specific about the OS environment.

Requirements

- Function must be **stateless**.
- No affinity with the underlying compute infrastructure.
- Local file system access, child processes, and similar artifacts to be limited to the lifetime of the request.
- Persistent state should be stored in Amazon S3, DynamoDB, etc..

- P.S. Easy to say, but requires discipline to stick to this.
- P.P.S. Does this sound similar to another system we have discussed?

Defining a Function

- Unique name.
- Provide the code (obviously)
- Use a deployment package if addl dependencies needed.
- State memory required by function. (CPU is proportional)
- State the timeout for the function. (Default is **3** seconds!)
- Give permissions to access other AWS services.

How does it work?

- (Sketch on the board)

Lambda Pricing in Oct 2018

- At first:
  - First 1M invocations are free.
  - 400,000 GB-s of memory x execution time are free.
- Then:
  - $0.20 per 1M invocations.
  - $0.00001667 per GB-s of memory x execution time.

- Careful: Functions can incur addlcosts by invoking other services.

Performance Limits

Resource	Limit
Function memory allocation	128 MB to 3008 MB, in 64 MB increments.
Function timeout	900 seconds (15 minutes)
Function environment variables	4 KB
Invocation payload (request and response)	6 MB (synchronous) 128 KB (asynchronous)
Deployment package size	50 MB (zipped) 256 MB (unzipped) 3 MB (console editor)
`/tmp` directory storage	512 MB
File descriptors	1024
Execution processes/threads	1024

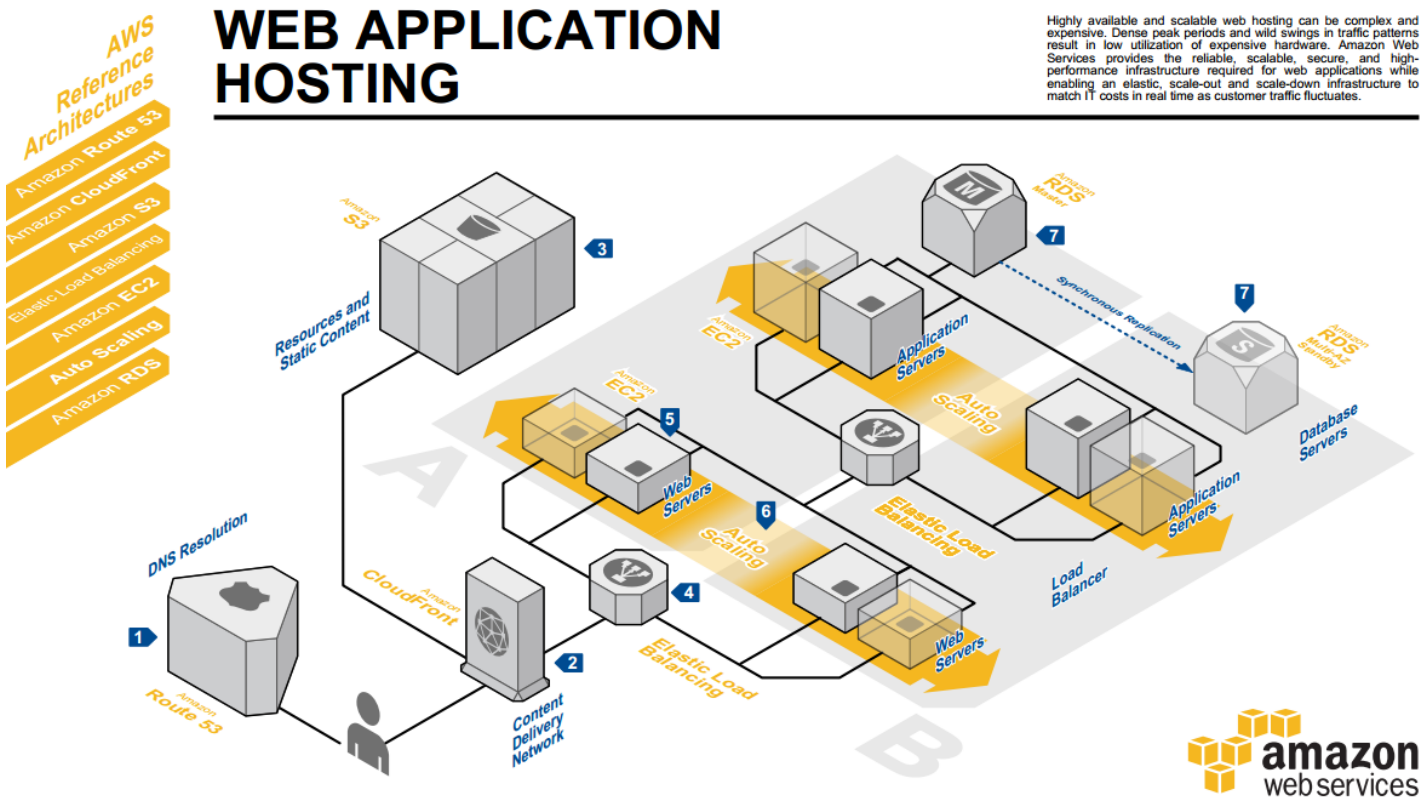
Principle: Hierarchy of Invocation Costs

- Steps to getting remote work done:
  - Provision physical hardware.
  - Activate virtual machine on hardware.
  - Deploy container on virtual machine.
  - Start process and "warm up" the program.
  - Load necessary data for task.
  - Invoke the function that constitutes your work.
- Should we setup/teardown everything for every invocation?
- What are the pros/cons of staying at the last level?
- (Hint: Think about what other concurrent users may do.)

Architecture Center

- Ideas for constructing large scale infrastructures using AWS:

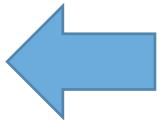
  http://aws.amazon.com/architecture/



Command Line Setup

- Go to your profile menu (your name) in the upper right hand corner, select “Security Credentials” and “Continue to Security Credentials”
- Select “Access Keys”
- Select “New Access Key” and save the generated keys somewhere.
- Edit ~/.aws/config and set it up like this:

```
[default]
output = json
region = us-west-2
aws_access_key = XXXXXX
aws_secret_access_key = YYYYYYYYYYY
```



Note the syntax here is different from how it was given in the web console!
AWSAccessKey=XXXXXX
AWSSecretAccessKey=YYYYYYYY

- Now test it: **aws ec2-describe-instances**

S3 Command Line Examples

```
aws    s3      mb     s3://bucket
. . .          cp     localfile   s3://bucket/key
               mv     s3://bucket/key    s3://bucket/newname
               ls     s3://bucket
               rm     s3://bucket/key
               rb     s3://bucket

aws    s3      help
aws    s3      ls help
```

EC2 Command Line Examples

```
aws    ec2    describe-instances
              run-instances --image-id ami-xxxxx -- count 1
                                --instance-type t1.micro --key-name keyfile
              stop-instances --instance-id i-xxxxxx

aws    ec2    help
aws    ec2    start-instances help
```

Warmup: Get Started with Amazon

- Skim through the AWS documentation.
- Sign up for AWS at http://aws.amazon.com
- (Skip the IAM management for now)
- Apply the service credit you received by email.
- Create and download a Key-Pair, save it in your home directory.
- Create a VM via the AWS Console
- Connect to your newly-created VM like this:
  - ssh -i my-aws-keypair.pem ec2-user@ip-address-of-vm
- Create a bucket in S3 and upload/download some files.

Demo Time

http://aws.amazon.com