Amazon Web Services (AWS)

Mastering Cloud Computing
Chapter 9.1
Paul Talaga

UNIVERSITY OF Cincinnati

AWS History (from wikipedia)

Isn't Amazon an online retailer?

Yes! And they need to manage lots of computers!

Sell their internally developed technology!

- 2003/2004 - Paper on idea - Simple Queue Service - EC2 (Cape Town)
- 2006 - Full AWS launch - SOAP/REST
- 2010 - All amazon.com retail on AWS
- Outages - 2011, 2012, none in 2013 or 2014?

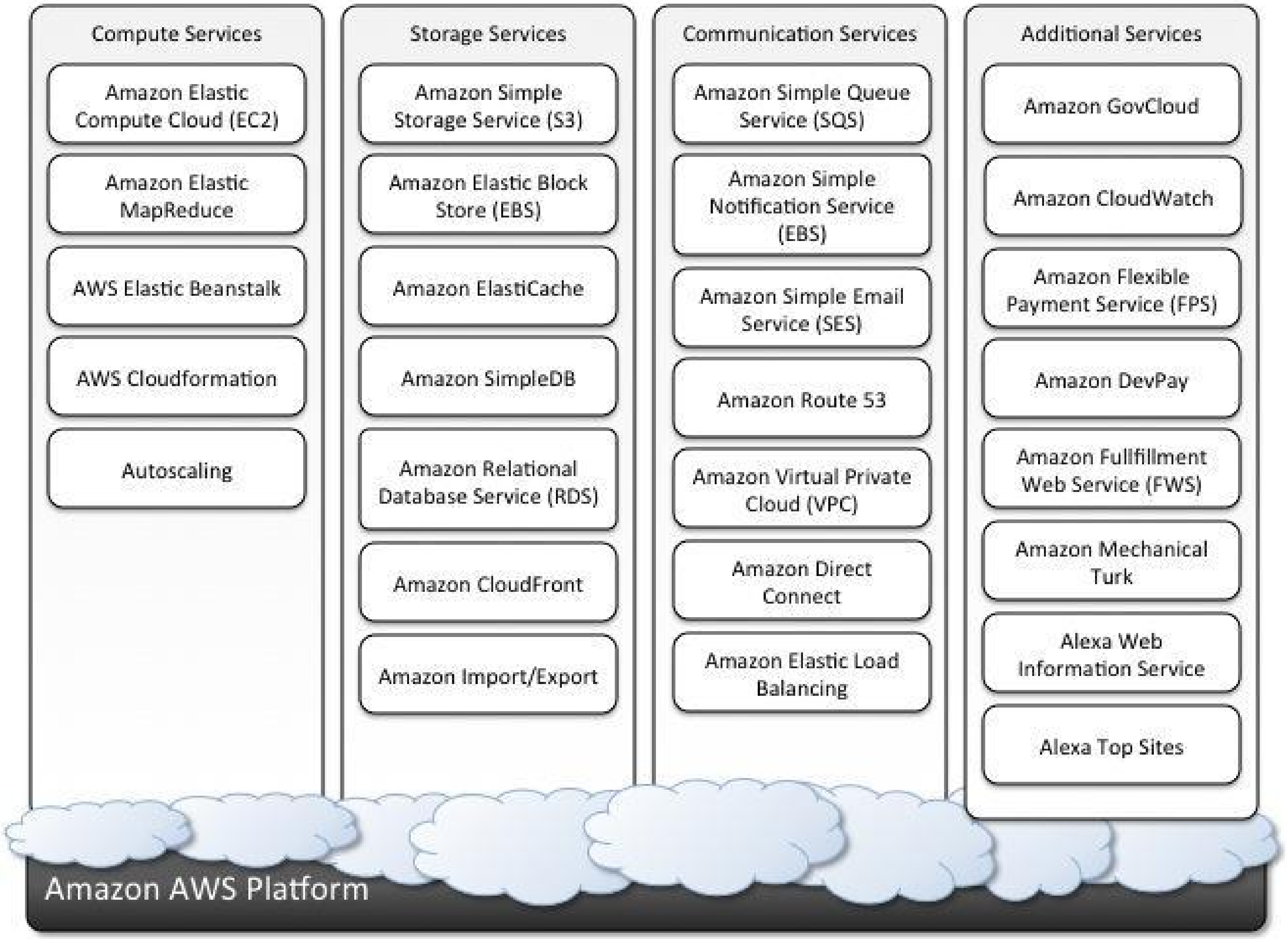
UNIVERSITY OF Cincinnati

AWS

- Users: Netflix, NASA, Obama Campaign, Pintrest, CIA, many startups
- Estimated revenue $1.5 **Billion** in 2012.
- AWS Certification Program - April 2013
- Concerns?
  - 'Central' point of failure - *Many* consumer services reliant on AWS!
  - Data security?
  - Vendor lock-in

UNIVERSITY OF Cincinnati

AWS Ecosystem


Compute Services
Amazon Elastic Compute Cloud (EC2)
Amazon Elastic MapReduce
AWS Elastic Beanstalk
AWS Cloudformation
Autoscaling
Storage Services
Amazon Simple Storage Service (S3)
Amazon Elastic Block Store (EBS)
Amazon ElastiCache
Amazon SimpleDB
Amazon Relational Database Service (RDS)
Amazon CloudFront
Amazon Import/Export
Communication Services
Amazon Simple Queue Service (SQS)
Amazon Simple Notification Service (EBS)
Amazon Simple Email Service (SES)
Amazon Route 53
Amazon Virtual Private Cloud (VPC)
Amazon Direct Connect
Amazon Elastic Load Balancing
Additional Services
Amazon GovCloud
Amazon CloudWatch
Amazon Flexible Payment Service (FPS)
Amazon DevPay
Amazon Fullfillment Web Service (FWS)
Amazon Mechanical Turk
Alexa Web Information Service
Alexa Top Sites
Amazon AWS Platform

Compute Services

Elastic Compute Cloud (EC2)

- Virtual machines - IaaS
- User can select:
  - Amazon Machine Image (AMI) - template to start a VM (market, or create your own)
  - # cores - ECU (EC2 Compute Unit) - 1 ECU = 2007 Opteron/Xeon
  - Memory
  - Local storage
  - Network & Security (firewall specifications)
  - Location and availability zone
  - # to start!

UNIVERSITY OF Cincinnati

EC2 Details….

- How to access securely? Passwordless SSH (private/public keys) for Linux
  - In unix, `ssh-keygen` to make pair
  - Can use your github public key!
  - Your public key gets injected into the image
  - `ec2-user` for username, unless noted
- Windows must use an Admin password retrieval system. Needs your private key.
- Must define a security policy: what TCP ports to allow in. 22 (ssh), 80 (http?) *and from where*.

UNIVERSITY OF Cincinnati

EC2 Instance Types

What type of machine to start:

General purpose

m1.small | m1.medium | m1.large | m1.xlarge | m3.medium | m3.large | m3.xlarge | m3.2xlarge

Compute optimized

c1.medium | c1.xlarge | c3.large | c3.xlarge | c3.2xlarge | c3.4xlarge | c3.8xlarge | cc2.8xlarge

Memory optimized

m2.xlarge | m2.2xlarge | m2.4xlarge | cr1.8xlarge

Storage optimized

hi1.4xlarge | hs1.8xlarge | i2.xlarge | i2.2xlarge | i2.4xlarge | i2.8xlarge

Micro instances

t1.micro

GPU instances

cg1.4xlarge | g2.2xlarge

http://aws.amazon.com/ec2/instance-types/

What type of OS (AMI)?

- awsMarketplace - buy/sell AMI’s!
  - Some AMI’s change extra /hr
- Pre-configured software!
  - MongoDB, Wordpress, Ruby Stack
- Linux Free/Paid
  - CentOS, Debian, RedHat, SUSE
- Windows
  - Server 2008, 2012
- Or create your own!!!

Pricing!!!!!

- New users get access to FREE TIER, for a year.
- Cost for running EC2 /hour (rounded up)
  - On-demand, reserve, spot pricing
- Cost for data transfer
  - Free inbound from internet
  - Tiered outbound to internet
  - Some cost between EC2 instances
- Cost for data storage (S3, EBS)

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

UNIVERSITY OF Cincinnati

AWS Locations & Availability Zones

- 9 Regions AWS locations (+ GovCloud)
  - Tokyo, Singapore, Sydney, Sao Paulo, Ireland, N.California, Oregon, N.Virginia, Frankfurt
- CloudFront/Route 53 distributed around globe
- Availability Zones are isolated parts of a region (datacenter)
  - Your zone #'s don't match others!
  - Distribute instances for reliability

View AWS Status

UNIVERSITY OF Cincinnati

Useful Notes:

- Use `sudo <command>` for commands requiring root privileges
- `sudo yum [search|install] <package>` as in `sudo yum install httpd` for Apache HTTPD Webserver
- Check startup services with `chkconfig` and turn on startup of service with `sudo chkconfig httpd on`
- Manually start a service with `sudo service httpd [start|stop|status|restart]`
- HTTPD default serving location is `/var/www/html/` and only root has write access.
- View HTTPD logs in `/var/log/httpd/* access_log error_log` but you'll need to be root to view these files.
- `tail <file>` is useful for viewing the end of log files. `tail -f <file>` will follow changes so you can see them in real time.
- If HTTPD isn't working, your firewall `iptables` may be running. `sudo iptables -L -n` to see config. `sudo service iptables stop` to turn off. (Security risk, but OK for our purposes)
- `wget http://someURLhere/somepage.html` - Download a page in terminal
- Can run startup code using `User data` base64 encoded:

```
#!
touch ~/PLEASE_WORK.txt
```

UNIVERSITY OF Cincinnati

AWS EC2 API

- Allows *programmatic* control over all Dashboard functions!
- Requires a separate key pair for authentication: AWS access key/secret
- Many supported languages, I'll demo Python using BOTO
- Be careful!!! Your program linked to CC!
- BOTO API, BOTO Examples

UNIVERSITY OF Cincinnati

BOTO Basics

```
import boto.ec2

conn = boto.ec2.connect_to_region("us-east-1",
   aws_access_key_id='<put your key in here>',
   aws_secret_access_key='<put your key in here>')

# Start instances
conn.run_instances(
            'ami-bba18dd2',
            key_name='talagakey2',   # your ssh keypair name
            instance_type='t1.micro',
            security_groups=['ssh-http']) # security group name
```

BOTO Basics

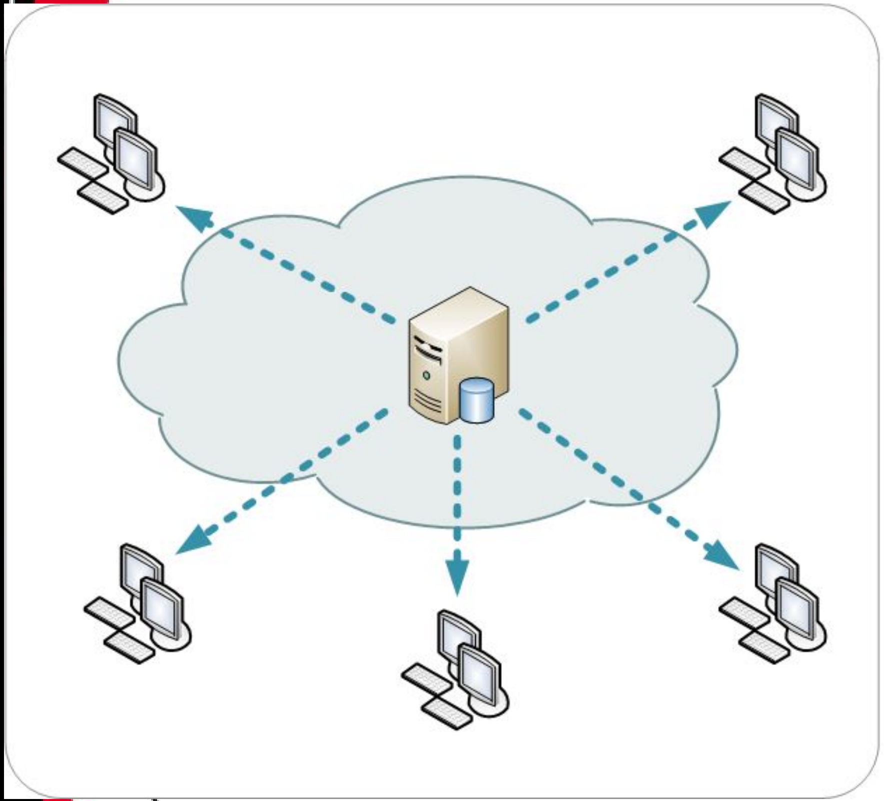
```
# print instances
instances = conn.get_only_instances()
for i in instances:
    print(“id”,i.id, “ DNS:”,i.public_dns_name)

# Shutdown all instances (terminate)

instance_ids = map(lambda a:a.id,instances)

conn.terminate_instances(instance_ids = instance_ids)
```

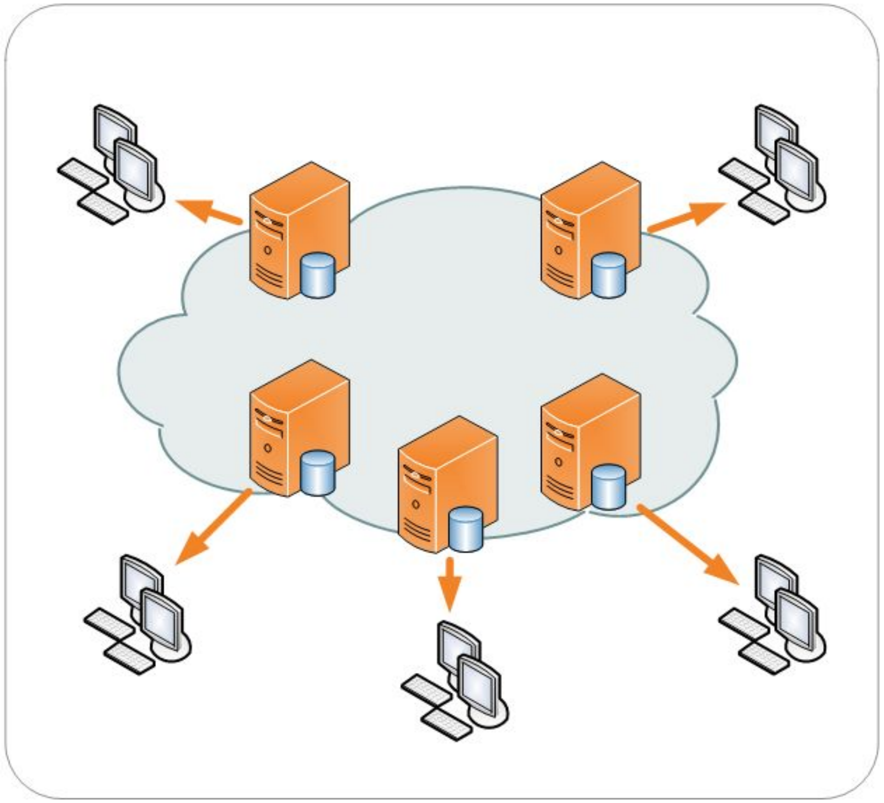
For more demos visit our GitHub repository: https://github.uc.edu/talagapl/cloud2014

UNIVERSITY OF Cincinnati

AWS Cloudfront

- Amazon's Content Delivery Network (CDN) - similar to Akamai, Limelight, and others





Simple Storage Service (S3)

- REST interface
- Eventually consistent!
- Cost: $0.03 / GB / Month + transfer
- 2 level hierarchy
  - Buckets - ALL users share namespace!
  - Objects - ‘files’ in a bucket key:value
- Accessible: http://s3.amazonaws.com/bucket_name https://s3.amazonaws.com/UCTest/beach.jpg renaming possible via download, delete, upload.
- Access:
  - S3 Management Console
  - BOTO API - Examples

UNIVERSITY OF Cincinnati

S3 Access in BOTO - I

```
from boto.s3.connection import S3Connection
from boto.s3.key import Key

conn = S3Connection(connection.access_key,connection.secret_access_key)

# Create a new bucket
#conn.create_bucket('uctest')

# Get a connection to the bucket
b = conn.get_bucket('uctest')

# Set a new key
k = Key(b)
k.key = 'NewFolder/DSC_8143.NEF'
#k.set_contents_from_string("mystery3")
wbytes = k.set_contents_from_filename('DSC_8143.NEF')

#wbytes = k.set_contents_from_filename('DSC_8143.NEF', policy= 'public-read',
   reduced_redundancy = True)
```

UNIVERSITY OF Cincinnati

S3 Access in BOTO - II

```
# Print all keys in bucket
print "All keys in bucket:"
for key in b.list():
  print key.key

# Get contents of key
k = Key(b)
k.key = 'NewFolder/DSC_8143.NEF'
k.get_contents_to_filename('myimage')
# newstring = k.get_contents_as_string()

# Delete all keys
for key in b.list():
  print key.delete()
```

UNIVERSITY OF Cincinnati

AWS Glacier

- AWS Long term storage (backups)
- INFREQUENT access - hours to retrieve
- Cheap! (S3 - $0.03->$0.027 / GB / Month)
  - Storage: $ 0.01 / GB / month (VA)
  - Download: 5% total /month free, then $0.05 /GB
  - Transfer: (out of AWS) 1st GB free
- Access:
  - Web Dashboard
  - BOTO API - S3 - Bucket Lifecycle Policy
  - BOTO API - Glacier

UNIVERSITY OF Cincinnati