Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU44000 Internet Applications

Week 6 Lecture 1

Conor Sheedy

Practical Cloud Computing with Amazon Web Services

Amazon Web Services, AWS

Offer a wide range of services

Original product was Infrastructure-as-a-service

- **EC2**
- **Many services built on top of that**

Amazon Elastic Compute Cloud

Virtual Machine Types

- Hardware
- Software

Availability Zones & Regions

Pricing

- Competitivity with running own infrastructure

Security Groups

Attached Storage : Elastic Block Store

The Virtual Machine Instance

User chooses to instantiate a virtual machine of a given type of hardware with a given type of software

- **Depends on application needs**
  - Eg. Fast CPU?
  - Lot of memory?

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

- **CPUs can 'burst' using a system of CPU credits/hour**
  - instances accumulate credits when idle and consume them when busy
- **Instances are allocated 3,6,12… CPU credits/hour**

We will mostly use:

Instance	vCPU*	CPU Credits / hour	Mem (GiB)	Storage	Network Performance
t2.nano	1	3	0.5	EBS-Only	Low

- **Each Virtual CPU (vCPU) is equivalent of one Intel Xeon Core or Hyperthread (depending on instance)**
- **Memory/Networking depends on instance type**

The Machine Image

- **Each machine instance is created with a local enduring storage**
- **The operating system and installed software constitute the Amazon Machine Image (AMI)**
- **Software licence fees can be included in the cost of running an instance**
- **‘Curated’ software stacks are available by ‘subscribing’ to sellers in the AMI Marketplace (https://aws.amazon.com/marketplace)**
- **Users can create their own machine image with the software mix they need**

Sample Pricing – EU(Ireland) 2019

vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
t3.nano	2	Variable	0.5 GiB	EBS Only	$0.0057 per Hour
t3.micro	2	Variable	1 GiB	EBS Only	$0.0114 per Hour
t3.small	2	Variable	2 GiB	EBS Only	$0.0228 per Hour
t3.medium	2	Variable	4 GiB	EBS Only	$0.0456 per Hour
p3dn.24xlarge	96	345	768 GiB	2 x 900 NVMe SSD	$33.711 per Hour
i3en.24xlarge	96	N/A	768 GiB	8 x 7500 NVMe SSD	$12.00 per Hour

Cost Competitivity

- **Can be a low cost option for businesses compared with running their own hardware**
- **Cost control becomes an important concern**
- **Ease of scaling could lead to costs scaling**
- **Can be a large fraction of total cost for some buisnesses**

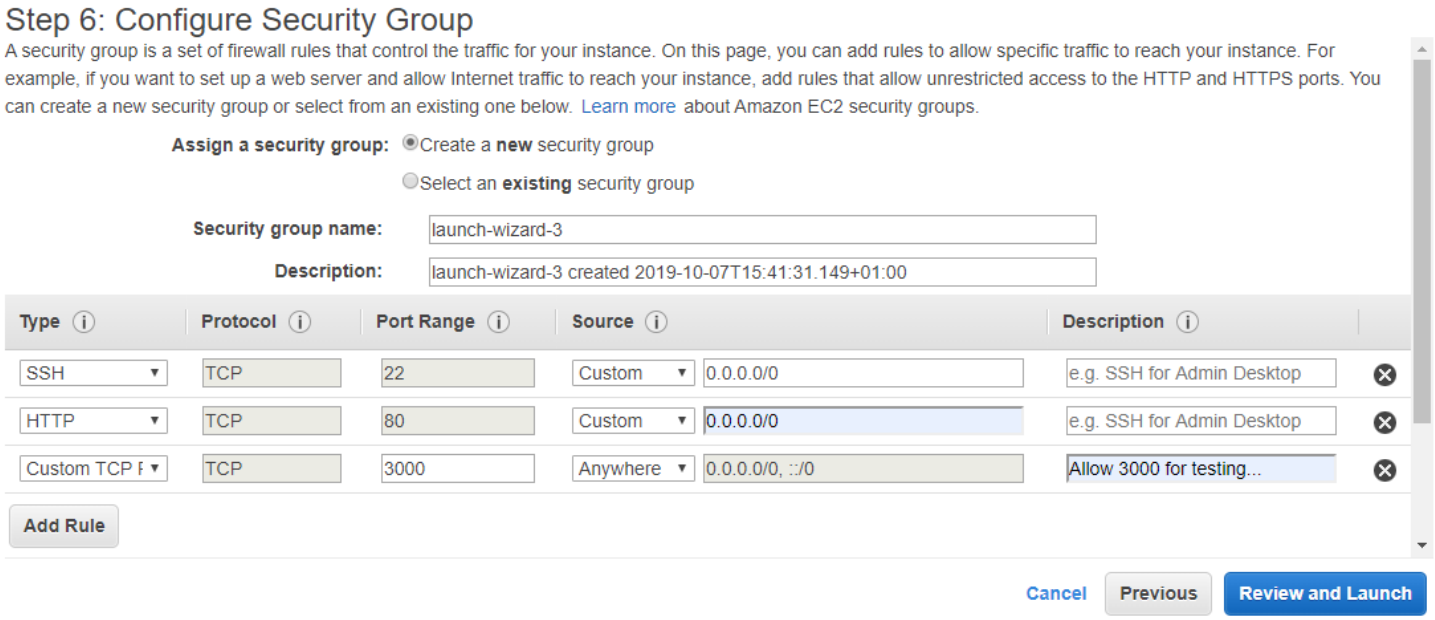
Disk Storage

- **Local hard disk(or SSD) is provided by Elastic Block Store (EBS) images**
- **By default each created machine instance gets one (LINUX AMI defaults to 8 GiB)**
- **After an instance is stopped, user is still charged rental for the EBS store – until the instance is terminated**
- **Charges are based on location and space used over time**
- **E.g. 2019, Europe(Ireland)**

  $0.11 per GB-month of provisioned storage

Launching the Instance – Security Environment

- **Each machine instance is launched within a 'Security Group'**
- **By default everything is closed – unless explicitly opened**
- **Security group can be created 'live' using the launch wizard or pre-configured**



AWS Student Environment

- **AWS recently updated learning support tools**
  - https://aws.amazon.com/education/awseducate/
  - Register now with tcd email
- **Sandboxed environment**
  - No such thing a ‘free’
  - Free Tier runs out
  - Credit card not needed in a Sandbox
  - Otherwise Credit card required
  - Mistakes can be costly
  - On gaining sufficient experience in the sandbox
    - Sign up at your own risk
    - Manage that risk appropriately

Storage Hierarchy

Elastic Block Store (EBS)

- Fast storage
- Can specify speed of access
- Indented for short term storage

Simple Storage Service (S3)

- Standard for longer term, larger storage
- Various flavours based on frequency or speed of Access

Glacier – a slow cheap (tape-based) storage for end-of-life data (perhaps held for compliance purposes)

- Suitable for archival storage

Simple Storage Service, S3

- **Introduced in 2006**
- **Used to store Objects – the Objects are grouped into Buckets**
- **Much Cheaper than using Elastic Block Storage, EBS:**

	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

- **Buckets are given names**
  - must be unique across AWS

S3 Buckets Used by

- **Dropbox up to 2016**
  - Freemium business model
  - Loss made on users of free
  - Needs to be covered by paying customers
- **Netflix as ‘System of Record’**
- **Tumblr, Pinterest**

- **Can be configured to serve static web file**
  - cheap and effective static website hosting

S3 Bucket Contents

- **A bucket can contain objects identified by a 'Key' = name**
- **The bucket has no internal structure (bag of objects), but the AWS interface supports folders by using '/' in the name**
- **If you upload a cat_photo.jpg into the folder /myphotos**
  - it uploads and creates and object with the name /myphotos/cat_photo.jpg
- **Access Control**

– Can be applied at Bucket and at Object Level

– Can be accessed using a REST style CRUD interface using: http://bucket.s3-aws-region.amazonaws.com or s3-aws-region.amazonaws.com/bucket

– Wrapped API available (AWS-SDK) for Node.js

S3 Usage Pricing

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 or Intelligent-Tiering	$0.01 per 1,000 requests

Node.js code to upload a file to an S3 bucket

```
/ Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create S3 service object
s3 = new AWS.S3({apiVersion: '2006-03-01'});

// call S3 to retrieve upload file to specified bucket
// process gets parameters from the command line passed
// to node.js
var uploadParams = {Bucket: process.argv[2], Key: ' ', Body: '
'};
var file = process.argv[3];

// Configure the file stream and obtain the upload parameters
var fs = require('fs');
var fileStream = fs.createReadStream(file);
fileStream.on('error', function(err) {
  console.log('File Error', err);
});
uploadParams.Body = fileStream;
var path = require('path');
uploadParams.Key = path.basename(file);
```

```
// call S3 to retrieve upload file to specified bucket
s3.upload (uploadParams, function (err, data) {
  if (err) {
    console.log("Error", err);
  } if (data) {
    console.log("Upload Success", data.Location);
  }
});
```

node s3_upload.js BUCKETNAME Filename

Archive Storage - Glacier

A slow cheap (tape-based) storage for end-of-life data (perhaps held for compliance purposes)

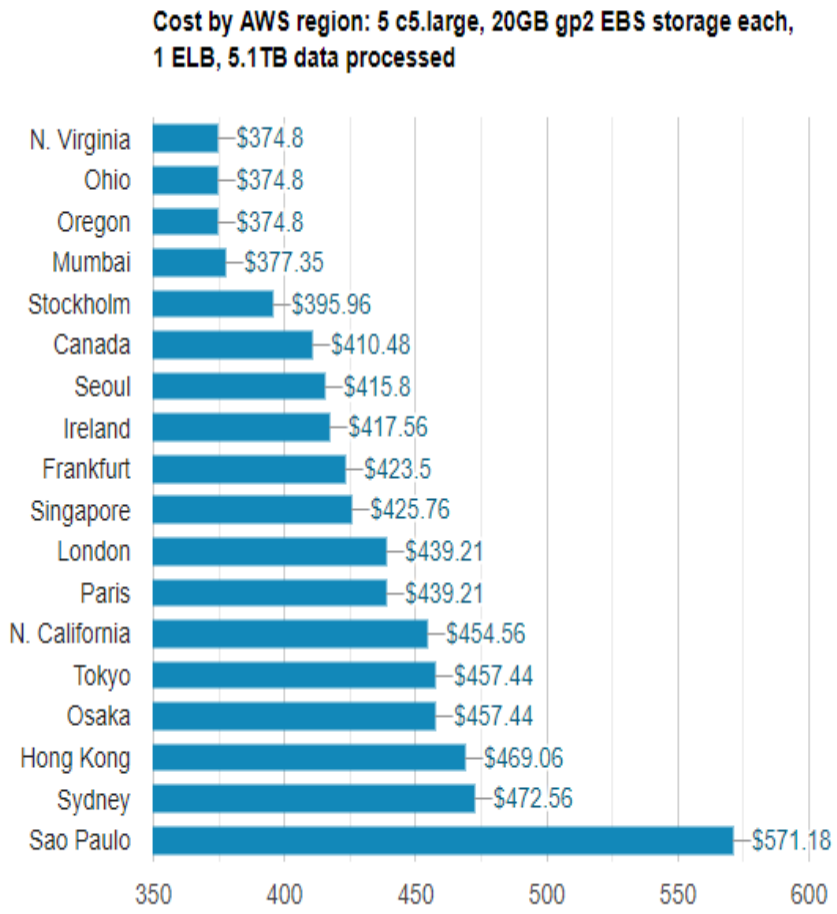
Glacier Storage Costs	$0.004 per GB/month
Retrieval Pricing (expedited)	0.01 (0.03) per GB
Retrieval Request Pricing	0.055 per 1,000 requests

Geographic Distribution

- **Amazon divides the world into 16 Geographic Regions**
  - e.g. US East, EU(Frankfurt)
- **Each region contains at least two availability zones**
  - These are essentially data center locations
  - If low latency is required, Region should be close to user
- **Things (servers, storage) are often replicated across availability zones**
- **Regions are distinct unless the user explicitly connects them**

Geographic Distribution

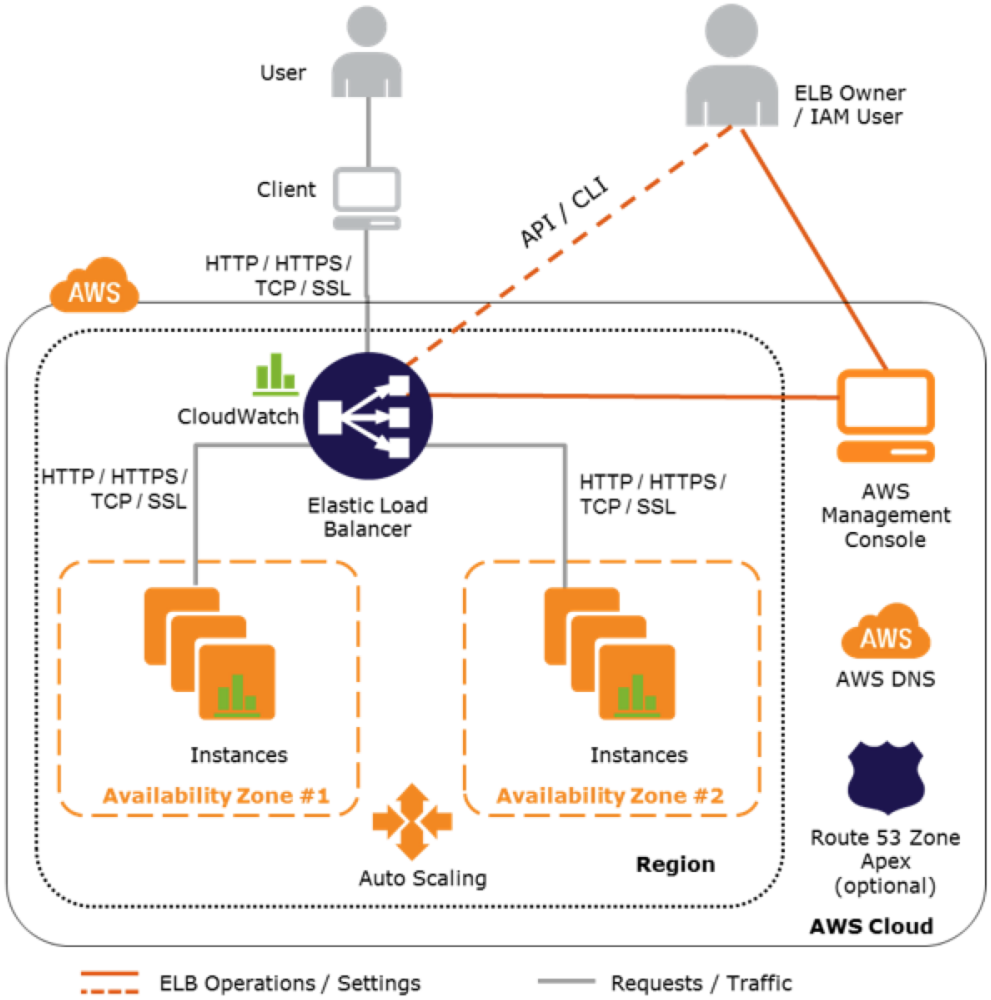
- **Pricing often varies a lot between regions**
  - Spot pricing
    - Unused capacity
    - https://aws.amazon.com/ec2/spot/
- **Factors in Choosing a location**
  - If low latency is required
  - Region should be close to user



Sharing load across Instances

- **Machine Instances can be added to a load-balancing group which means their load is passed through a LB node**
- **Application level load balancing**
  - Done at OSI layer 7
  - each HTTP request is routed to different nodes

- **Network Level load Balancing**
  - Incoming TCP connections/UDP are routed to different nodes
- **AWS Video on elastic load balancer**
  https://www.youtube.com/watch?v=YO4L_9poF3g


User
Client
ELB Owner
/ IAM User
API / CLI
HTTP / HTTPS /
TCP / SSL
AWS
CloudWatch
HTTP / HTTPS /
TCP / SSL
Elastic Load
Balancer
HTTP / HTTPS /
TCP / SSL
AWS
Management
Console
AWS
AWS DNS
Instances
Availability Zone #1
Instances
Availability Zone #2
Auto Scaling
Region
Route 53 Zone
Apex
(optional)
AWS Cloud
ELB Operations / Settings
Requests / Traffic

Scaling a Service

- **Manually**
- **Establish an Auto-Scaler and add VMs to the group**
- **Establish Launch Configuration**
  - Use an AMI ‘Golden Image”
- **Monitoring – define what constitutes a ‘Health Check’**
  - EC2 Instance Status
  - Fetching a particular URL
- **Autoscaler will terminate unhealthy instances**
- **Will start more to maintain ‘desired’ number of instances**
- **Common to use multiple ‘availability zones’**
  - so your service will survive failure of an entire data center

Cloud Database Services

- **Database Speed, Reliability and Cost are major pillars of most networked applications**
- **Historically people have used their own privately run databases based on technology like SQL or noSQL**
- **In migrating to the cloud, this is a major choice**
- **Difficult to change once the choice is made**
  - Vendor Lock-in
  - Migration issues are a significant consideration
- **Each major cloud provider prefers a different technology**

– Amazon AWS DynamoDB (also have AWS RDS for relational)

– Google Cloud BigTable

– Microsoft SQL

AWS Database Services - DynamoDB

- **noSQL Database service**
- **At highest level, DynamoDB is a Key:Value Store**
- **It stores items in tables with one or a combination of attributes stored as a primary key**
- **Tables are partitioned and replicated for high availability, reliability and performance**
- **Architecture described in 2007 SOSP paper:Dynamo: Amazon's Highly Available Key-value Store**
  https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf
- **Uses Leslie Lamport's Paxos algorithm to achieve consensus among partitions**
  https://lamport.azurewebsites.net/pubs/paxos-simple.pdf

AWS DynamoDB Tutorial

https://www.youtube.com/watch?v=2mVR_Qgx_RU (9 minutes)

DynamoDB Points to Note

- **It's a NoSQL database with very high performance**
  - can handle 10 trillion requests per day and support peaks of more than 20 million requests per second

- **Read Performance is very high (and cheap)**
- **Write Performance lower (and more expensive)**
- **Highly Replicated with "Eventual Consistency" by default**
  - Can switch on "Strong Consistency" but lower performance and more expensive

Accessing Dynamodb from Javascript

- **Amazon provides a wrapped API aws-sdk to access the DynamoDB service**

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
 AttributeDefinitions: [
  {
   AttributeName: 'CUSTOMER_ID',
   AttributeType: 'N'
  },
  {
   AttributeName: 'CUSTOMER_NAME',
   AttributeType: 'S'
  }
 ],
 KeySchema: [
  {
   AttributeName: 'CUSTOMER_ID',
   KeyType: 'HASH'
  },
  {
   AttributeName: 'CUSTOMER_NAME',
   KeyType: 'RANGE'
  }
 ],
 ProvisionedThroughput: {
  ReadCapacityUnits: 1,
  WriteCapacityUnits: 1
 },
 TableName: 'CUSTOMER_LIST',
 StreamSpecification: {
  StreamEnabled: false
 }
};

// Call DynamoDB to create the table
ddb.createTable(params, function(err, data) {
 if (err) {
  console.log("Error", err);
 } else {
  console.log("Table Created", data);
 }
});
```

Exercise

- **Create a DynamoDB Databse**
- **Populate it with items taken from an S3 bucket**
- **Allow Simple queries**
- **Destroy the database**

```
KeySchema: [
   {
    AttributeName: 'CUSTOMER_ID',
    KeyType: 'HASH'
   },
   {
    AttributeName: 'CUSTOMER_NAME',
    KeyType: 'RANGE'
   }
 ],
 ProvisionedThroughput: {
  ReadCapacityUnits: 1,
  WriteCapacityUnits: 1
 },
 TableName: 'CUSTOMER_LIST',
 StreamSpecification: {
  StreamEnabled: false
 }
};

// Call DynamoDB to create the table
ddb.createTable(params, function(err, data) {
 if (err) {
  console.log("Error", err);
 } else {
  console.log("Table Created", data);
 }
});
```