

Table of Contents

Chapter 1 - Amazon Web Services (AWS) Overview.....	9
1.1 Overview of Cloud Computing.....	10
1.2 NIST Cloud Definition.....	11
1.3 Cloud Service types.....	12
1.4 Amazon Web Services.....	13
1.5 Multi Tenancy.....	14
1.6 Shared Responsibility Model.....	15
1.7 Areas of Responsibility of AWS	16
1.8 Your Areas of Responsibility	17
1.9 AWS Value proposition.....	18
1.10 The AWS Distributed Architecture	19
1.11 AWS Services.....	21
1.12 Managed vs Unmanaged Amazon Services.....	22
1.13 Compute Services.....	23
1.14 Networking Services.....	24
1.15 Identity and Authentication Services.....	25
1.16 Storage And Content Delivery.....	26
1.17 Database Services.....	27
1.18 Messaging Services.....	28
1.19 Management and Deployment Services.....	29
1.20 Development tools.....	30
1.21 Application, Analytic, Mobile, IoT Services.....	31
1.22 Accessing AWS	32
1.23 Summary.....	33
Chapter 2 - Amazon Virtual Private Cloud and Elastic Compute Cloud.....	34
2.1 Amazon Virtual Private Cloud (VPC).....	35
2.2 VPCs, AZs, and Regions.....	36
2.3 VPC Core Components.....	38
2.4 When You Create a VPC	39
2.5 The Create VPC Dialog.....	41
2.6 The VPC Dashboard.....	42
2.7 When You Delete a VPC	43
2.8 Subnets.....	44
2.9 Private and Public Subnets.....	46
2.10 Route Tables.....	47
2.11 Routes in a Route Table Example.....	49
2.12 Internet Gateways.....	50
2.13 An Elastic IP Address.....	52
2.14 Security Groups.....	53
2.15 Security Group Rules.....	54
2.16 Example of A Security Group's Rules.....	55
2.17 Putting It All Together.....	56
2.18 Network Access Control Lists.....	57
2.19 VPC Caps.....	58

2.20 Amazon Elastic Compute Cloud (EC2)	59
2.21 AWS Marketplace for OSes.....	61
2.22 AWS Marketplace for Tools and Applications.....	62
2.23 Shop Around for Cheaper EC2 Instances.....	63
2.24 Spot Instance Pricing History.....	64
2.25 Instances Default Quotas.....	65
2.26 Accessing EC2.....	66
2.27 Amazon Elastic Block Store (EBS) Overview.....	67
2.28 EBS Store Details.....	69
2.29 Instance Tagging.....	71
2.30 Newer Tag Features.....	72
2.31 EC2 Instance Types.....	73
2.32 The Instance Types Matrix.....	75
2.33 The T2 Instance Type (Example of a Low-end Type).....	78
2.34 The I2 Instance Type (Example of a High-end Type).....	79
2.35 X1 Instance	80
2.36 Modifying an Instance	81
2.37 The EC2 Dashboard.....	83
2.38 EC2 Pricing	84
2.39 Cluster Networking.....	87
2.40 Dedicated Instances.....	88
2.41 VM Import / Export to/from AWS.....	89
2.42 Elastic IP Address	90
2.43 EC2 Service Level Agreement.....	91
2.44 Summary.....	93
Chapter 3 - AWS Identity and Access Management.....	94
3.1 AWS Identity and Access Management (IAM).....	95
3.2 IAM Groups.....	96
3.3 Working with IAM.....	97
3.4 The IAM Dashboard.....	98
3.5 IAM Principals.....	99
3.6 Root Account Access vs. IAM User Access	100
3.7 Roles.....	101
3.8 Creating a Role in AWS Management Console.....	102
3.9 Accessing AWS.....	103
3.10 Identity Providers.....	105
3.11 Need Identity Management for Mobile Apps?	107
3.12 AWS Key Management Service (KMS).....	109
3.13 User Management	111
3.14 Password Policies.....	113
3.15 Using Multi-Factor Authentication Devices	114
3.16 Hardware-based and Virtual MFA Devices.....	115
3.17 Summary.....	116
Chapter 4 - AWS Simple Storage Service	117
4.1 What is AWS Simple Storage Service (S3).....	118
4.2 AWS Storage.....	119

4.3 Regions.....	121
4.4 S3 Regions.....	122
4.5 Getting started with S3	123
4.6 Using BitTorrent.....	124
4.7 More on Buckets.....	125
4.8 Bucket Configurable Properties	126
4.9 Advanced S3 Bucket Properties.....	127
4.10 The Bucket Creation Dialog in the Management Console.....	128
4.11 Bucket Permissions.....	129
4.12 Bucket-level Operations.....	131
4.13 Authorization of REST Requests.....	133
4.14 Adding Cross-Origin Resource Sharing Configuration	134
4.15 Event Notifications.....	135
4.16 The Requester Pays Option	136
4.17 The Object Key	137
4.18 Object Versioning.....	138
4.19 Example of Object Properties.....	140
4.20 Object Storage Class Levels.....	141
4.21 Object Life-cycle Configuration.....	142
4.22 Amazon S3 Data Consistency Model.....	144
4.23 Eventually Consistent Reads vs Consistent Reads.....	145
4.24 Amazon S3 Security.....	146
4.25 S3 Use Case: Backup and Archiving.....	147
4.26 Another S3 Use Case: Static Web Hosting.....	148
4.27 S3 Use Case: Disaster Recovery.....	149
4.28 AWS S3 Pricing.....	150
4.29 Amazon S3 Transfer Acceleration.....	151
4.30 Amazon S3 SLA Definitions	153
4.31 Amazon S3 SLA Service Commitment.....	154
4.32 S3 Service Limits - Buckets.....	155
4.33 S3 Service Limits.....	156
4.34 Summary.....	157
Chapter 5 - Caching and Content Delivery Network Services.....	158
5.1 CloudFront Content Delivery Network.....	159
5.2 CloudFront Features.....	160
5.3 CloudFront Use Cases.....	161
5.4 Accessing CloudFront.....	162
5.5 Amazon ElastiCache	163
5.6 The Redis and Memcached Engines.....	164
5.7 Summary.....	165
Chapter 6 - AWS Databases.....	166
6.1 AWS Database Services.....	167
6.2 The CAP Theorem	168
6.3 Mechanisms to Guarantee a Single CAP Property.....	169
6.4 NoSQL Systems CAP Triangle.....	171
6.5 Relational vs NoSQL Databases.....	172

6.6 AWS Databases Overview	173
6.7 AWS RDS Engine Options Page in the Management Console.....	175
6.8 Database Instances.....	176
6.9 Multi-AZ Deployment	177
6.10 Two-AZ Master-Slave RDS Solution Architecture.....	178
6.11 Licensing.....	179
6.12 Migrating Your On-Premise Databases to RDS.....	180
6.13 Aurora.....	181
6.14 Aurora DB Instance Details In AWS Console.....	183
6.15 Redshift	184
6.16 Amazon RDS Use Cases.....	186
6.17 RDS Service Limits.....	187
6.18 Summary.....	189
Chapter 7 - Monitoring and Managing Applications in AWS.....	190
7.1 AWS Management Tools Overview.....	191
7.2 AWS Well Architected Framework.....	192
7.3 AWS Well Architected Framework Pillars.....	193
7.4 Operational Excellence:	194
7.5 Security.....	195
7.6 Reliability.....	196
7.7 Performance Efficiency.....	197
7.8 Cost Optimization.....	198
7.9 What is CloudWatch.....	199
7.10 How CloudWatch Works.....	200
7.11 Use Cases.....	201
7.12 Log Management.....	202
7.13 Amazon CloudWatch in the AWS Management Console.....	203
7.14 CloudWatch AWS Services Integration.....	204
7.15 Monitoring EC2 Instances.....	205
7.16 Collecting Metrics and Logs from EC2 Instances and On-Premises Servers with the CloudWatch Agent.....	206
7.17 CloudWatch Alarms.....	208
7.18 Alarms in the AWS Management Console Examples.....	209
7.19 Amazon CloudWatch Events	210
7.20 AWS CloudTrail.....	211
7.21 CloudTrail Dashboard in the AWS Management Console.....	213
7.22 CloudTrail Integration with CloudWatch.....	215
7.23 How CloudTrail Works.....	216
7.24 CloudTrail Use Cases.....	217
7.25 Trusted Advisor.....	218
7.26 Trusted Advisor Dashboard.....	219
7.27 Summary.....	221
Chapter 8 - Amazon DynamoDB.....	222
8.1 Main NoSQL Database Storage Types.....	223
8.2 Amazon DynamoDB.....	225
8.3 DynamoDB is an Eventual Consistent NoSQL Database.....	226

8.4 DynamoDB Tables are Created in and Tied to a Region	227
8.5 Accessing DynamoDB.....	228
8.6 DynamoDB Core Components.....	229
8.7 Data Types.....	231
8.8 Example of an Item with Attributes.....	232
8.9 The Primary Key.....	233
8.10 Partition Key and Sort Key	234
8.11 Example of a Table with Partition Key and Sort Key	236
8.12 Secondary Indexes.....	237
8.13 Provisioned Capacity.....	238
8.14 Partition Metrics.....	240
8.15 Maximizing DynamoDB Throughput.....	241
8.16 Summary.....	242
Chapter 9 - AWS Storage Options.....	243
9.1 Storage Types in AWS.....	244
9.2 Overview of AWS Storage Offerings.....	245
9.3 Introduction to EFS.....	246
9.4 EFS Characteristics.....	247
9.5 Introduction to Storage Gateway.....	248
9.6 Storage Gateway Types.....	249
9.7 Data transfer to/from AWS.....	250
9.8 Introduction to Snowball and Snowmobile.....	251
9.9 Summary.....	252
Chapter 10 - Elastic Compute Cloud High Availability.....	253
10.1 Amazon Elastic Compute Cloud (EC2)	254
10.2 High Availability in EC2.....	255
10.3 Challenges of High Availability.....	256
10.4 Challenges solved.....	257
10.5 Elastic Load Balancing.....	258
10.6 Classic Load Balancer.....	259
10.7 Application Load Balancer.....	260
10.8 Network Load Balancer.....	261
10.9 Scaling	262
10.10 EC2 AutoScaling.....	263
10.11 Summary.....	265
Chapter 11 - Orchestration in AWS.....	266
11.1 Overview of Orchestration in AWS.....	267
11.2 Why Orchestration?.....	268
11.3 Infrastructure as Code.....	269
11.4 CloudFormation.....	270
11.5 CloudFormation Template.....	271
11.6 Structure of a CF Template.....	272
11.7 Example CF Template - Parameters.....	274
11.8 Example CF Template - Resources.....	275
11.9 Example CF Template - Outputs.....	276
11.10 OpsWorks.....	277

11.11 OpsWorks Use-cases.....	278
11.12 OpsWorks Operations.....	279
11.13 Elastic Beanstalk.....	280
11.14 Elastic Beanstalk Components.....	281
11.15 Elastic Beanstalk Architecture.....	282
11.16 Elastic Beanstalk Supported Platforms.....	283
11.17 Comparison of deployment modes.....	284
11.18 Summary.....	285
Chapter 12 - DNS in AWS.....	286
12.1 Introduction to Route53.....	287
12.2 Route 53 hosted zones.....	288
12.3 Route 53 Routing Policies.....	289
12.4 Route 53 DNS Health Checks.....	290
12.5 Summary.....	291
Chapter 13 - Passing the exam.....	292
13.1 Exam Overview.....	293
13.2 Knowledge Prerequisites.....	294
13.3 Exam Structure.....	295
13.4 Knowledge Domains	296
13.5 Domain 1: Cloud Concepts.....	297
13.6 Domain 2: Security.....	298
13.7 Domain 3: Technology.....	299
13.8 Domain 4: Billing and Pricing.....	300
13.9 Preparing for the exam.....	301
13.10 Additional Exam Resources.....	302
13.11 Summary.....	303

Chapter 1 - Amazon Web Services (AWS) Overview

Objectives

Key objectives of this module

- Overview of Cloud Computing
- AWS Value Propositions
- Regions and Availability Zones
- The Shared Responsibility Model
- Overview of AWS Services – too much detail, not all services covered
- AWS Management Console



1.1 Overview of Cloud Computing

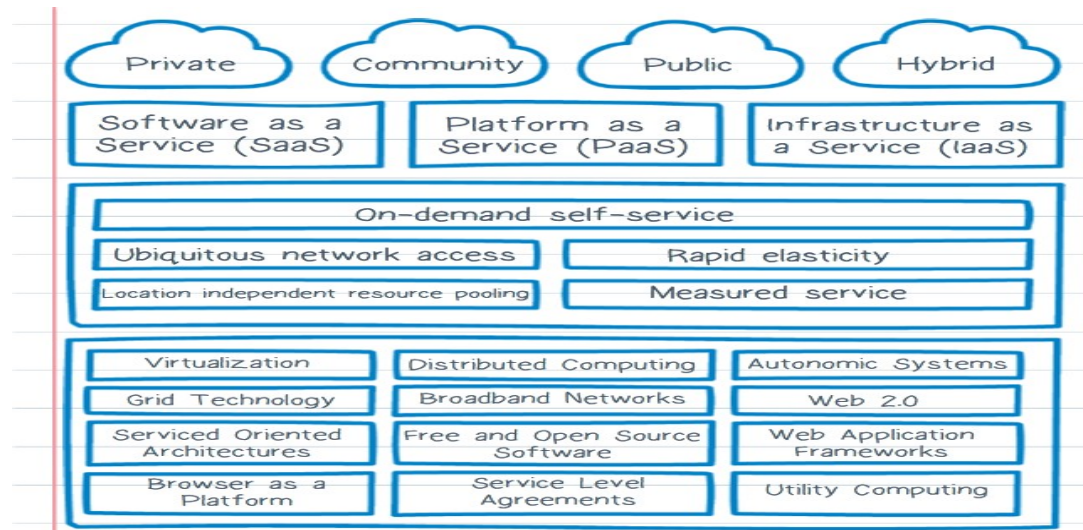
- Delivering IT resources “on demand”
- Utilizing cloud “vendors” like AWS, Azure, GCP etc.
- CapEX vs OpEX
- Geographically distributed





1.2 NIST Cloud Definition

- On-demand self-service solution
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

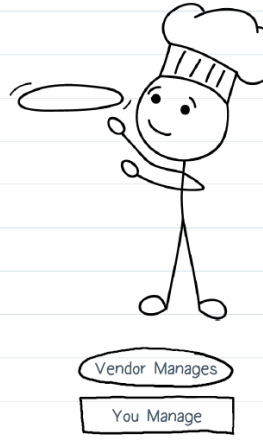




1.3 Cloud Service types

- IaaS
- PaaS
- SaaS
- XaaS

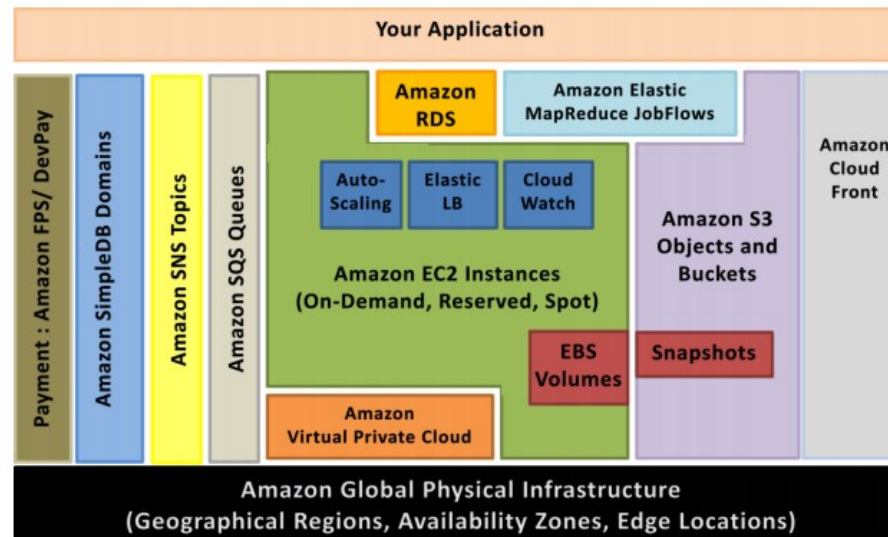
Pizza-as-a-Service

Traditional On-Premises	Infrastructure as a Service (IaaS)	Platform as a Service (PaaS)	Software as a Service (SaaS)	
Dining Table	Dining Table	Dining Table	Dining Table	
Soda	Soda	Soda	Soda	
Electric/Gas	Electric/Gas	Electric/Gas	Electric/Gas	
Oven	Oven	Oven	Oven	
Fire	Fire	Fire	Fire	
Pizza Dough	Pizza Dough	Pizza Dough	Pizza Dough	
Tomato Sauce	Tomato Sauce	Tomato Sauce	Tomato Sauce	
Toppings	Toppings	Toppings	Toppings	
Cheese	Cheese	Cheese	Cheese	
Made at home	Take & Bake	Delivery	Dining Out	



1.4 Amazon Web Services

- Market leader in public cloud services
- Offers a variety IaaS, PaaS and SaaS services
- Global reach with 20+ regions 160+ edge locations
- All services addressable via API calls
- Ability to automate all aspects of your application





1.5 Multi Tenancy

- Sharing infrastructure with other users
- Visualization and containers
- Tenant isolation
- The noisy neighbor problem





1.6 Shared Responsibility Model

- While AWS is responsible for providing cloud-grade infrastructure with the perimeter security and intrusion detection in place, you are responsible for security of your accounts, networks, and applications, including
 - ◇ User access control
 - ◇ User roles
 - ◇ Application passwords
 - ◇ Instance OS patching
 - ◇ Network configuration (public / private)



1.7 Areas of Responsibility of AWS



Source: <http://cloudacademy.com/blog/aws-shared-responsibility-model-security/>



1.8 Your Areas of Responsibility



Source: <http://cloudacademy.com/blog/aws-shared-responsibility-model-security/>



1.9 AWS Value proposition

- With AWS we can take advantage of:
 - ◇ On-demand infrastructure provisioning
 - ◇ Scalable and elastic resource capabilities
 - ◇ CapEX over OpEX
 - ◇ Pay-as-you go (usage-based) costing model
 - ◇ Shorter time to market for your applications
 - ◇ Global-as-local reach
 - The global network of AWS Edge locations consists of 160+ points of presence worldwide
 - ◇ Ability to experiment at zero or very low cost
 - ◇ AWS's culture of innovation



1.10 The AWS Distributed Architecture

- The AWS cloud platform is deployed across a number of geographical regions (20 as of 2019)
- Each region includes two or more "Availability Zones" (AZ), which are isolated from each other data centers (60 AZ's as of 2019)
 - ◇ Some Amazon services can operate across AZ's (e.g. S3)
 - ◇ Other services require you to set up and configure replication across AZ's to achieve service resilience against data center outages



1.11 AWS Services

- As of 2019, AWS has around 100 services, including compute, security and identity, storage, networking, relational databases and NoSQL solutions, analytics, the IoT, etc.
- AWS uses a number of ODM (original design manufacturer) products designed and developed per Amazon's specs
- Not all services are available in all regions
- In subsequent slides, we will quickly review some of the commonly used services



1.12 Managed vs Unmanaged Amazon Services

- Depending on the type of an AWS service, its scalability and high availability properties are handled either by
 - ◇ AWS (the managed services) or
 - ◇ By you (the unmanaged services)



1.13 Compute Services

- The main services in this category include:
- Elastic Compute Cloud (EC2)
 - ◇ Auto Scaling
 - ◇ Elastic Load Balancing (ELB)
- Elastic Beanstalk
- Elastic Container Service (ECS)
 - ◇ Elastic Container Registry (ECR)
- AWS Lambda
 - ◇ AWS Step Functions



1.14 Networking Services

- Virtual Private Cloud (VPC)
 - ◇ IGW and NAT
 - ◇ Security Groups and NACLs
- VPN with Virtual Gateway (VGW) and DirectConnect
- Amazon Route53
- AWS Shield
- AWS WAF



1.15 Identity and Authentication Services

- Identity and Access Management (IAM)
- AWS Directory Service
- AWS Certificate Manager
- AWS Key Management Service (KMS)
- AWS CloudHSM



1.16 Storage And Content Delivery

- Elastic Block Storage (EBS)
 - ◇ Storage attached as virtual disk(s) to your EC2 machines
- Simple Storage Service (S3)
 - ◇ Durable, highly-scalable object (file) storage
- Glacier
 - ◇ An archival service; a cost effective albeit slow alternative to S3
- CloudFront
 - ◇ A global accelerated content delivery service (CDS) for static and/or streaming content, e.g. S3 objects



1.17 Database Services

- Relational Database Service (RDS)
- DynamoDB
 - ◇ Cloud-grade NoSQL system
- ElastiCache
 - ◇ Compatible with Redis and Memcached
- Redshift
 - ◇ AWS petabyte-scale data warehouse system



1.18 Messaging Services

- Simple Queue Service (SQS)
- Simple Notifications Service (SNS)
- Simple Email Service (SES)
- Amazon MQ
- **Note:** These are PaaS-type of capabilities



1.19 Management and Deployment Services

- CloudWatch
- CloudTrail
- AWS Config
- Trusted Advisor
- AWS Systems Manager
- CloudFormation
- OpsWorks



1.20 Development tools

- Cloud9
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- CodeStar



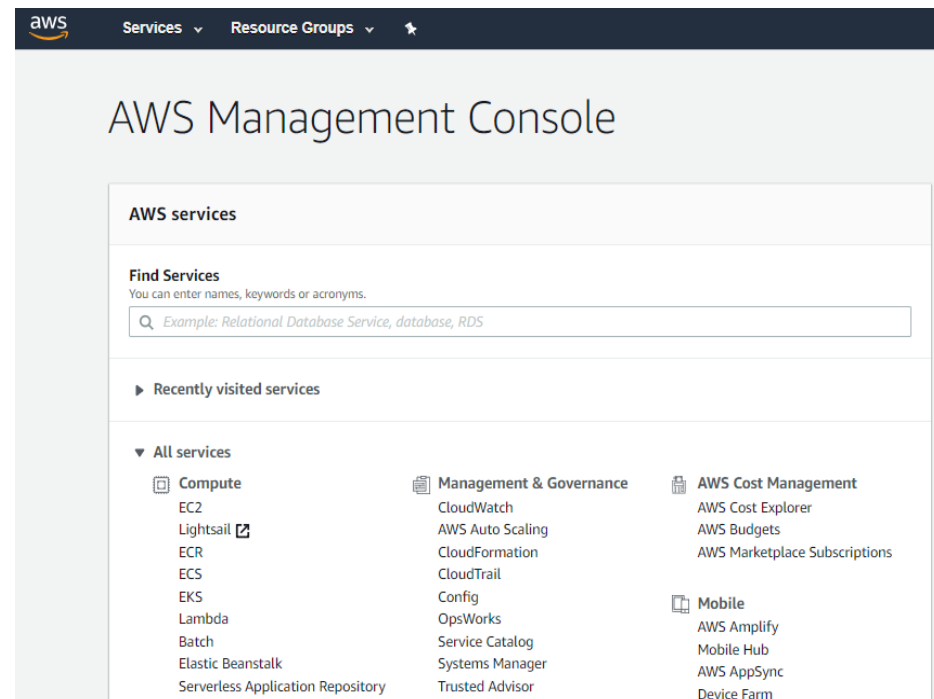
1.21 Application, Analytic, Mobile, IoT Services

- API Gateway
- Amazon SWF
- EMR
- DataPipeline
- Kinesis
- ElastiSearch
- Athena
- QuickSight
- Cognito
- Mobile Analytics
- Device Farm
- IoT
- ML
- SageMaker



1.22 Accessing AWS

- You have the following means of accessing AWS:
 - ◇ **The AWS Management Console (Web UI)**
 - ◇ The AWS Command-line Interface (AWS CLI)
 - ◇ The AWS Software Development Kits (SDKs)





1.23 Summary

- The AWS Cloud offers a variety of services that you can leverage when building your solutions in the cloud
- Efficiencies offered by AWS can be used to run your business very efficiently
- AWS offers agility and the ability to mix and match managed and unmanaged services to deliver

Chapter 2 - Amazon Virtual Private Cloud and Elastic Compute Cloud

Objectives

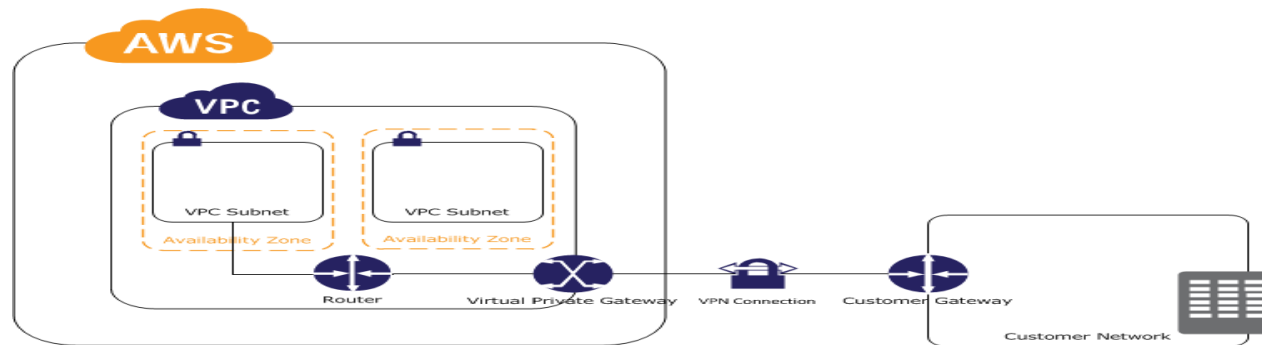
Key objectives of this chapter

- Overview of
 - ◇ Amazon Virtual Private Cloud (VPC), and
 - ◇ Elastic Compute Cloud (EC2)



2.1 Amazon Virtual Private Cloud (VPC)

- Amazon VPC is an isolated virtual network of the AWS cloud
- Used for placement of EC2 instances, Amazon RDS databases, and other resources provisioned by you





2.2 VPCs, AZs, and Regions

- You can create multiple VPCs within a region (subject to limits currently in effect, discussed later)
- VPC spans all the Availability Zones in the region
- After creating a VPC, you can add one or more uniquely identified subnets in each Availability Zone, and subnets cannot span Zones
 - ◇ Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location



2.3 VPC Core Components

- A VPC consists of the following core components:
 - ◇ Subnets
 - ◇ Route tables
 - ◇ Security groups — Act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level
 - ◇ Network access control lists (ACLs) — Act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level
 - ◇ Flow logs — Capture information about the IP traffic going to and from network interfaces in your VPC
 - ◇ DHCP Options Sets



2.4 When You Create a VPC ...

- You need to specify an IPv4 address range using a Classless Inter-Domain Routing (CIDR) block of non-routable for public Internet IP addresses
 - ◇ For example, 10.0.0.0/16
- The allowed IP address space within a VPC is between 16 (with a netmask of /28) and 65,536 (with a netmask of /16)
- You cannot change the IP address range of a VPC once it was created
- The IP address ranges must not overlap those of any other VPC with which the VPC is to be connected



2.5 The Create VPC Dialog

Create VPC ✕

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You must specify an IPv4 address range for your VPC. Specify the IPv4 address range as a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. You cannot specify an IPv4 CIDR block larger than /16. You can optionally associate an Amazon-provided IPv6 CIDR block with the VPC.

Name tag ⓘ

IPv4 CIDR block* ⓘ

IPv6 CIDR block* ☒ No IPv6 CIDR Block ⓘ
☐ Amazon provided IPv6 CIDR block

Tenancy ⓘ

Cancel Yes, Create

- Tenancy can be **Default** or **Dedicated** (single-tenant hardware)



2.6 The VPC Dashboard

Create VPC		Actions									
Search VPCs and their properties										<< 1 to 2 of 2 VPCs >>	
	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set	Route table	Network ACL	Tenancy	Default VPC	
	UserVPC	vpc-d87125bc	available	172.16.0.0/24		dopt-d37394b6	rtb-043e3e60	acl-9bc9e6ff	Default	No	
	SystemDefault-VPC	vpc-0a66c06f	available	172.31.0.0/16		dopt-d37394b6	rtb-11c86574	acl-9c8627f9	Default	Yes	



2.7 When You Delete a VPC ...

- Do Not Delete the Default VPC!
 - ◇ Restoring it is not a simple process
- Deleting a custom VPC also deletes core and optional components associated with this VPC:
 - ◇ Subnets
 - ◇ Security Groups
 - ◇ Network ACLs
 - ◇ Internet Gateways
 - ◇ Route Tables
 - ◇ Network Interfaces



2.8 Subnets

- In every AWS region, you are given a choice of a default VPC with one subnet in each Availability Zone (AZ)
 - ◇ You can add your custom subnets in each AZ
 - ◇ Subnets cannot span AZs
 - ◇ So there is one-to-one relationship between a subnet and an AZ
- The smallest subnet you can have is netmasked with /28 (16 IP addresses)
- AWS reserves the last IP address and the first four IP addresses for its own needs, so you are left with 5 fewer IPs out of the IP range specified by your subnet's CIDR block
- The default subnet has a netmask of /20



2.9 Private and Public Subnets

- Subnets can be created public or private
- Public subnets are reachable from the public Internet
- Private subnets are only visible within their VPC
- Resources created in a subnet are assigned private IPs that are non-routable on the Internet



2.10 Route Tables

- A route table contains configurable rules that control the traffic for subnets within a VPC
- These traffic routing rules are called **routes**
- Every VPC automatically comes with a main route table
- Each subnet must be associated with a route table; if you don't specify one, the main route table will be used
- Each route table contains a default route called the **local route** which enables communication within your VPC
 - ◇ This route cannot be deleted or modified
- You can create your own custom routes
- A route specifies a destination CIDR and a target



2.11 Routes in a Route Table Example

subnet-cc50af95

Summary Route Table Network ACL Flow Logs

Edit

Route Table: [rtb-11c86574](#)

Destination	Target
172.31.0.0/16	local
0.0.0.0/0	igw-efcc028a



2.12 Internet Gateways

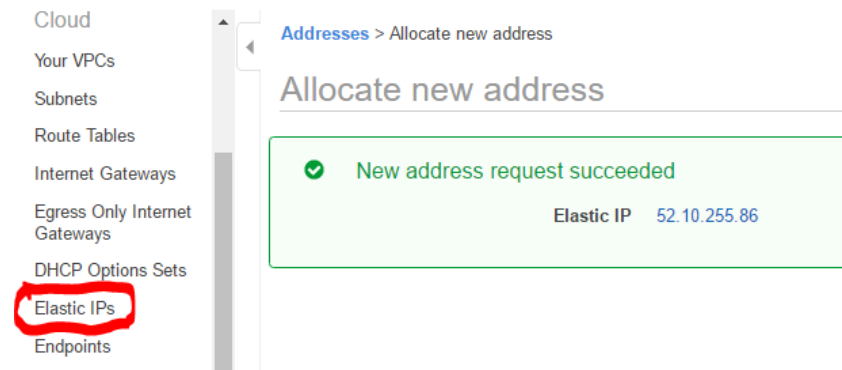
- An Internet Gateway (IGW) is a VPC component that allows network traffic between your EC2 instances and the Internet
- An IGW makes your subnet public
- An IGW is used as a target in the route table, e.g.:

`0.0.0.0/0 igw-efcc028a`
- IGWs also perform bi-directional NAT'ing for instances that have been assigned a public IP, or an Elastic IP
 - ◇ Each EC2 instance on creation is assigned a private non-routable on the public Internet IP address that needs to be translated into the public one for communicating with the outside world



2.13 An Elastic IP Address

- An Elastic IP Address (EIP) is a static public IP address that you can allocate on demand from a regional pool of addresses maintained by AWS and assign it to one of your EC2 instances
- You first allocate an EIP in your VPC, then you can assign it to your EC2 instance



- EIPs can be moved between EC2 instances within your VPCs created within the same region
- You cannot share EIPs across regions



2.14 Security Groups

- A security group is a virtual firewall that controls inbound and outbound network traffic to your VPCs through firewall rules
- If you don't create a custom security group in your VPC, a default security group will be attached to your EC2 instances
- You can declaratively change the rules
- You can only specify ALLOW rules, but not DENY rules
 - ◇ You can specify DENY rules with NACLs (discussed later)
- By default, no inbound traffic is allowed, and all outbound traffic is allowed
- Security groups are stateful in that return traffic is automatically allowed regardless of the rules applied to the opposite traffic direction
- Your custom security groups, by default, do not allow traffic between EC2 instances launched in them
 - ◇ The default security group does allow such traffic



2.15 Security Group Rules

- For each rule, you specify the following:
 - ◇ **Protocol:** The allowed protocol
 - The most common protocols are TCP, UDP, and ICMP
 - ◇ **Port range:** You can specify a single port number (for example, 22 for SSH), or a range of port numbers (for example, 17000-18000)
 - ◇ **ICMP type and code:** The ICMP type and code
 - ◇ **Source or destination:** The traffic source (inbound rules) or traffic destination (outbound rules)



2.16 Example of A Security Group's Rules

sg-b44e07d1

Summary

Inbound Rules

Outbound Rules

Edit

Type	Protocol	Port Range	Source
SSH (22)	TCP (6)	22	sg-b44e07d1
HTTP* (8080)	TCP (6)	8080	0.0.0.0/0

sg-b44e07d1

Summary

Inbound Rules

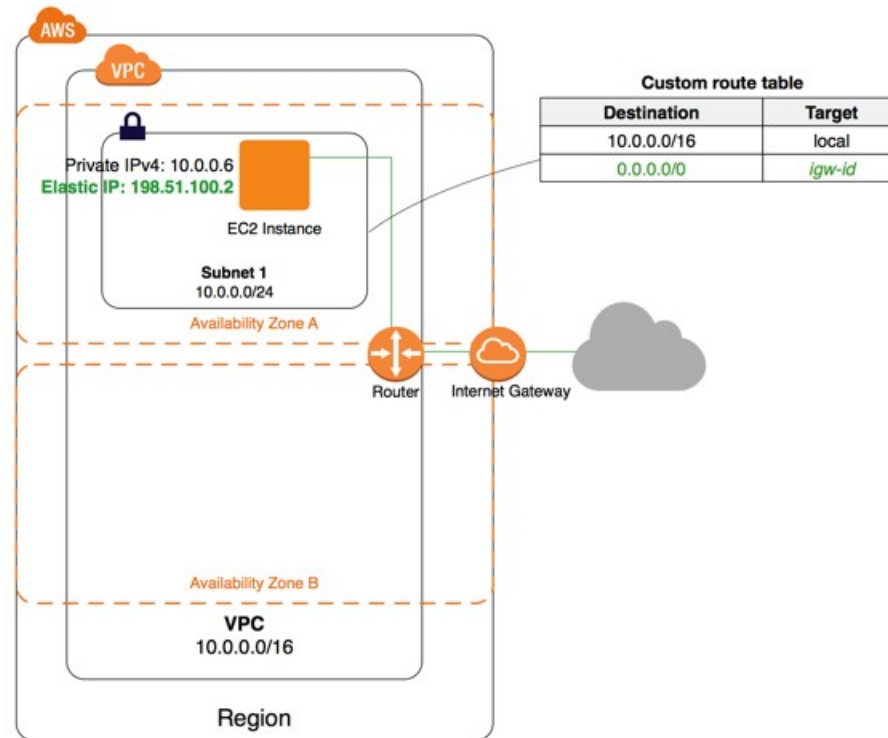
Outbound Rules

Edit

Type	Protocol	Port Range	Destination
ALL Traffic	ALL	ALL	0.0.0.0/0



2.17 Putting It All Together



Source: http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Internet_Gateway.html



2.18 Network Access Control Lists

- Network Access Control Lists (NACLs) are stateless firewalls attached to subnets
- NACLs provide a second layer of security to your VPCs on top of security groups (the first security layer)
- A NACL contains a list of numbered rules for allowing or denying network traffic that are evaluated in ascending order
- VPCs come with a default NACL that allows all traffic in both directions
- You can create your own NACLs where, by default, all inbound and outbound traffic is blocked
- In contrast to security groups, NACLs also support DENY rules
- In contrast to security groups, NACL rules are stateless: you need to explicitly allow return traffic
- NACL rules are applied to all EC2 instances in the associated subnet
 - ◇ Security groups are applied on a per-instance basis



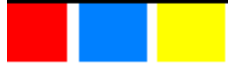
2.19 VPC Caps

- According to Amazon VPC documentation (<https://aws.amazon.com/vpc/details/>), the following caps are in effect (as of 2017):
 - *You can have up to five (5) nondefault Amazon VPCs per AWS account per region.**
 - *You can have up to four (4) secondary IP ranges per VPC**
 - *You can create up to two hundred (200) subnets per Amazon VPC.**
 - *You can have up to five (5) Amazon VPC Elastic IP Addresses per AWS account per region.**
 - *You can have up to ten (10) Hardware VPN Connections per Amazon VPC.**
 - ◇ ** Should you need to exceed these limits, you complete a form like the one below available here*
 - ◇ *https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Appendix_Limits.html*









2.20 Amazon Elastic Compute Cloud (EC2)

- Amazon EC2 provides web-scale compute capability
- EC2 instances (deployed as virtual machines) are bootable from an Amazon Machine Image (AMI) of the operating system of your choice, which may optionally have pre-installed software and associated configuration files
 - ◇ Currently, EC2 offers a selection of 3000+ AMI's of popular Open Source and commercial software, available from the Amazon's marketplace
 - ◇ You can also create your own AMI
- You have a wide range of supported OS images to chose from:
 - ◇ Various Linux distros
 - ◇ Windows
- Security is managed through security groups (virtual firewalls)



2.21 AWS Marketplace for OSes

	Amazon Linux AMI (HVM / 64-bit) Amazon Web Services \$0.013 to \$8.14/hr incl EC2 charges		Red Hat Enterprise Linux (RHEL) 7 (...) Amazon Web Services \$0.073 to \$8.27/hr incl EC2 charges
	CentOS 7 (x86_64) with Updates HVM Centos.org \$0.00/hr for software		SUSE Linux Enterprise Server 11 (HVM) AWS MP Catalog \$0.023 to \$8.24/hr incl EC2 charges
	Ubuntu Server 14.04 LTS (HVM) Canonical Group L... \$0.00/hr for software		Microsoft Windows Server 2012 RTM Amazon Web Services \$0.018 to \$9.348/hr incl EC2 charges



2.22 AWS Marketplace for Tools and Applications

Software Infrastructure	Business Software
Application Development	Business Intelligence
Application Servers	Financial Services
Application Stacks	Collaboration
Big Data	Content Management
Databases & Caching	CRM
Network Infrastructure	eCommerce
Operating Systems	Education & Research
Security	High Performance Computing
Developer Tools	Media
Issue & Bug Tracking	Project Management
Monitoring	Storage & Backup
Source Control	
Testing	



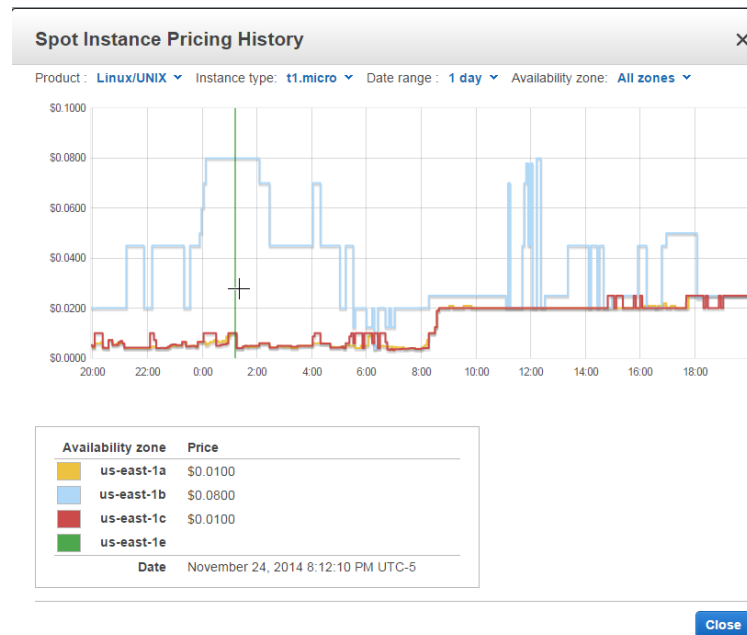
2.23 Shop Around for Cheaper EC2 Instances...

- You have a choice of
 - ◇ On-demand instances (regular price choice; more on that later...)
 - You pay for the running instances by the hour
 - ◇ Reserved instances (a cheaper choice; good for steady usage patterns)
 - You make a low (compared with the on-demand instance), one-time upfront payment
 - ◇ Spot instances (the cheapest choice)
 - You can bid for unused EC2 capacity, and get it when the price slides below the desired threshold



2.24 Spot Instance Pricing History

- The spot instance pricing fluctuates over a period of time and availability zone
- You can check the Pricing History in the EC2 dashboard to see the trend to better prepare for pricing changes



- You can select a time horizon (from 1 day up to 3 months), instance OS, and EC2 instance type per availability zone



2.25 Instances Default Quotas

- Per region, you can run up to
 - ◇ 20 *on-demand* instances
 - ◇ Buy and run 20 *reserved* instances
 - ◇ Make a request for and run 5 *spot* instances
- Should you wish to provision more instances, complete the Amazon EC2 instance request form



2.26 Accessing EC2

- AWS Management Console
- The AWS command-line tool (CLI)
 - ◇ You can run AWS commands on EC2 instances in the cloud, or on local Linux or Windows machines
 - See <https://docs.aws.amazon.com/cli/latest/reference/ec2/index.html> for more details
- SSH for Linux flavors
- Remote Desktop Protocol (RDP) for Windows®



2.27 Amazon Elastic Block Store (EBS) Overview

- The Elastic Block Store (EBS) is a network-linked persistent storage volume that you can attach to EC2 instances
- An EBS volume is seen by your EC2 instance as a local hard drive
- Launched EC2 instances can also receive a free local instance store of limited capacity (10GiB) which is ephemeral and gets recycled (totally erased) when the server is stopped
 - ◇ You should use the local instance store only for temporary data that don't require durable persistence
 - ◇ Instance store volumes are created from a template stored in Amazon S3 (which negatively affects the boot time)
- The truly persistent (that survive instance restart) EBS volumes are referred to as Amazon EBS-backed; those volumes have a maximum storage of 10TiB
 - ◇ **Note:** You will pay for this storage even if your instance to which the volume is attached is stopped



2.28 EBS Store Details

- EBS offers raw, unformatted block-level storage
 - ◇ After you have attached an EBS volume, you will need to format it and create a file system on the device before you can start using it
- You may experience some I/O delays (5 to 50 %) when starting to use a newly added EBS volume
 - ◇ You may also experience spikes in CPU usage during such times
 - ◇ Be aware of this as EBS often serves as your boot volumes in EC2 for operating systems
- An EBS volume must be created in the same Availability Zone as the EC2 instance you are going to attach the volume to
- Each EBS volume is automatically replicated within its Availability Zone (AZ) offering high guarantees of data durability and protection against loss of your data



2.29 Instance Tagging

- A good operational practice is to tag your instances with *key=value* pairs (done with a couple of mouse clicks in the *EC2 Dashboard*)
- You can apply a maximum of 50 tags per instance, was 10 per instance until early 2017
- Tag samples:
 - ◇ Environment = UAT
 - ◇ Project_name = Keystone
 - ◇ Version = 1.29
- Tags are used for
 - ◇ Track ownership
 - ◇ Drive their cost accounting processes
 - ◇ Implement compliance protocols
 - ◇ Control access to resources via IAM policies



2.30 Newer Tag Features

- Tag on Creation – You can now specify tags for EC2 instances and EBS volumes as part of the API call that creates the resources
- Enforced Tag Usage – Ability to author IAM policies that mandate the use of specific tags on EC2 instances or EBS volumes
- Resource-Level Permissions – CreateTags and DeleteTags functions now support IAM's resource-level permissions
- Enforced Volume Encryption – Ability to Author IAM policies that mandate the use of encryption for newly created EBS volumes



2.31 EC2 Instance Types

- AWS Cloud offers a wide selection of EC2 instance types to help you match different use cases with optimum computer capacity
- Instance types include such parameters as:
 - ◇ CPU, RAM, instance storage, and networking capacity (all resources are virtual!)
- Amazon recommends to measure the actual performance of your application by running it under load to identify the suitable instance type and validate your solution architecture
- Most instance types are based on Intel Xeon processor



2.32 The Instance Types Matrix

- EC2 instance types are grouped by instance families that have similar base run-time profiles suitable for specific use cases
 - ◇ Within each family, you have models with more vCPUs, RAM and storage
- Main families:
 - ◇ **T2/T3** (General Purpose Category)
 - ◇ **M4/M5/M5a/M5d** (General Purpose Category)
 - ◇ **C4/C5/C5d/C5n** (Compute Optimized Category)
 - ◇ **R4/R5/R5aR5d and X1** (Memory Optimized Category)
 - ◇ **I3, D2 and H1** (Storage Optimized Category)
 - ◇ **G2/G3, P2/P3 and F1** (Accelerated Computing Category)
- **Note:** The original T1, M1, C1, CC2, M2, CR1, CG1, and H1 instance types have been deprecated



2.33 The T2 Instance Type (Example of a Low-end Type)

- The T2 type is used for low compute capacity instances
- This type is suitable for development, prototyping, build servers, and small web applications
- T2 instances are positioned as instances with "Burstable Performance", meaning that they offer a baseline level of CPU performance with the ability to burst above the baseline for a short period of time

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



2.34 The I2 Instance Type (Example of a High-end Type)

- The I2 type is storage optimized with fast SSD-backed instance storage optimized for random I/O performance
- This type gives you an option of very cost-efficient IOPS rates and enhanced networking with great packet per second (PPS) rate and low latencies
- Use cases: NoSQL systems (MongoDB, Cassandra, Hadoop, etc.)

Model	vCPU	Mem (GiB)	Storage (GB)
i2.xlarge	4	30.5	1 x 800 SSD
i2.2xlarge	8	61	2 x 800 SSD
i2.4xlarge	16	122	4 x 800 SSD
i2.8xlarge	32	244	8 x 800 SSD



2.35 X1 Instance

- Introduced in May 2016
- Comes with 2 TB DDR4 based memory, 128 vCPUs, and 10 Gbps of dedicated bandwidth to Amazon EBS
- Suitable for computationally and memory intensive enterprise-class jobs
 - ◇ Apache Spark
 - ◇ SAP HANA
 - ◇ etc.
- You can get it for \$3.97 / hr with a 3-year plan



2.36 Modifying an Instance

- EC2 allows you to change the instance type to better match the computing needs
- Instance type modifications is done when the instance is stopped
- This operation can be performed using the AWS Management Console, SDK API, or scripted via AWS CLI
- It is also possible to change an instance's security group
 - ◇ This can be done while the instance is running
 - ◇ You may want to do this to tighten or loosen security rules



2.37 The EC2 Dashboard

- Through the EC2 Dashboard you get administrative access to the following resources:
 - ◇ Instances
 - ◇ Elastic IPs
 - ◇ Volumes
 - ◇ Snapshots
 - ◇ Key Pairs
 - ◇ Load Balancers
 - ◇ Placement Groups
 - ◇ Security Groups



2.38 EC2 Pricing

- Upon sign-up, new AWS users qualify for the AWS's Free Usage Tier discounts: about 750 free hours of *t2.micro* EC2 instance per month for 1 year
- Regular prices for on-demand instances vary by
 - ◇ Instance type
 - ◇ Region
 - ◇ OS and Server Type, e.g.:
 - Linux, RHEL, SUSE Linux Enterprise Server (SLES), Windows, Windows with SQL Standard, Windows with SQL Web
- Instance up-time is measured from when an instance is launched until it is terminated or stopped
 - ◇ Partial instance-hour consumed (e.g. instance running for 1 minute) will be billed as a full hour



2.39 Cluster Networking

- The C3, I2, CR1, G2, and HS1 family instances support cluster networking
- Clusters of these instances are transparently deployed into an environment that provides high-bandwidth, low-latency networking between all instances in the cluster
- Cluster networking is suitable for such use cases as high performance analytics and similar resource intensive applications



2.40 Dedicated Instances

- A dedicated EC2 instance is provisioned to a single user and runs on dedicated hardware (isolated from other tenants)
- Designed for use cases where corporate policies or regulatory requirements demand physical isolation of your EC2 instances on a single-tenant base
- Dedicated instances also offer better run-time performance



2.41 VM Import / Export to/from AWS

- You can import your existing virtual machines into Amazon EC2 by using the EC2 API (CLI) tools
- You can import the following VM types:
 - ◇ VMware ESX and VMware Workstation VMDK images
 - ◇ Citrix Xen VHD images
 - ◇ Microsoft Hyper-V VHD images
 - When you import your Microsoft Windows VM images, AWS will provide the appropriate Microsoft Windows Server license key for your VM
- You can also export previously imported EC2 instances



2.42 Elastic IP Address

- An EC2 instance can be manually or programmatically assigned a static IP address (a.k.a. Elastic IP address)
- Currently, your account is limited to 5 Elastic IP addresses per region
 - ◇ You can apply for more Elastic IP address, if needed
- Any additional IP addresses are allocated per region



2.43 EC2 Service Level Agreement

- In its SLA, AWS makes a Service Commitment to this effect:
- *"AWS will use commercially reasonable efforts to make Amazon EC2 and Amazon EBS each available with a Monthly Up-time Percentage ... of at least 99.95%, in each case during any monthly billing cycle (the "Service Commitment"). In the event Amazon EC2 or Amazon EBS does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below."*
- *"Service Credits are calculated as a percentage of the total charges paid by you (excluding one-time payments such as upfront payments made for Reserved Instances) for either Amazon EC2 or Amazon EBS"*



2.44 Summary

- In this module we reviewed the core network and compute capabilities offered by the AWS cloud platform:
 - ◇ Virtual Private Cloud (VPC), and
 - ◇ EC2

Chapter 3 - AWS Identity and Access Management

Objectives

Key objectives of this chapter

- Overview of the AWS Identity and Access Management Service



3.1 AWS Identity and Access Management (IAM)

- IAM is the AWS user management, authentication and authorization service
- It manages users and their permissions within your AWS account through **users**, **groups**, and **roles** by applying security **policies**
- IAM uses AWS Key Management Service (KMS) to create and control the user encryption keys
- IAM is natively integrated into all the AWS Services
- AWS offers IAM for no charge



3.2 IAM Groups

- Groups help organize users into units managed together
- Group operations include:
 - ◇ Delete / Rename group
 - ◇ Add / Remove users to / from a group
- Group permissions are defined in IAM policies
- There is no default groups
- Groups cannot be nested
- A single IAM user can belong to multiple groups



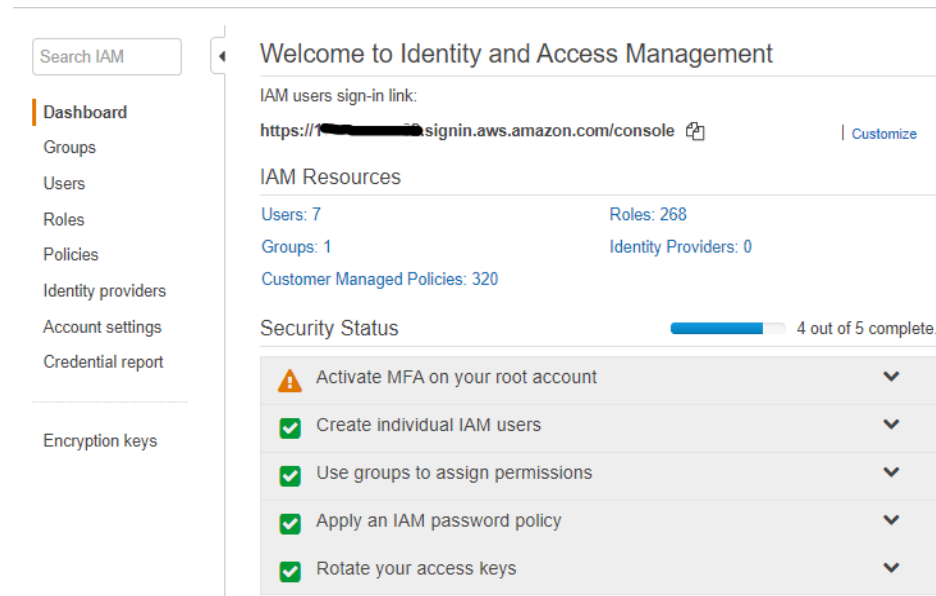
3.3 Working with IAM

- You interface with IAM using:
 - ◇ AWS Management Console
 - ◇ AWS CLI (the AWS tool)
 - ◇ AWS SDKs



3.4 The IAM Dashboard

- Access to the IAM Dashboard is available from the AWS Management Console
- The IAM Dashboard provides a single point of control of all the aspects of identity and access management





3.5 IAM Principals

- A principal is an IAM entity that is allowed to interact with AWS resources
- There are three types of AWS principals:
 - ◇ The root user – the one that was used to create and first access your account with AWS
 - **Important Note:** Use root account to immediately create a delegate IAM user for all AWS housekeeping tasks and keep the root's credentials away, then it is recommended to delete the delegate IAM user's access keys and start using MFA
 - ◇ IAM user
 - ◇ Roles/temp security tokens



3.6 Root Account Access vs. IAM User Access

- AWS root account is created when you sign up with AWS
- You cannot control this account privileges which have full access to AWS resources
- IAM users are created by the root account in line with the least privilege principle
 - ◇ All permissions are implicitly denied by default unless explicitly allowed
 - ◇ There is no default permissions
- The root controls IAM users permissions by revoking or modifying them; an IAM user can also be removed altogether




3.7 Roles

- IAM roles are a secure way to grant permissions to entities that you trust to access AWS resources
- With actor (users) in designated roles, you can enforce the Principle of Least Privilege
- Examples of entities include the following:
 - ◇ IAM user in another account
 - ◇ An Application running on an EC2 instance that needs access to certain AWS resources
 - ◇ Users from a corporate directory who use identity federation with SAML
 - ◇ Users authenticated through OpenID Connect (OIDC) that enables federation with any web / social identity provider that supports OIDC, e.g. Facebook and Google
 - The main point here is that you can securely grant users authenticated outside of AWS IAM permissions to access AWS resources



3.8 Creating a Role in AWS Management Console

 Services ▾ Resource Groups ▾ ☆

🔔 SX @ 1133-6336-5482 ▾ Global ▾

Create role 1 2 3 4

Select type of trusted entity

 **AWS service**
EC2, Lambda and others

 **Another AWS account**
Belonging to you or 3rd party

 **Web identity**
Cognito or any OpenID provider

 **SAML 2.0 federation**
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.



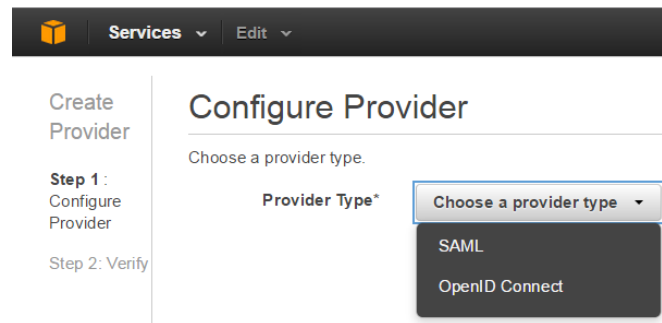
3.9 Accessing AWS

- In order to access AWS resources, you need to be authenticated by AWS
- Each IAM user has three pieces of credentials:
 - ◇ **Password** - used for AWS Management Console login (used along with the IAM user's ID)
 - ◇ **Access key ID & Secret access key** – used for remote web service-based login and the AWS tool
- An actor(user) in a defined IAM role receives Access key ID & Secret access key, plus an IAM Session Token with a configurable expiration time



3.10 Identity Providers

- IAM supports integration with two types of Identity Providers through the Single Sign-On mechanism:
 - ◇ Security Assertion Markup Language (SAML) -based providers (widely used in Enterprise space)
 - Version 2.0 is supported
 - ◇ OpenID Connect-based (web / social network SSO protocol)
- Both options enable you to use your organization's existing identity provider without the need to provide AWS credentials when signing in to AWS





3.11 Need Identity Management for Mobile Apps?

- To enable your clients to sign-up and sign-in to your mobile and web app, use Amazon Cognito
- With Cognito, you have the options to authenticate users through SAML identity solutions, social identity providers such as Facebook and Twitter
- With Cognito you can synchronize data across your devices



3.12 AWS Key Management Service (KMS)

- KMS is the focal point for managing user encryption keys, which include such operations as
 - ◇ Create, enable, disable and audit the use of keys
- You need to define an IAM user and role within an account to administer keys
- Keys are created within an AWS Region
 - ◇ The key has the region id prefixed to it (this concept is referred to as the ARN (Amazon Resource Name))
 - e.g.: **arn:aws:kms:us-east-1:xxxxxxxxxx**
- EBS, S3 and other services use KMS for data encryption services
- KMS Software Development Kit (SDK) enables you to incorporate encryption in your applications as well



3.13 User Management

- IAM performs the following standard user management operations:
 - ◇ Manage user password
 - IAM can assign an auto-generated password
 - The password can be setup to expire at next sign-in
 - ◇ Manages access keys
 - Access keys enable you to make secure REST, command line interface (CLI), the AWS SDKs calls to any AWS service API
 - ◇ Manage signing certificates
 - ◇ Delete user
 - ◇ Add user to groups



3.14 Password Policies

- IAM supports user password policies (rules) to enhance password strength
- The following password rules are available (which can be used in combination):
 - ◇ Require at least one uppercase letter
 - ◇ Require at least one lowercase letter
 - ◇ Require at least one number
 - ◇ Require at least one non-alphanumeric character
 - ◇ Allow users to change their own password
 - ◇ Enable password expiration
 - Password expiration period (in days):
 - ◇ Prevent password reuse
 - Number of passwords to remember
 - ◇ Password expiration requires administrator reset



3.15 Using Multi-Factor Authentication Devices

- Multi-Factor Authentication (MFA) greatly enhances security by requiring additional piece of credentials in the form of a unique authentication code generated in an authentication device when accessing the AWS Cloud
- When MFA is enabled, users are required to provide the (usual) username and password (treated as the first factor - something that the user knows), plus an authentication code from their AWS (the second factor - something that the user has)
- IAM provides steps to help you setup and assign an MFA device to the IAM user
- Users should not re-use MFA devices (devices should not be shared)
- MFA authentication is recommended for login accounts to AWS to provide an additional layer of security



3.16 Hardware-based and Virtual MFA Devices

- The device can be hardware-based or virtual
 - ◇ A hardware device can be a keyfob or a display card (in a convenient shape, like a credit card)
 - Hardware devices are "tamper-evident"
 - ◇ A virtual device is any device on which you can install a "time-based one-time password" application; basically, it can be any smart phone (Android, Blackberry, iPhone, and Windows Phone)
 - Security of virtual devices is rated lower than that provided by physical devices



3.17 Summary

- IAM is the user management, authentication and authorization service
- IAM uses the AWS Key Management Service (KMS) for managing user encryption keys
- You can enhance security of your AWS cloud-based solutions by applying password policies and using the multi-factor authentication option supported by IAM

Chapter 4 - AWS Simple Storage Service

Objectives

Key objectives of this chapter

- Describe the Amazon Simple Storage Service



4.1 What is AWS Simple Storage Service (S3)

- Amazon Simple Storage Service (S3) is a secure, durable, highly-scalable object (file) storage
- S3 was Amazon's first publicly available web service launched in the United States in March 2006 and in Europe in November 2007
- You only pay for the actual storage you use
- S3 can be used as a stand-alone service or integrated with other AWS services
- For redundancy, client's data is stored across a number of locations and multiple devices in each location
- Amazon has not published any details of the S3's design



4.2 AWS Storage

- S3 handles arbitrary objects from 1 byte up to 5 TB
- Every object can have up to 2 kilobytes of metadata
- Objects are stored in buckets (containers) that may be used to build a folder-like hierarchical storage structure
- Amazon Machine Images (AMI's) can be exported to S3 as bundles
- S3 exposes its functionality through the command-line and web services interfaces (REST, SOAP, and BitTorrent)
 - ◇ **Note:** SOAP support over HTTP is deprecated and new S3 features will not be supported for SOAP



4.3 Regions

- You can choose the geographical region where Amazon S3 will store the buckets you create
- Choosing the right region can help optimize object upload/download latency, minimize transfer costs, or address specific regulatory requirements
- According to AWS documentation: “[S3] *Objects stored in a region never leave the region unless you explicitly transfer them to another region. The objects stored in an Amazon S3 region physically remain in that region. Amazon S3 does not keep copies or move it to any other region. However, you can access the objects from anywhere, as long as you have necessary permissions.*”
 - ◇ So, S3 stores its data in the region where the bucket is created. The data is distributed across multiple availability zones in that region



4.4 S3 Regions

- The following S3 regions are supported
 - ◇ US: Standard, West (Oregon), West (N. California)
 - ◇ EU: Ireland, Frankfurt
 - ◇ Asia Pacific: Singapore, Tokyo, Sydney
 - ◇ South America: Sao Paulo



4.5 Getting started with S3

- First, you need to create a bucket
 - ◇ When you create a bucket, you have an option choose a Region
 - ◇ You can also enable logging for your bucket, which would allow you to get detailed access logs on bucket-related activities
- After the bucket is created, you can start uploading objects to the bucket
 - ◇ While objects are in the bucket, they follow pre-defined rules that you establish when you create the bucket; for example, S3 offers you a way to manage objects' life-cycle
 - ◇ You can directly change some object parameters (e.g. enable object encryption)



4.6 Using BitTorrent

- S3 supports the BitTorrent Internet distribution and file sharing protocol
- Using BitTorrent can reduce your bandwidth costs
- You activate BitTorrent by adding the *?torrent* parameter at the end of your GET request in the REST API
- The Requester Pay arrangement (covered later in this module), if enabled, does not provide support for the BitTorrent downloads



4.7 More on Buckets

- Buckets help organize the user storage namespace at the top level
- They are also used for calculating S3 storage and data transfer charges
- S3 objects are publicly addressable from anywhere on the Internet
- Bucket names are shared across S3, so they must be unique inside the bucket; you can create sub-folders where you can store objects, e.g. `/webagedev/L1/L2`



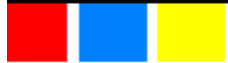
4.8 Bucket Configurable Properties

- **Versioning**
 - ◇ Keep multiple versions of an object in the same bucket.
- **Server access logging**
 - ◇ Set up access log records that provide details about access requests.
- **Static website hosting**
 - ◇ Host a static website, which does not require server-side technologies.
- **Object-level logging**
 - ◇ Record object-level API activity using the CloudTrail data events feature (additional cost).
- **Default encryption**
 - ◇ Automatically encrypt objects when stored in Amazon S3, if enabled
 - ◇ Options:
 - **None**
 - **256** (Server-Side Encryption with Amazon S3-Managed Keys)
 - **AWS-KMS** (Server-Side Encryption with AWS KMS-Managed Keys)



4.9 Advanced S3 Bucket Properties

- **Object lock**
 - ◇ Prevent objects from being deleted.
- **Tags**
 - ◇ Use tags to track your cost against projects or other criteria.
- **Transfer acceleration**
 - ◇ Enable fast, easy and secure transfers of files to and from your bucket.
- **Events**
 - ◇ Receive notifications when specific events occur in your bucket.
- **Requester pays**
 - ◇ The requester (instead of the bucket owner) will pay for requests and data transfer.



4.10 The Bucket Creation Dialog in the Management Console

The screenshot shows the 'Create bucket' dialog in the AWS Management Console, specifically the 'Configure options' step (step 2 of 4). The dialog is titled 'Create bucket' and has a progress bar at the top with four steps: 1. Name and region (completed), 2. Configure options (current step), 3. Set permissions, and 4. Review. The main content area is titled 'Properties' and contains several sections with checkboxes and links:

- Versioning**: ☐ Keep all versions of an object in the same bucket. [Learn more](#)
- Server access logging**: ☐ Log requests for access to your bucket. [Learn more](#)
- Tags**: You can use tags to track project costs. [Learn more](#)
Below this, there are two input fields labeled 'Key' and 'Value', and a button labeled '+ Add another'.
- Object-level logging**: ☐ Record object-level API activity using AWS CloudTrail for an additional cost. See [CloudTrail pricing](#) or [learn more](#)
- Default encryption**: ☐ Automatically encrypt objects when they are stored in S3. [Learn more](#)
- Advanced settings**: A dropdown arrow icon.
- Object lock**: ☐ Permanently allow objects in this bucket to be locked. [Learn more](#)
Below this, there is a note: 'Object lock requires bucket versioning to be enabled.'



4.11 Bucket Permissions

- Bucket objects allow you to set permissions for listing, upload/delete, and other operations per active user (called *Grantee*)
- You can also grant permissions to multiple accounts with added conditions (see slide's notes for a bucket policy sample) using bucket policies
- S3 offers you a wizard-like AWS policy generator tool for creating access control policies for AWS products and resources [<http://awspolicygen.s3.amazonaws.com/policygen.html>]
- In addition to the bucket-level permissions, you can set up object-level permissions by configuring an access control list (ACL)



4.12 Bucket-level Operations

- Get total bucket size
- Change storage class (available options are shown in the slide's notes)
- Change encryption
- Change metadata
- Make public
- Delete
- Copy



4.13 Authorization of REST Requests

- S3 performs authorization of REST requests based on the access control list associated with each bucket and object
- The bucket owner can create an authenticated access end-point URL with time-bounded access window
- With the time-bounded access window established, the URL will be only accessible for a period of time, such as the next 60 minutes, or the next 24 hours



4.14 Adding Cross-Origin Resource Sharing Configuration

- You can add Cross-Origin Resource Sharing (CORS) permissions for your buckets
- CORS permissions allow you to selectively grant web applications running on other domains access to the content in your Amazon S3 bucket
- Take the following example
 - ◇ Create a website referenced at **<http://website.s3-website-us-east-1.amazonaws.com>**
 - ◇ You store JavaScript for your website in your new S3 bucket **website.s3.amazonaws.com**
 - ◇ A browser would normally block JavaScript from allowing GET and PUT requests to your bucket, but with CORS, you can configure your bucket to explicitly enable cross-origin requests
- CORS configuration for our example:

```
<CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>website.s3-website-us-east-1.amazonaws.com </AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
    <MaxAgeSeconds>3000</MaxAgeSeconds>
    <AllowedHeader>Authorization</AllowedHeader>
  </CORSRule>
</CORSConfiguration>
```



4.15 Event Notifications

- S3 offers the event notification mechanism that is set up at the bucket level
- You can subscribe and get notifications on the following events:
 - ◇ Reduced Redundancy Storage Object loss
 - ◇ Object created
 - ◇ Object copied
 - ◇ Multi-part upload complete
- You can choose the following destinations for notifications:
 - ◇ SNS
 - ◇ SQS Queue
 - ◇ AWS Lambda function (user-defined code)



4.16 The Requester Pays Option

- At the bucket level, you can enable the Requester Pays (RP) for object download from that bucket
- Under this object transfer arrangement, the requester pays for the requests targeting this bucket and the cost of data transfer
- After you set up RP, requesters must include the **x-amz-request-payer** stanza as an HTTP header for POST and GET requests, or as a parameter in a REST request
- **Note:** The following RP requests are not supported:
 - ◇ Anonymous requests
 - ◇ BitTorrent
 - ◇ SOAP requests
- The bucket owner still pays for the object storage



4.17 The Object Key

- A key is a unique identifier for an object within a bucket
- The object key is made up of the folder name(s), if any, and the object's file name, e.g. *L1/L2/your_object_name*
- The combination of a bucket, key, and version (if enabled), uniquely identifies each user object in S3, which makes your objects publicly accessible via a unique URL, e.g.

`https://s3.amazonaws.com/webagedev/L1/L2/Mobius-strip-Flickr.jpg`



4.18 Object Versioning

- The Object versioning feature for Amazon S3 was introduced in 2010 as an additional level of data protection
- Versioning gets enabled at the bucket level
- By default your bucket is not versioned
- By default, GET REST requests will retrieve the object's most recent version written version
 - ◇ Previous object versions can be retrieved by specifying a version in the GET request
- Versioning also allows users recover deleted object
- Once enabled, object versioning cannot be disabled
 - ◇ You are allowed to suspend object versioning



4.19 Example of Object Properties

hospital-1.dat

DownloadCopy pathSelect from

Latest version ▾

Overview

Key	hospital-1.dat
Size	52.7 KB
Expiration date	N/A
Expiration rule	N/A
ETag	12591f15cb6f4250f4e9878441b6b524
Last modified	Jan 14, 2019 2:05:12 PM GMT-0500
Object URL	https://s3.amazonaws.com/xfiles2019/hospital-1.dat

Properties

Storage class	Standard
Encryption	None
Metadata	1
Tags	0 Tags
Object lock	Disabled

Permissions

Owner	
Object	
Read	1 Grantees
Write	1 Grantees
Object permissions	
Read	1 Grantees
Write	1 Grantees



4.20 Object Storage Class Levels

- S3 ensures durability of your data by replicating copies of the data to different Availability Zones, which can be controlled by you using the Storage Class Levels

Storage class	Designed for	Availability Zones	Min storage duration	Min billable object size	Monitoring and automation fees	Retrieval fees
<input checked="" type="radio"/> Standard	Frequently accessed data	≥ 3	-	-	-	-
<input type="radio"/> Intelligent-Tiering	Long-lived data with changing or unknown access patterns	≥ 3	30 days	-	Per-object fees apply	-
<input type="radio"/> Standard-IA	Long-lived, infrequently accessed data	≥ 3	30 days	128KB	-	Per-GB fees apply
<input type="radio"/> One Zone-IA	Long-lived, infrequently accessed, non-critical data	≥ 1	30 days	128KB	-	Per-GB fees apply
<input type="radio"/> Glacier	Data archiving with retrieval times ranging from minutes to hours	≥ 3	90 days	-	-	Per-GB fees apply
<input type="radio"/> Reduced Redundancy (Not recommended)	Frequently accessed, non-critical data	≥ 3	-	-	-	-

[Cancel](#)[Save](#)



4.21 Object Life-cycle Configuration

- You can configure object life-cycle events, such as archiving, or deleting objects (e.g. log files) after a specified time period
- Rules controlling life-cycle events are configured at the bucket level and apply to the whole bucket
- You can define the following actions on objects:
 - ◇ **Archive Only**
 - And keep the object in S3
 - ◇ **Permanently Delete Only**
 - ◇ **Archive and then Permanently Delete**
 - Essentially, move the object from S3 to the archive
- Here is a visual representation of a rule for archiving a file in the Glacier archival storage after 1 day of holding it in S3:



- You can define up to 1000 such rules



4.22 Amazon S3 Data Consistency Model

- High availability of objects in S3 is achieved by replicating data across multiple servers and multiple locations
- If a PUT request (used to upload objects) completes successfully, your data may be persisted only in some parts of S3 with some additional time required to replicate (propagate) the updates across the rest of the system
- A single key updates (e.g. via using the REST PUT verb) are atomic (they are not partial)
 - ◇ In some circumstances, a subsequent read on that key might still return the old data
 - ◇ Atomic updates across keys are not guaranteed
- In most regions, S3 provides read-after-write consistency for PUTS of new objects and eventual consistency for the overwrite PUTS
- Currently, Amazon S3 does not support object locking and you will need to implement the required object-locking mechanism at the application level
 - ◇ With two or more concurrent PUT updates against the same key, the request which updates last wins



4.23 Eventually Consistent Reads vs Consistent Reads

Eventually Consistent Read	Consistent Read
Stale reads possible	No stale reads
Lowest read latency	Potential higher read latency
Highest read throughput	Potential lower read throughput



4.24 Amazon S3 Security

- Amazon S3 provides four different access control mechanisms:
 - ◇ Identity and Access Management (IAM) policies
 - ◇ Access Control Lists (ACLs)
 - ◇ Bucket policies
 - ◇ Query string authentication
- Storage data can be optionally encrypted with AES 256; the key is handled transparently by S3
- You can use S3 permissions to grant or deny access for data upload or download operations



4.25 S3 Use Case: Backup and Archiving

- In addition to your data, you can store snapshots of EBS volumes, RDS databases, and other AWS resources related to your projects
- The Amazon S3's versioning capability can give you further protection against data loss for your stored data
- Using object life-cycle rules (that you define at the bucket level), you can set up basic workflows for deleting and archiving data sets
- Where the object retrieval time is not the primary requirement, the Amazon Glacier service can be your cost-efficient cloud storage solution



4.26 Another S3 Use Case: Static Web Hosting

- You can host all the static content of your website on Amazon S3
 - ◇ The static content includes HTML pages, JavaScript and CSS files, images, etc.
 - ◇ You can refresh static content by uploading the updated versions of those static content assets
- You configure website hosting at the bucket level
- Once you enable your bucket for static website hosting, all your content in the bucket is publicly accessible via the Amazon S3 HTTP endpoint at the following sub-domain:

`<your-bucket-name>.<your-region>.amazonaws.com`



4.27 S3 Use Case: Disaster Recovery

- Disaster Recovery (DR) is a set of activities undertaken by a company to prepare for and recover from any possible service interruption (disaster) that may seriously impact their business
 - ◇ Many companies establish business continuity processes that provide for any hardware or software failure, a network or power outage, physical damage to facilities in the form of fire or flooding, etc.
- The AWS Cloud platform is designed with DR scenarios in mind that enable rapid service restoration
- You can use the AWS Cloud infrastructure for hosting your “pilot light” (scaled down and a bare minimum service setup) environments or full “hot standby” environments that enable rapid system fail-over in case of a massive service disruption



4.28 AWS S3 Pricing

- In S3, you pay for the storage you actually use
- Prices are based on the region in which you created your S3 bucket
- You will be charged for the following operations:
 - ◇ Storage (by type: Standard, Reduced Redundancy, and Glacier)
 - ◇ Requests (by type: GET, PUT, etc.)
 - ◇ Data Transfer (by direction and by rate tiers)
- Newly signed-up customers get 5 GB of Amazon S3 standard storage, 20,000 Get Requests, 2,000 Put Requests, and 15GB of data transfer out each month for one year



4.29 Amazon S3 Transfer Acceleration

- To speed up data transfer to and from a bucket, enable Transfer Acceleration on the bucket
- You can achieve up to 3X acceleration
- You only pay if enabling Transfer Acceleration results in a performance improvement
- Transfer Acceleration leverages Amazon CloudFront's globally distributed edge locations and as the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path
- Transfer Acceleration will help when
 - ◇ You upload to a bucket from across the globe (large network latencies)
 - ◇ You transfer terabytes of data (data size)



4.30 Amazon S3 SLA Definitions

- S3 defines the “Error Rate” with the following formula:
 - ◇ The total number of internal server errors with a status “InternalError” or “ServiceUnavailable” returned by Amazon S3
divided by
 - ◇ The total number of requests during that five minute period
- “Monthly Up-time Percentage” is calculated by
 - ◇ Subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle



4.31 Amazon S3 SLA Service Commitment

- *"AWS will use commercially reasonable efforts to make Amazon S3 available with a Monthly Up-time Percentage... of at least 99.9% during any monthly billing cycle... In the event Amazon S3 does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below"*

Monthly Uptime Percentage	Service Credit Percentage
Equal to or greater than 99% but less than 99.9%	10%
Less than 99%	25%

- A "Service Credit" is a dollar credit that AWS may credit back to an eligible Amazon S3 account



4.32 S3 Service Limits - Buckets

- Unique bucket name across **all of AWS**
- Name of bucket must be between 3-63 character long, must not contain uppercase characters or underscores, must start with a lowercase letter or number
- 100 Buckets per account (soft)



4.33 S3 Service Limits

- Buckets
 - ◇ Unique bucket name across **all of AWS**
 - ◇ Name of bucket must be between 3-63 character long, must not contain uppercase characters or underscores, must start with a lowercase letter or number
 - ◇ 100 Buckets per account, but can be increased
- Items
 - ◇ Maximum size of one item is 5TB
 - ◇ Maximum upload size of an item is 5GB
 - ◇ Maximum number of parts in a multi part upload is 10.000
- Performance
 - ◇ 3,500+ PUT/POST/DELETE requests per second per prefix in a bucket
 - ◇ 5,500+ GET requests per second per prefix in a bucket
 - ◇ Unlimited performance with the right prefix distribution



4.34 Summary

- In this chapter we reviewed the capabilities of the AWS Simple Storage Service

Chapter 5 - Caching and Content Delivery Network Services

Objectives

Key objectives of this chapter

- Overview of AWS Caching and Content Delivery Network



5.1 CloudFront Content Delivery Network

- A global accelerated content delivery network (CDN) for static and/or streaming content, e.g. S3 objects
- CloudFront organizes your content into distributions based on the location(s) of the original version of your files
 - ◇ Each distribution is assigned a unique domain name (e.g. *QXYZ.cloudfront.net*) that is referenced through the global network of AWS edge locations
 - ◇ When your clients request an object using this domain name, they are automatically routed to the nearest geographical locations (called edges)
- S3 objects (images, documents, etc.) can be replicated and cached using CloudFront in a number of different geographical locations



5.2 CloudFront Features

- Globally distributed and massively scaled
 - ◇ The CloudFront network has 160+ points of presence (edges) across the globe that take full advantage of highly-resilient Amazon backbone network
- Highly programmable
 - ◇ You can use Lambda@Edge functions to respond to CloudFront events
- Secure at points of presence (edges)
 - ◇ “Always on” DDoS monitoring and mitigation
 - ◇ TLS connection negotiation with the highest security ciphers
 - TLS connections with clients terminate at a nearby edge location
 - ◇ Viewer authentication with signed URLs



5.3 CloudFront Use Cases

- Static content caching to speed up content delivery to your clients
- Support for global companies that require fast download speeds of large files (e.g. software updates)
- Fast and secure data upload
- Collecting asset access statistics (via access log files) for subsequent operational reporting
- Live and on-demand video streaming
 - ◇ 4K bandwidth delivery



5.4 Accessing CloudFront

- You can work with CloudFront using any of the standard AWS tools:
 - ◇ AWS Management Console
 - ◇ CloudFormation
 - ◇ CLI, and
 - ◇ SDKs



5.5 Amazon ElastiCache

- ElastiCache is a scalable (fully managed) in-memory cache in the AWS Cloud
 - ◇ Supports up to 3.55 TiB of in-memory data
- Provides sub-millisecond latency for object retrieval
- As of 2017, ElastiCache is compatible with Redis and Memcached



5.6 The Redis and Memcached Engines

■ Redis

- ◇ In-memory data structure store used as database, cache and message broker. ElastiCache for Redis offers Multi-AZ with Auto-Failover and enhanced robustness
- ◇ Encryption at-rest is supported
- ◇ Automatic back-up with a specified retention period
- ◇ A choice of the cache server type and number of cache replicas
 - You pay for what you use!

■ Memcached

- ◇ High-performance, distributed memory object caching system, intended for use in speeding up dynamic web applications
- ◇ You can specify the cache server type and count
 - Again, you pay for what you use!



5.7 Summary

- In this module we reviewed the following AWS services:
 - ◇ CloudFront, and
 - ◇ ElastiCache

Chapter 6 - AWS Databases

Objectives

Key objectives of this chapter

- AWS database services



6.1 AWS Database Services

- Relational Database Service (RDS)
 - ◇ Supporting OLAP & OLTP use cases
- NoSQL
- Data Warehouse
 - ◇ OLAP



6.2 The CAP Theorem

- The CAP theorem was formulated by Eric Brewer
 - ◇ <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>
- It states that any distributed computer system can have at most two of three desirable properties:
 - ◇ **C** - consistency is equivalent to having a single up-to-date copy of the data; all client always have the same view of the data
 - ◇ **A** - high availability of the data; all clients can always read and write
 - ◇ **P** - tolerance to network partitions (in distributed system deployments)



6.3 Mechanisms to Guarantee a Single CAP Property

- Some of the mechanisms to ensure a single CAP property are:
 - ◇ Consistency - pessimistic locking
 - ◇ Availability - optimistic locking
 - ◇ Partition - two-phase commit
- Big Data systems almost always partition data, leaving designers with a choice between data consistency and availability



6.4 NoSQL Systems CAP Triangle



Source: <http://blog.nahurst.com/visual-guide-to-nosql-systems>



6.5 Relational vs NoSQL Databases

Relational (SQL)	NoSQL (non-relational)
Normalized	Denormalized
Vertical scalability (horizontal scalability as an added value)	Two-way scalability
Standards-based Query Language (SQL)	Mostly NoSQL engine-specific language, occasionally with some elements of SQL
Transaction are supported (with full support for ACID properties)	Usually not supported
Strict schema enforced via DDL	Schema-less (this feature is also known as <i>flexible schema</i> , or <i>schema on demand</i>)
Available and Consistent (the AC database type)	Mostly Partitioned and Available (the PA, a.k.a. Eventually Consistent storage type)









6.6 AWS Databases Overview

- Amazon Relational Database Service (RDS) [<https://aws.amazon.com/rds/>]
 - ◇ Covers the needs for both OLAP & OLTP
 - ◇ Provides for hassle-free setup, deployment, scaling, and automated administration tasks like backups, replication, and patching of these SQL databases (multiple versions are supported):
 - PostgreSQL,
 - Oracle,
 - Microsoft SQL Server (various editions, ranging from Express to Enterprise),
 - MariaDB (enterprise-grade open source database built by the MySQL team),
 - MySQL Community Edition,
 - Amazon Aurora
 - ◇ Suitable for complex transactions or SQL queries with up to 30K IOPS (15K reads/second + 15K writes/second) on a single node/shard
- DynamoDB (AWS managed NoSQL service)



6.7 AWS RDS Engine Options Page in the Management Console

<input checked="" type="radio"/> Amazon Aurora 	<input type="radio"/> MySQL 	<input type="radio"/> MariaDB 
<input type="radio"/> PostgreSQL 	<input type="radio"/> Oracle 	<input type="radio"/> Microsoft SQL Server 



6.8 Database Instances

- You manage one or more database instances (your choice of commercial or open source database) using the API provided by RDS
- Your database instances run in isolated environment within private network segments
- You have a choice of database instance classes that range from 1 vCPU with 1GB of memory up to 32 vCPUs with 244 GB
 - ◇ You can change the class as your needs / budget change

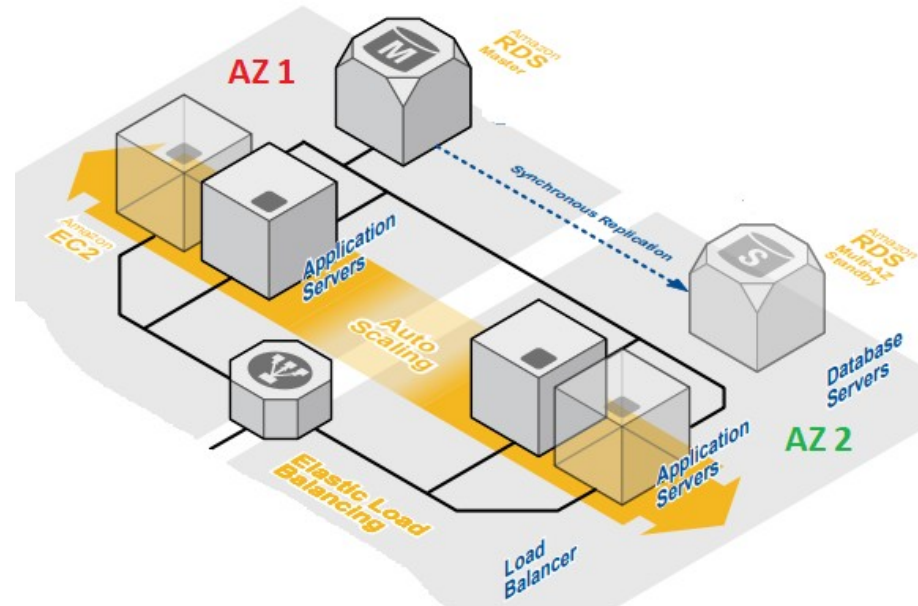


6.9 Multi-AZ Deployment

- Amazon RDS supports Multi-AZ deployments for highly available and fault-tolerant database solutions
- In a Multi-AZ architecture, you deploy your database in two AZ's:
 - ◇ You place the Master database in one AZ, and
 - ◇ A secondary database instance in another
 - ◇ AWS automatically replicates the data from the Master to the secondary node
- AWS generates a unique DNS name that is resolved to a specific IP address. The database clients use that DNS name when connecting to the database.



6.10 Two-AZ Master-Slave RDS Solution Architecture



- ◇ Adapted from the AWS Application Architecture Center's blueprint https://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf



6.11 Licensing

- Commercial databases, e.g. Oracle and Microsoft SQL Server, require appropriate licenses to be shared with Amazon
- Two licensing models are offered to AWS clients:
 - ◇ License included
 - Offered by RDS and included in the database instance price
 - ◇ Bring Your Own License (BYOL)



6.12 Migrating Your On-Premise Databases to RDS

- There are two ways for on-prem database migration:
 - ◇ AWS Database Migration Service
 - ◇ The target database's native import tools
 - You do a backup on premise, upload the dump to AWS, and then import the file into the RDS version of that Database



6.13 Aurora

- Amazon Aurora is a MySQL and PostgreSQL-compatible relational database offered as part of RDS
- Aurora is distributed, fault-tolerant, and self-healing storage system with replication across three Availability Zones (AZs)
- Continuous backup to Amazon S3
- Offers the performance and availability of traditional enterprise relational databases with cost-effectiveness of open source databases
- Delivers 5x the MySQL performance and 3x that of PostgreSQL
- Positioned by Amazon as an enterprise-grade cloud database platform for your existing MySQL and PostgreSQL databases



6.14 Aurora DB Instance Details In AWS Console

DB engine

Aurora - compatible with MySQL 5.6.10a

Capacity type [Info](#)

☐ Provisioned

You provision and manage the server instance sizes.

☒ Provisioned with Aurora parallel query enabled [Info](#)

You provision and manage the server instance sizes, and Aurora improves the performance of analytic queries by pushing processing down to the Aurora storage layer (currently available for Aurora MySQL 5.6)

☐ Serverless [Info](#)

You specify the minimum and maximum of resources for a DB cluster. Aurora scales the capacity based on database load.

DB instance class [Info](#)

db.r4.16xlarge — 64 vCPU, 488 GiB RAM



Multi-AZ deployment [Info](#)

☒ Create Replica in Different Zone

☐ No



6.15 Redshift

- Amazon Redshift is a data warehouse system used for OLAP use cases
- Usually used in combination with RDS for its source of data
- Exabyte-range storage
- Allows you run SQL against unstructured data
 - ◇ You need Redshift Spectrum service for that
- Supports parallelized query execution across a cluster of nodes
- Supports open data formats, like
 - ◇ Avro, CSV, Grok, Ion, JSON, ORC, Parquet, RCFile, RegexSerDe, SequenceFile, TextFile, and more
- Allows for integration with a variety of BI tool using standard JDBC/ODBC connectors



6.16 Amazon RDS Use Cases

- **Web and mobile applications**
 - ◇ RDS offers clients high throughput, scalability, high availability (via Multi-Availability Zone (Multi-AZ) deployment), and massive storage capacity
 - ◇ Users: airbnb
- **E-commerce**
 - ◇ RDS help companies large and small meet Payment Card Industry (PCI) compliance
- **Mobile and online games**
 - ◇ RDS offers familiar database engines that easily scale to meet user demand
 - ◇ Users: Bandai Namco Studios



6.17 RDS Service Limits

- MySQL, MariaDB
 - ◇ Database sizes up to 16TB
- Database name limitations
 - ◇ MySQL, MariaDB, PostgreSQL and Aurora – 1-63 characters
 - ◇ Oracle – up to 8 characters
 - ◇ Microsoft SQL – follows the naming schema of the SQL version
- High Availability limitations
 - ◇ MySQL, MariaDB, PostgreSQL – only one pair of Active-Passive servers in synchronous replication (Multi-AZ)
 - ◇ Aurora – Supports Multi-AZ on the primary instance plus adds support for multiple readers. Readers can be promoted to writers instead of using Multi-AZ.
- Read Replica limits
 - ◇ MySQL, MariaDB, PostgreSQL – up to 5 read replicas
 - ◇ Aurora – up to 15 read replicas + additional 16 cross-region replicas



6.18 Summary

- In this chapter we reviewed the main database options available for AWS clients

Chapter 7 - Monitoring and Managing Applications in AWS

Objectives

Key objectives of this chapter

- AWS Management Tools Overview
- AWS Well Architected Framework
- CloudWatch
- CloudTrail
- Trusted Advisor



7.1 AWS Management Tools Overview

- Provisioning
 - ◇ AWS CloudFormation
 - ◇ AWS Service Catalog
 - ◇ Elastic Beanstalk
- Operations Management
 - ◇ AWS Systems Manager
 - ◇ AWS CloudTrail
 - ◇ AWS Config
- Configuration Management
 - ◇ AWS OpsWorks (Chef, Puppet)
- Monitoring and Logging
 - ◇ AWS CloudWatch

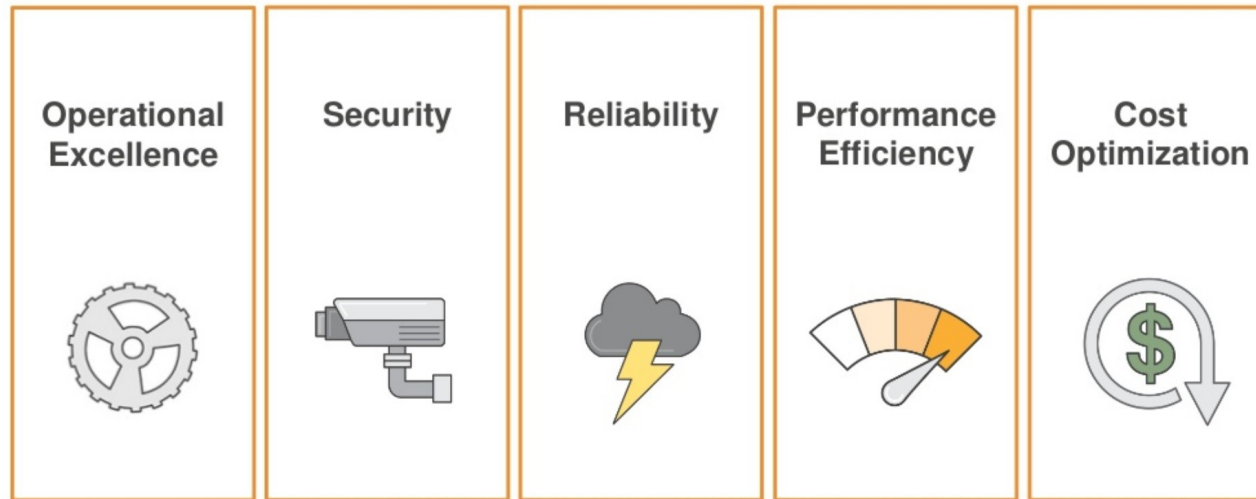


7.2 AWS Well Architected Framework

- Questions developed by AWS experts to help customers analyze their environment
- The framework does not provide, implementation details, architectural patterns or relevant case studies
- The framework enables customers to:
 - ◇ Assess their environments against AWS best practices
 - ◇ Improve their infrastructure practices and deployments
 - ◇ Understand the business impact of architectural decisions



7.3 AWS Well Architected Framework Pillars





7.4 Operational Excellence:

- Align with business objectives
- Give alerts and respond to them in an automated manner
- Perform operations with code
- Make incremental changes
- Learn from failures
- Test for responses to unexpected events
- Document current procedures



7.5 Security

- Protect information, systems and assets
- Do risk assessments
- Have mitigation strategies
- Secure at all layers
- Enable traceability
- Implement a principle of least privilege
- Automate best practices



7.6 Reliability

- Automatically recover from infrastructure or service disruptions
- Scale horizontally to increase availability
- Use multiple availability zones
- Choose instance type based on application needs
- Stop guessing about capacity
- Test recovery procedures



7.7 Performance Efficiency

- Make efficient use of resources
- Enable latency-based routing
- Adopt latest technologies and architectures
- Experiment often



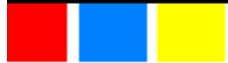
7.8 Cost Optimization

- Avoid unneeded costs
- Assess resource utilization
- Analyze and attribute expenditure
- Match supply and demand
- Optimize over time
- Delete unused resources
- Use consolidated billing, spot instances, and reserved instances
- Rightsize before/after migrations.

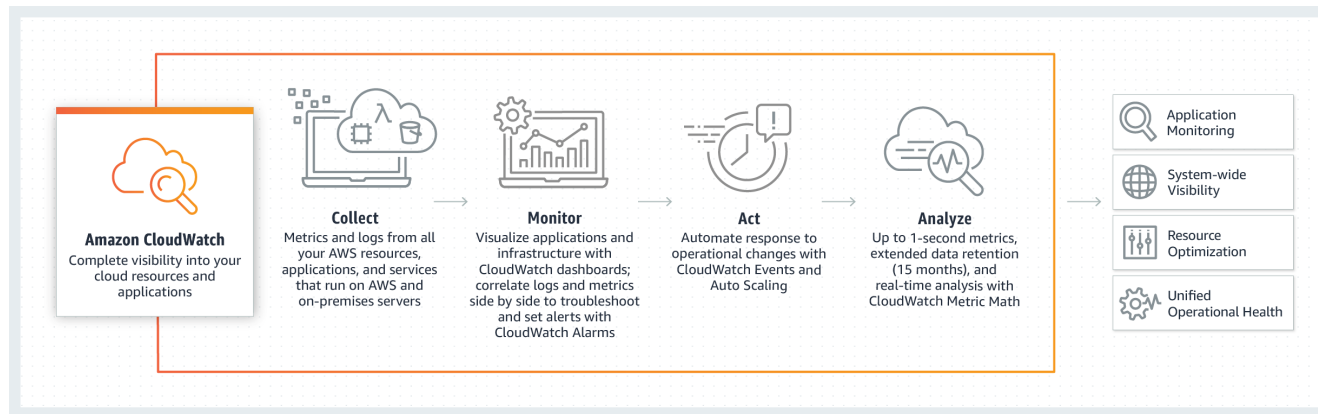


7.9 What is CloudWatch

- Amazon CloudWatch enables users to monitor their AWS resources and applications that run on AWS and/or on-premise servers
- CloudWatch collects monitoring and operational data in the form of logs, metrics, alarms, and events from AWS services, user and third party applications (e.g. Microsoft IIS server, MongoDB, Nginx, etc.)
- This service provides users with a unified view of enlisted AWS resources
- AWS CloudWatch documentation can be found here:
<https://docs.aws.amazon.com/cloudwatch>



7.10 How CloudWatch Works



Source: AWS Documentation



7.11 Use Cases

- Infrastructure monitoring and troubleshooting
 - ◇ Monitor key metrics and logs, visualize your application and infrastructure stack, create alarms, and correlate metrics and logs to understand and resolve root cause of performance issues
- Resource optimization
 - ◇ Use CloudWatch Alarms to automate capacity and resource planning through Auto Scaling
- Application monitoring
 - ◇ Trigger automated CloudWatch Alarms and Lambda workflows to improve customer experience
- Log analytics
 - ◇ Explore, analyze, and visualize your logs instantly to address operational issues and improve applications performance

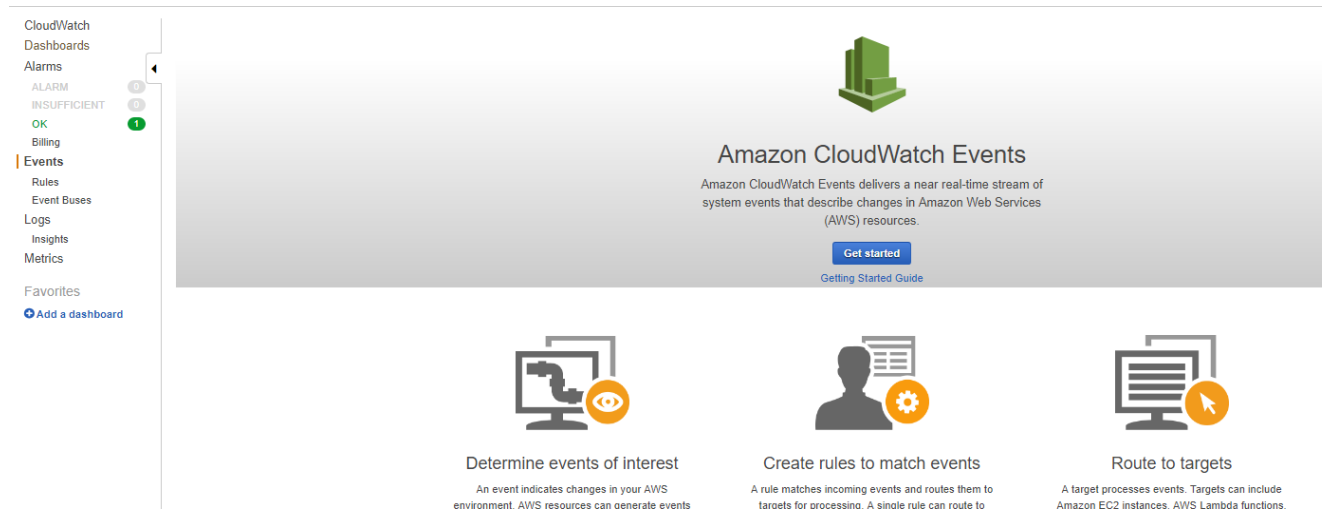


7.12 Log Management

- CloudWatch as a log management solution offers users the following services:
 - ◇ Store log data from multiple sources in a central location
 - ◇ Enforce retention policy on those logs so they are available for a specific period
 - ◇ Offer a searching facility to look inside the logs for important information
 - ◇ Generate alerts based on metrics users define on the logs
 - Alerts can be sent off using the SNS service



7.13 Amazon CloudWatch in the AWS Management Console





7.14 CloudWatch AWS Services Integration

- CloudWatch is natively integrated with 70+ AWS services, including EC2, DynamoDB, S3, ECS, Lambda, CloudTrail, Amazon API Gateway, etc.
- Predefined metrics are generated and published at a minute interval
 - ◇ For a complete list of AWS Services and their CloudWatch metrics, visit <https://amzn.to/2T6xm07>



7.15 Monitoring EC2 Instances

- AWS EC2 users are automatically registered for Amazon CloudWatch and EC2 Basic Monitoring at no additional charge
- Users can upgrade to Detailed Monitoring when creating a new EC2 instance or for existing instances
- Amazon CloudWatch also automatically monitors metrics of
 - ◇ Elastic Load Balancers (EBS)
 - Request count and latency
 - ◇ EBS volumes
 - Read/write latency



7.16 Collecting Metrics and Logs from EC2 Instances and On-Premises Servers with the CloudWatch Agent

- CloudWatch Logs Agent can be installed manually or either through CloudFormation or Chef
- CloudWatch Agent, when installed on your EC2 or on-premise servers, enables you to do the following:
 - ◇ Collect system-level metrics from Amazon EC2 instances beyond standard metrics
 - ◇ Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS
 - ◇ Retrieve custom metrics from your applications or services using the **StatsD** and **collectd** protocols
 - StatsD is supported on both Linux and Windows
 - collectd is supported only on Linux servers
- For more details, visit <https://amzn.to/2xtw722> and <https://bit.ly/2mYhkaY>



7.17 CloudWatch Alarms

- CloudWatch alarms enable users to set up system triggers to respond to specific events
- For example, you can set up Auto Scaling to add or remove EC2 instances in line with the inbound traffic profile
 - ◇ This elastic QoS can help optimize costs and improve resource utilization



7.18 Alarms in the AWS Management Console Examples



Source: AWS Documentation



7.19 Amazon CloudWatch Events

- Through Amazon CloudWatch events you can respond to state changes in your AWS resources or when a particular threshold is exceeded
 - ◇ For example,
 - CloudWatch will provide monitoring of EC2 instances with respect of operational performance, including such metrics as CPU utilization, I/O operations, network traffic (in/out), and response latencies. It can also be configured to monitor HTTP status codes in Apache logs, etc.
 - CloudWatch generates an event when the state of an EC2 instance changes from pending to running or when Auto Scaling launches an instance
- For collecting custom log data, you need to install and configure CloudWatch agents that will be sending your logs to the CloudWatch Logs service and then create your specific metric filter there



7.20 AWS CloudTrail

- The AWS CloudTrail is a web service that enables governance, compliance, operational auditing, and risk auditing of user AWS accounts
- This service continuously monitors user activity and resource usage and provides visibility into related user actions across AWS infrastructure, AWS Management Console, API calls from AWS SDKs and CLI, etc.
- CloudTrail captures actions made directly by the user or on behalf of the user by an AWS service
- When a particular activity occurs in your AWS account, that activity is automatically recorded in event logs making event history available for subsequent security analysis, resource change tracking, troubleshooting, and incident investigations
 - ◇ The system, where applicable, records the source IP address and time from where the call related to the activity was made



7.21 CloudTrail Dashboard in the AWS Management Console

CloudTrail

Dashboard

Event history

Trails

Dashboard

View events in your AWS account for the last 90 days, create trails, and manage existing trails. [Learn more](#)

View trails

Recent events

These are the most recent events recorded by CloudTrail. To view all events for the last 90 days, go to Event history.

	Event time	User name	Event name	Resource type
▶	2019-01-15, 10:38:52 AM	S9	RunInstances	EC2 VPC and 6 more
▶	2019-01-15, 10:38:51 AM	S9	AuthorizeSecurityGroupIngress	EC2 SecurityGroup
▶	2019-01-15, 10:38:50 AM	S9	CreateSecurityGroup	EC2 VPC and 1 more
▶	2019-01-15, 09:59:57 AM	S7	DeleteKeyPair	EC2 KeyPair
▶	2019-01-15, 09:54:49 AM	S7	TerminateInstances	EC2 Instance

View all events

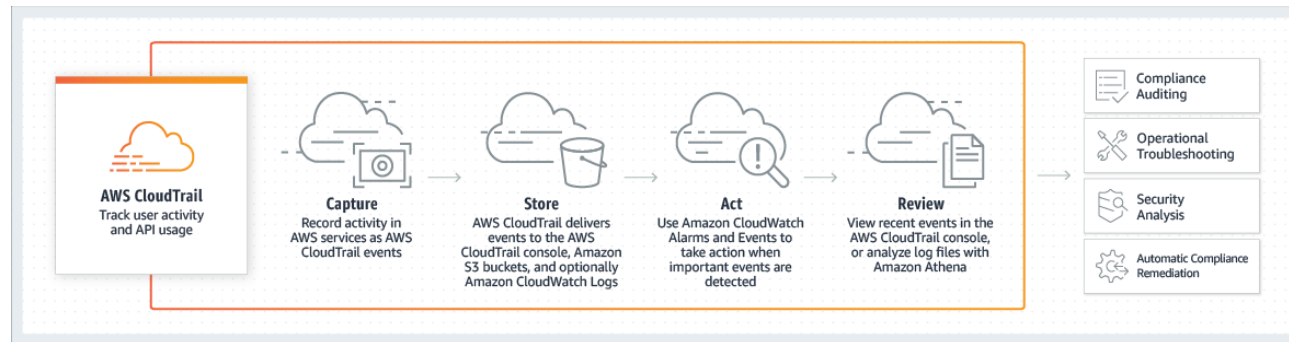


7.22 CloudTrail Integration with CloudWatch

- CloudTrail is integrated with Amazon CloudWatch Logs for log data search
- Integration with CloudWatch Events service enables users to set up workflows triggered in response to events that can potentially pose security threat
 - ◇ An example of such an event is an attempt to make a private S3 bucket public



7.23 How CloudTrail Works



Source: AWS Documentation



7.24 CloudTrail Use Cases

- Compliance
 - ◇ CloudTrail dramatically simplified compliance audits for out-of-compliance events and fast responses to auditing requests
 - ◇ CloudTrail ensures compliance with internal policies and regulatory standards by providing a history of activity in your AWS account. For more information, visit <https://bit.ly/2Reo6K4>
- Security Analysis
 - ◇ You can analyze user behavior patterns by ingesting AWS CloudTrail events history into your log management and analytics solutions
- Data Exfiltration Detection (an act of unauthorized copying, transfer or retrieval of data)
 - ◇ Done via S3 object-level API events recorded in CloudTrail
 - ◇ You can also configure an action in response to such events using Amazon CloudWatch Events and AWS Lambda
- Operational Issue Troubleshooting
 - ◇ Done using AWS API call history produced by AWS CloudTrail
 - ◇ For example, you can quickly identify changes done to Amazon VPC security groups



7.25 Trusted Advisor

- AWS on-line expert service that can analyze your AWS environment and offers recommendations that can help you reduce cost, tighten security, increase performance, improve fault tolerance, and optimally provision resources following AWS best practices
- There are two tiers of Trusted Advisor service:
 - ◇ **Free** - available to all AWS customers
 - This type performs only core checks
 - ◇ **Business or Enterprise** - based on support plans



7.26 Trusted Advisor Dashboard



Recommended Actions

▶	Security Groups - Specific Ports Unrestricted	Refreshed: 2 minutes ago	
	Checks security groups for rules that allow unrestricted access (0.0.0.0/0) to specific ports. 140 of 291 security group rules allow unrestricted access to a specific port.		
▶	MFA on Root Account	Refreshed: 2 minutes ago Previous status: Yellow	
	Checks the root account and warns if multi-factor authentication (MFA) is not enabled. MFA is not enabled on the root account.		
▶	VPC Elastic IP Address	Refreshed: 2 minutes ago Previous status: Green	
	Checks for usage that is more than 80% of the VPC Elastic IP Address Limit. 1 of 16 items have usage that is more than 80% of the service limit.		
▶	RDS Subnets per Subnet Group	Refreshed: 2 minutes ago Previous status: Green	
	Checks for usage that is more than 80% of the RDS Subnets per Subnet Group Limit. 1 of 4 items have usage that is more than 80% of the service limit.		
▶	IAM Use	Refreshed: 2 minutes ago	



7.27 Summary

- In this chapter, we reviewed the elements of AWS monitoring capabilities and ways to manage AWS resources and user applications

Chapter 8 - Amazon DynamoDB

Objectives

Key objectives of this chapter

- NoSQL database storage types
- Amazon DynamoDB



8.1 Main NoSQL Database Storage Types

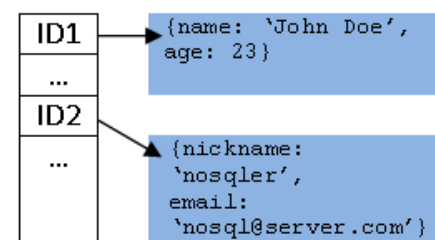
- Conventionally, NoSQL systems are classified into three main categories by their storage types:
 - ◇ Key-Value stores
 - ◇ Column stores
 - ◇ Document stores

Key / Value

Key1	Value1
Key2	Value2
...	...
KeyN	ValueN

Column

Document





8.2 Amazon DynamoDB

- Fully managed NoSQL database service
- Offers flexible key-value and document data store models
- Provides consistent, single-digit millisecond latency at any scale
- Suitable for applications with ~100K IOPS with sharding to support high throughput
 - ◇ Can handle 10 trillion (1,000,000,000,000; one million million; 10^{12} ;) requests per day with a peak of over 20 million TPS
- Uses SSDs
- As a NoSQL database, DynamoDB is not designed to support transactions
 - ◇ Some transactional support is provided through the DynamoDB Streams service
 - DynamoDB Streams allow you to tap into and capture data modification events in DynamoDB tables (more information to a bit later...)



8.3 DynamoDB is an Eventual Consistent NoSQL Database

- DynamoDB is a PA system as per the CAP triangle's designation
- You can ensure strong consistency for reads by specifying some optional parameters in the read request
 - ◇ You could observe longer response times for such requests



8.4 DynamoDB Tables are Created in and Tied to a Region

- If your business spans, say, three geographies, e.g. Europe, Asia, and the US, and you need to setup and maintain the Customer DynamoDB-based table to hold your clients' profiles, you will end up having three identical tables created in three different regions
 - ◇ **Note:** You might want to use S3 or AWS Data pipeline to replicate the table(s) [<https://aws.amazon.com/blogs/aws/cross-region-import-and-export-of-dynamodb-tables/>]
- AWS offers you a generic solution in the form of the **Global table** (see the related discussion later in this module)



8.5 Accessing DynamoDB

- AWS Management Console
- CLI (the aws tool)
- The API
 - ◇ **Note:**
 - Applications connect to DynamoDB via an HTTP/s end-point



8.6 DynamoDB Core Components

- DynamoDB offers users these components to work with: **tables**, **items**, and **attributes**
- A table is a collection of items
 - ◇ Item has a size limit of 400K
- An item is a collection of attributes (think of it as a record with a varying number of attributes)
- An attribute is a key-value pair
 - ◇ Multi-valued attributes are supported; those are organized into sets that do not allow duplicates
- DynamoDB uses a **primary key** to uniquely identify each item in a table and (optionally) secondary indexes to provide more querying flexibility
- Other than the primary key, DynamoDB tables are **schemaless**



8.7 Data Types

- **Scalar** data type:
 - ◇ String, Number, Boolean, Null
 - ◇ Some scalars have size limits
- **Set** data type
 - ◇ Represent a unique list of one or more scalars
- **Document** data types
 - ◇ List and Map
 - ◇ Modeled after JSON, allow for nested structures



8.8 Example of an Item with Attributes

```
{  
  "id" : 100001,  
  "txnName" : "Pay_Bill",  
  "payees" : [234, 567, 890],  
  "details": {  
    "refNumber" : "RC_200"  
    "txnDateTime" : "Dec 31, 23:59:59:999"  
  }  
}
```




8.9 The Primary Key

- Tables have the **Primary Key** to uniquely identify each item
- You can have two or more attributes making up the primary key
 - ◇ E.g. “Artist”, “Album”, and “Song_Title”
- Each primary key attribute must be a scalar: string, number, or binary
- The primary key can be a composite key made of two keys: one is **partition key**, the other **sort/range key**.



8.10 Partition Key and Sort Key

- Each table must have a **Partition Key** that in, simple scenarios, is the primary key
 - ◇ Partition key is used to point to a data shard location
- Optionally, the **Sort Key**
 - ◇ Used to support rich queries against items using such conditionals as “begins with”, “between”, “counts”, “==, <, >, <=, =>”, “top/bottom N values”, etc.
- Two items in a table may have the same partition portion of the primary key, but then they must have different sort/range key values
 - If a sort key is set up, data is arranged by the sort key within a partition
- You cannot change this configuration after a table is created



8.11 Example of a Table with Partition Key and Sort Key

Table name: **Sensor_Readings**

Partition Key: **Sensor_ID**

Sort Key: **epoch**

Attributes: **Voltage, Voltage_Polarity**

1001	1545082379885	Voltage=4.89, Voltage_Polarity="P"
1001	1545082383888	Voltage=4.82, Voltage_Polarity="P"
1001	1545082385056	Voltage=4.91, Voltage_Polarity="P"
9987	1545082386196	Voltage=3.89, Voltage_Polarity="N"
1001	1545082387557	Voltage=4.81, Voltage_Polarity="P"
9987	1545082388537	Voltage=4.01, Voltage_Polarity="N"



8.12 Secondary Indexes

- In addition to the primary key, you are allowed to define one or more secondary indexes, which come in two types:
 - ◇ Global Secondary Index (GSI)
 - An index across all partitions
 - You can have multiple GSIs
 - ◇ Local Secondary Index (LSI)
 - LSI is an index within a partition
 - You can have only one local LSI
- You can define up to 5 global secondary indexes and 5 local secondary indexes per table



8.13 Provisioned Capacity

- When you create a DynamoDB table, you are required to specify the read/write capacity you need for your expected workloads
- DynamoDB then will go ahead and allocate the needed infrastructure capacity to meet your requirements with sustained, low-latency response time
- Overall capacity is measured in read and write capacity units
 - ◇ These values can be adjusted later after monitoring application's performance using AWS CloudWatch service to fine-tune your scaling decisions
- If your application exceeds the provisioned throughput capacity on a table or index, it is subject to request throttling



8.14 Partition Metrics

- A single partition
 - ◇ Can hold about 10GB of data
 - ◇ Supports 3,000 Read capacity units (RCUs) and 1,000 Write capacity units (WCUs)



8.15 Maximizing DynamoDB Throughput

- Follow these basic rules when designing your table:
 - ◇ Ensure access to the table is evenly spaced over the key-space
 - ◇ Ensure that requests arrive evenly spaced in time



8.16 Summary

- In this chapter we discussed the following concepts and components related to Amazon DynamoDB:
 - ◇ NoSQL database storage types
 - ◇ Amazon DynamoDB data model
 - ◇ Partitions and Sort Keys
 - ◇ Capacity and Throughput

Chapter 9 - AWS Storage Options

Objectives

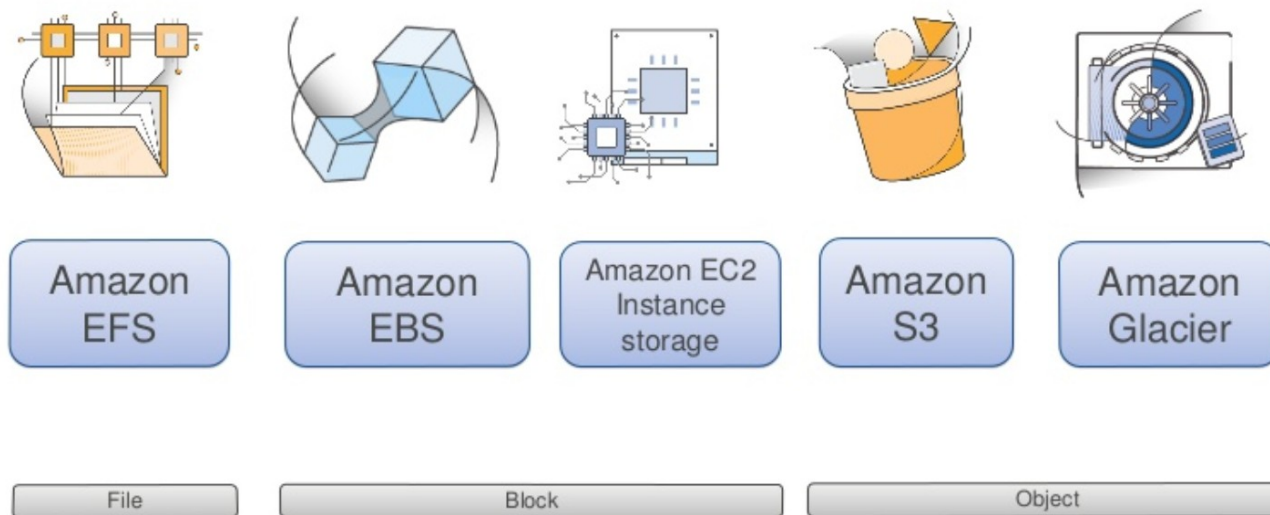
Key objectives of this chapter

- Learn about additional AWS storage options
- AWS EFS
- AWS Storage Gateway
- AWS Snowball and Snowmobile



9.1 Storage Types in AWS

- In AWS we will find three types of storage options
 - ◇ Block Storage (EBS and
 - ◇ File Storage
 - ◇ Object Storage





9.2 Overview of AWS Storage Offerings

- Elastic Block Service (EBS) and Instance Store
 - ◇ Block storage requirements for single EC2 and RDS instances
- Simple Storage Service (S3)
 - ◇ Object storage and content delivery over HTTP/HTTPS
- Glacier
 - ◇ Archiving and long term storage at the lowest prices
- **Elastic File System**
 - ◇ Block storage requirements for distributed access to multiple EC2 instances
- **AWS Storage Gateway**
 - ◇ Hybrid on-premise and AWS storage solution
- **Snowball and Snowmobile**
 - ◇ Physical transfer of data from on-prem to AWS



9.3 Introduction to EFS

- A fully managed, high performance, highly available, and highly scalable network file system
- Delivers a standard Network File System v4 (NFSv4) protocol
- Is able to deliver NFS data-stores of unlimited sizes
- Designed to match the characteristics and durability and availability of S3
 - ◇ Replicated across multiple availability zones within a region
- No upfront fee for using it - we only pay for the storage consumed



9.4 EFS Characteristics

- High performance
 - ◇ Over 7000 file operations per second per file-system
- Replication
 - ◇ EFS-to-EFS or EFS-to-NFS
 - ◇ Backup copies, multiple offices, global websites etc.
- Encryption
 - ◇ AES 265
 - ◇



9.5 Introduction to Storage Gateway

- Software appliance that allows us to seamlessly connect our on-premises environment with AWS cloud-based storage back-ends
- A virtual machine in our on-premises data center and is able to deliver file-based storage, data volumes, and a virtual tape library device backed by Amazon S3 and Glacier



9.6 Storage Gateway Types

- File Gateway
 - ◇ A network file system service that allows you to access files on S3 via standard NFSv3, NFSv4.1, SMB 2, and SMB 3 protocols
- Volume Gateway
 - ◇ A service that allows us to mount block-level volumes stored on S3 via iSCSI protocol with two modes of operation:
 - Locally Cached Volumes: Data is served from S3 and only caches frequently read data and buffers writes on the local Storage Gateway
 - Locally Stored Volumes: All data is served from the local volume and regular snapshots are taken of the local volume and stored to S3
- Tape Gateway
 - ◇ A service that presents a virtual tape library that can be used with our traditional on-premises backup solutions



9.7 Data transfer to/from AWS

- Big data – big transfer times
 - ◇ 1Gb file on a 1Gbit line - about 8 seconds
 - ◇ 1TB file on a 1Gbit line - about 2.5 hours
 - ◇ 10B file on a 1Gbit line - about 25 hours
 - ◇ 1PB file on a 1Gbit line - about 107 days
- Questions to ask?
 - ◇ What is my data-set?
 - ◇ How often is the data-set created?
 - ◇ How long will it take to transfer to AWS?
 - ◇ Does it makes sense increasing the throughput of the up-link?
- For Large data-sets AWS offers an alternative
 - ◇ Physically transfer the data using Snowball and Snowmobile



9.8 Introduction to Snowball and Snowmobile

- Physical storage for transporting large data sets (PB and ZB-scale) from on-premise to AWS
- Snowball
 - ◇ Physical device with disks that supports 50TB or 80TB of raw storage
 - ◇ Has one 10GbE connection for data transfer
 - ◇ Ability to use multiple Snowballs for larger data-sets
 - ◇ Ability to use for one-time migrations or repetitive jobs (weekly, monthly full backup etc.)
 - ◇ Encrypted and secured in a tamper proof case
 - ◇ Order from AWS, transfer data and use included next-day shipping
- Snowmobile
 - ◇ An 18-wheeler with a 45ft shipping container full of disks
 - ◇ Support up to 100PB of physical storage
 - ◇ Supports approx. 1Tb/s of data transfer
 - ◇ Can transfer 100PB in less than 10 days



9.9 Summary

- In this chapter we looked at the storage concepts in AWS. We took a look at the services that have not yet been covered in the preceding chapters, namely:
 - ◇ The Elastic File System (EFS)
 - ◇ The Storage Gateway
 - ◇ The Snowball and Snowmobile

Chapter 10 - Elastic Compute Cloud High Availability

Objectives

Key objectives of this chapter

- Overview of
 - ◇ Making Applications Highly Available
 - ◇ Introduction to Autoscaling



10.1 Amazon Elastic Compute Cloud (EC2)

- Amazon EC2 provides scalable compute capacity
- EC2 instances are virtual machines run as an unmanaged service
- How do we make the applications running in EC2 highly available?
- How can we meet the user demand and scale our application accordingly?



10.2 High Availability in EC2

- EC2 instances always map to VPC subnets
- Each VPC network should be designed with subnets in separate availability zones
- Launching instances into separate subnets thus distributes EC2 instances across the region for high availability



10.3 Challenges of High Availability

- Replication or delivery of content across instances
- Maintaining state in web front-ends
- Distribute traffic across instances
- Determining fail-over procedures



10.4 Challenges solved

- Create stateless instances and deliver content from other services (S3, RDS, DynamoDB etc.)
- Maintain state outside of the application (DynamoDB, ElastiCache..)
- Introduce Elastic Load Balancing to distribute traffic
- Automate fail-over by implementing AutoScaling



10.5 Elastic Load Balancing

- The load balancing service in AWS
- Allows for traffic distribution across EC2 instances, ECS tasks and other network targets (including external and on-premise servers)
- Ability to load balance across availability zones (cross-zone load balancing)
- Three types of load balancers
 - ◇ Classic Load Balancer (layer 4 + some layer 7)
 - ◇ Application Load Balancer (layer 7)
 - ◇ Network Load Balancer (layer 4)



10.6 Classic Load Balancer

- Previous Generation
- Supports layer 4 distribution of traffic across instances
- Supports some layer 7 functionality
 - ◇ Sticky sessions
 - ◇ XForwardedFor header
 - ◇ HTTPS offloading
- Supports health checks
 - ◇ HTTP/HTTPS 200 response
 - ◇ TCP port open
 - ◇ SSL connection handshake



10.7 Application Load Balancer

- Next Generation
- Understands layer 7 packets
- Can forward traffic to multiple groups according to path
 - ◇ www.example.com/images – to image back-end
 - ◇ api.example.com – to the API back-end
 - ◇ example.com/threads – to the forum back-end
- Perfect for modern applications and micro-services
- Supports direct integration with the Web Application Firewall (WAF)
- Supports all health checks and layer 7 functions of classic load balancer



10.8 Network Load Balancer

- Next Generation
- Super-fast pure layer 4 (up to millions of connections per second)
- Works on a single static IP per availability zone
- Multiple static IPs across multiple zones for high availability
- Forwards packets with destination IP visible
- Does not support any layer 7 functionality
- Perfect for web-scale micro-services and latency-sensitive applications



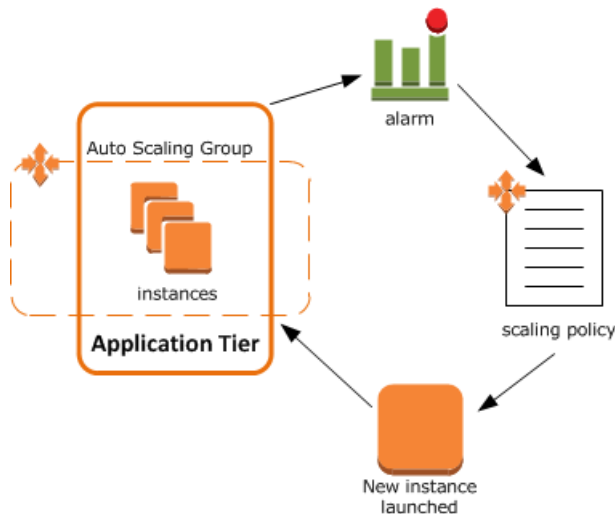
10.9 Scaling

- Single instance capacity is limited and can serve X amount of users
- Multiple instances in parallel can serve multiples of X
- Determine metrics to scale upon
- Define policies for scaling
- Notify load balancer of scaling events



10.10 EC2 AutoScaling

- Allows for integration with ELB and CloudWatch to scale according to load on the environment
- Determine metrics (CPU load, number of connections, custom metrics etc.)
- Determine thresholds – at what % or level and after how long should scaling kick in (example CPU above 60% for 10 minutes)
- Determine actions – what happens when the alarm goes off (add one instance, add 20% more instances, remove instances etc.)





10.11 Summary

- In this module we reviewed the ability to use EC2 as a service that can meet the scale of the incoming traffic. We learned about the following technologies:
 - ◇ EC2
 - ◇ ELB
 - ◇ CloudWatch Alarms
 - ◇ AutoScaling

Chapter 11 - Orchestration in AWS

Objectives

Key objectives of this chapter

- Infrastructure as Code
- Introduction to CloudFormation
- Introduction to OpsWorks
- Introduction to Elastic Beanstalk



11.1 Overview of Orchestration in AWS

- AWS provides us with a set of tools and services that allow for highly customized orchestration of deployment, management, operations, installation and delivery of applications
- Tools used
 - ◇ CloudFormation
 - ◇ OpsWorks
 - ◇ Elastic Beanstalk
 - ◇ AWS Systems Manager



11.2 Why Orchestration?

- Deploy infrastructure in an automated manner
- Automate the management tasks and operations
- Automate application installs and patching
- Automated deployments
- Simplify the infrastructure



11.3 Infrastructure as Code

- The IaC concept defines that we treat our infrastructure in the same manner as we would the code that we develop.
- IaC allows us to:
 - ◇ Deliver fully automated infrastructure deployments, updates and decommissioning
 - ◇ Increase repeatability - use the same specification document in multiple environments (Dev, Test, Prod)
 - ◇ Increase reliability - test the infrastructure like we would test code (Test, QA, Staging...) and integrate with CD/CD tools
- IaC requires us to:
 - ◇ Determine our infrastructure deployment through a specification document
 - ◇ Store the specification document in a repository
 - ◇ Enable versioning of the document to track changes



11.4 CloudFormation

- The standard tool for delivering IaC deployments in AWS
- Has the following components
 - ◇ The CloudFormation Template – specifies the infrastructure to deploy
 - ◇ The CloudFormation Storage and versioning – stores the template (on S3) and allows for automated versioning through change sets
 - ◇ The CloudFormation Engine – performs the infrastructure deployment (creates stacks)
 - ◇ The CloudFormation Stack – a set of connected cloud resources that are deployed from a template



11.5 CloudFormation Template

- Specification document written in JSON or YAML
- Can be written in any text editor or WYSIWYG tool
- AWS CloudFormation Designer



11.6 Structure of a CF Template

- AWS Template Format Version:
 - ◇ Denotes the CloudFormation template version that the template conforms to
- Description:
 - ◇ Arbitrary string of text that will help us understand what the template does
- Metadata:
 - ◇ Information that we would like to pass to the services and instances that are being deployed
- Parameters:
 - ◇ AWS values to be passed to the resources section
- Mappings:
 - ◇ Mappings can optionally be created to apply the appropriate parameters to the appropriate environment or region
- Conditions:
 - ◇ Conditions can optionally determine whether a resource is created or not, depending on a certain value



- Transform:
 - ◇ The Transform section can optionally be used with AWS Serverless components like AWS Lambda to define specifics and versions of the AWS Serverless Application Model to be used in the template
- Resources:
 - ◇ This is the only mandatory part of the template as it defines the actual resources that the CloudFormation service will be creating
- Outputs:
 - ◇ We can use the Outputs section to return a certain information upon deployment



11.7 Example CF Template - Parameters

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "This template creates a t2 EC2 instance using the Amazon Linux AMI and allows SSH access.",
  "Parameters" : {
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",
      "Type" : "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription" : "Must be an existing EC2 KeyPair."
    },
    "InstanceType" : {
      "Description" : "The EC2 instance type",
      "Type" : "String",
      "Default" : "t2.micro",
      "AllowedValues" : [ "t2.micro", "t2.small", "t2.medium", "t2.large" ],
      "ConstraintDescription" : "Must be one of the t2 EC2 instance types."
    }
  }
},
```

Notes:

The parameters define the KeyName that needs to be an existing key from an EC2 account – there is a constraint put on it. The other one is InstanceTypes – this one limits us to different t2 instance types



11.8 Example CF Template - Resources

```
"Resources" : {  
  "EC2Instance" : {  
    "Type" : "AWS::EC2::Instance",  
    "Properties" : {  
      "InstanceType" : { "Ref" : "InstanceType" },  
      "SecurityGroups" : [ sg-asdfg12345 ],  
      "KeyName" : { "Ref" : "KeyName" },  
      "ImageId" : [ ami-asdfg12345 ]    }  
    }  
  },  
}
```

Notes:

The resources being deployed pull the key name and instance type from the parameters section – point out the Ref: sections in bold..



11.9 Example CF Template - Outputs

```
"Outputs" : {  
  "InstanceId" : {  
    "Description" : "InstanceId of the newly created EC2 instance",  
    "Value" : { "Ref" : "EC2Instance" }  
  },  
  "PublicIP" : {  
    "Description" : "Public IP address of the newly created EC2 instance",  
    "Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicIp" ] }  
  }  
}
```

Notes:

The output will display (in bold) the instance ID and the IP address that was assigned to the instance when it was deployed.



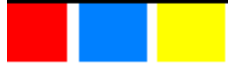
11.10 OpsWorks

- Fully managed configuration management service that delivers a cloud-based Puppet Enterprise or Chef Automate environment
- Fully compatible with Chef/Puppet - uses industry standard Chef recipes/cookbooks and Puppet modules/manifests
- With OpsWorks, we get complete control within our operating system. We can control any aspect of the configuration, including the following:
 - ◇ Software, patch, and update installation
 - ◇ Controlling system configurations and operating system settings
 - ◇ Running scripts and applications
 - ◇ Performing regular maintenance jobs
 - ◇ Modifying files
- Essentially, we are able to control any aspect of the operating system with changes ranging from a few characters in a file to a completely automated install and configuring tasks.



11.11 OpsWorks Use-cases

- Some typical use cases for OpsWorks are:
 - ◇ Legacy applications where a redeployment is not an option
 - ◇ Hybrid environments where we need unified control of both AWS and on-premise resources
 - ◇ Compliance with a certain configuration
 - ◇ Large clusters with a lot of small dynamic changes
 - ◇ Updating and patching fleets of servers
 - ◇ Migrating existing Chef or Puppet environments to a managed solution



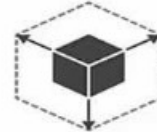
11.12 OpsWorks Operations



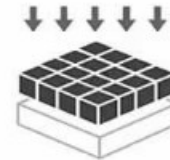
A **stack** represents the compute infrastructure and applications that you want to manage together.



A **layer** defines how to set up and configure a set of instances and related resources.



Decide how to scale: manually, with **24/7** instances, or automatically, with **load-based** or **time-based** instances.



Then deploy your app to specific instances and customize the deployment with Chef recipes.



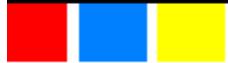
11.13 Elastic Beanstalk

- Allows us to focus on the code
- Frees our developers and SysOps engineers from having to design, deliver, operate, and maintain the infrastructure
- Can handle all the aspects of the deployment:
 - ◇ Instance and operating system deployment
 - ◇ Installation of the programming language interpreter
 - ◇ An HTTP service to present the Elastic Beanstalk application
 - ◇ AWS services (ELB, RDS, SQS, and so on)
 - ◇ Custom prerequisites such as frameworks, runtimes, libraries, and modules

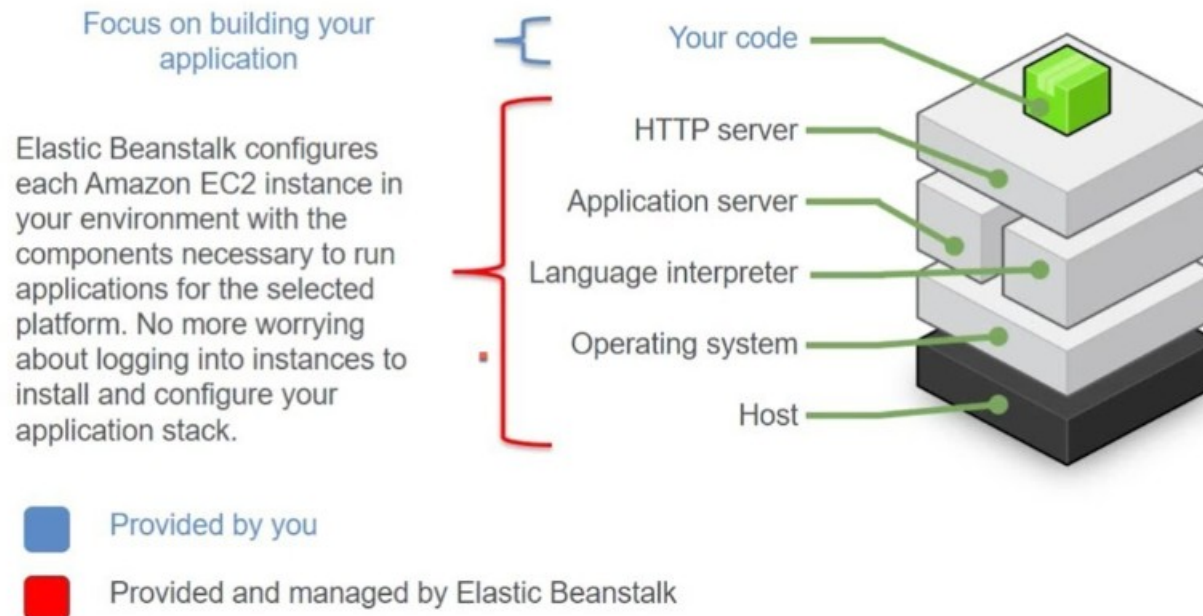


11.14 Elastic Beanstalk Components

- Application
 - ◇ A logical grouping for multiple environments
- Environment
 - ◇ A specific version of a programming language and platform to run on
 - ◇ Can have multiple versions of the application running in multiple environments
 - ◇ Web tier - application front-end that serves the Elastic Beanstalk page
 - ◇ Worker tier - application back-end that performs tasks
- Configuration
 - ◇ Contains all of the necessary information for the Elastic Beanstalk service to build, deploy, and even customize the environment before it's ready to serve the code



11.15 Elastic Beanstalk Architecture



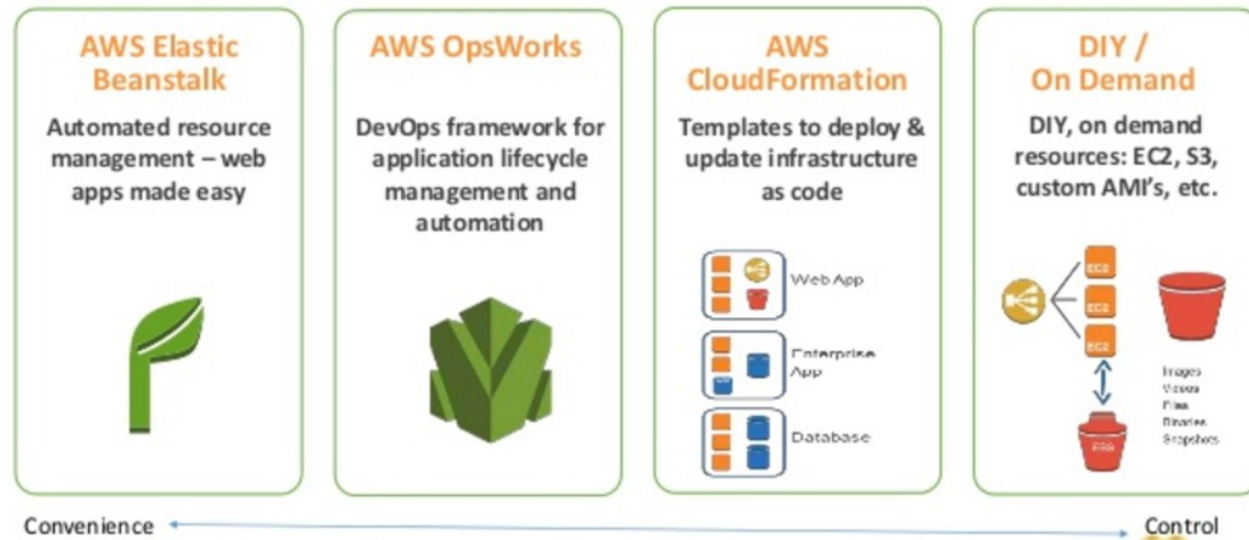


11.16 Elastic Beanstalk Supported Platforms

- Packer Builder
- Docker
- Go programming language 1.11 version 2.9.4
- Java SE 7 and 8 and Java 6, 7, and 8 on Tomcat
- A variety of the .NET Framework and Core versions running on Windows server 2008-2016 and IIS Elastic Beanstalk server version 7.5, 8, 8.5, and 10
- Node.js versions 4, 5, 6, 7, 8, and 10
- PHP language versions 5.4 to 7.2
- Python language versions 2.6 to 3.6
- Ruby language versions 1.9 to 2.6



11.17 Comparison of deployment modes





11.18 Summary

- In this module we reviewed the tools and services that allow us to deliver orchestration and automation in AWS. We took a look how to choose the correct service and discussed the following services:
 - ◇ CloudFormation
 - ◇ OpsWorks
 - ◇ Elastic Beanstalk

Chapter 12 - DNS in AWS

Objectives

Key objectives of this chapter

- Introduction to Route53
- Working with Hosted Zones and DNS Records
- DNS Health Checks



12.1 Introduction to Route53

- A smart DNS service - “the next generation of DNS”
- Is API addressable allowing for automation of the provisioning of our infrastructure and the DNS records
- Has 100% SLA for service delivery
- Has the ability to manage the traffic flow to our application
- Built-in DNS health checks and DNS fail-over allow us to automatically exclude IP addresses of servers that are non-operational from the DNS response
- Can be integrated with CloudFront, S3, and ELB allowing us to easily configure and maintain the websites served through these services on Route 53



12.2 Route 53 hosted zones

- Ability to register domains
- Create hosted zones and start adding records
- Public or private hosted zones are supported
- Integrates with ECS for service discovery
- Most standard DNS record types are supported
 - ◇ A, AAAA, CAA, CNAME, MX, NAPTR, NS, PTR, SOA, SPF, SRV, TXT

Notes:

See <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html> for supported record type details



12.3 Route 53 Routing Policies

- Simple routing
 - ◇ One hostname to one IP
- Latency-based routing
 - ◇ Finds closest origin based on measured latency to the client
- Weighted routing
 - ◇ Assign different weight to different IPs (Very useful in Blue/Green Deployments)
- Fail-over routing
 - ◇ Allows for health checking the destination and failing over one site is not accessible
- Geo-location and geo-proximity routing
 - ◇ Finds best origin based on geo-location or geo-proximity settings
- Multi-value response
 - ◇ One hostname can return up to 10 answers (Very useful for peer-to-peer, video streaming etc.)



12.4 Route 53 DNS Health Checks

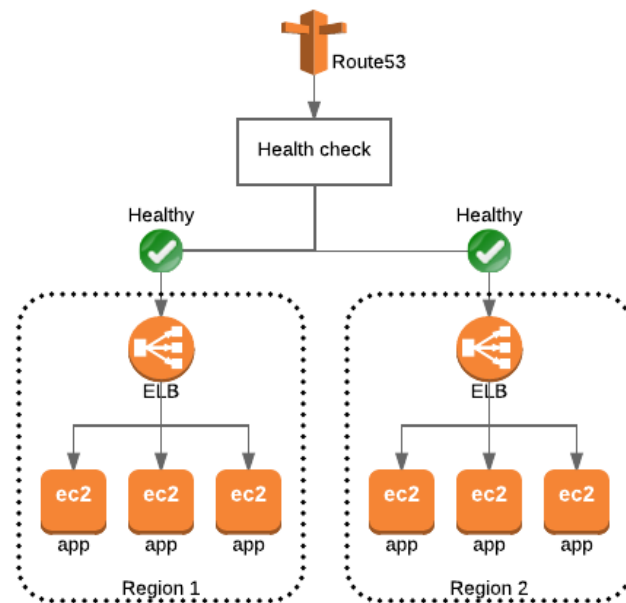


Figure 1 - Both regions operating normally

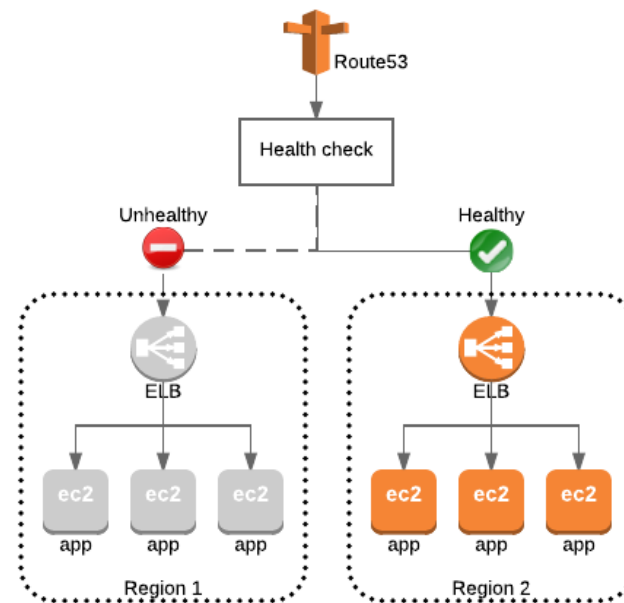


Figure 2 - region 1 experiencing issues



12.5 Summary

- In this module we have taken a quick look at the AWS Route 53 service that allows us to deliver DNS services to our application in an automated fashion. We also took a look at how Route 53 can be used to deliver high availability across multiple sites or regions.

Chapter 13 - Passing the exam

Objectives

Key objectives of this chapter

- Exam Overview
- Exam Resources



13.1 Exam Overview

- The Practitioner exam tests the ability to demonstrate an overall understanding of the AWS Cloud and demonstrates that the exam taker understands how to:
 - ◇ Define what the AWS Cloud is
 - ◇ Describe basic AWS Cloud architectural principles
 - ◇ Describe the AWS Cloud value proposition
 - ◇ Describe key services on the AWS platform
 - ◇ Describe basic security and compliance aspects of the AWS platform
 - ◇ Define the billing, account management, and pricing models
 - ◇ Identify sources of documentation or technical assistance
 - ◇ Describe basic/core characteristics of deploying and operating in the AWS Cloud



13.2 Knowledge Prerequisites

- Required basic understanding of IT services and their uses in the AWS Cloud platform
- Recommended at least six months of experience with the AWS Cloud in any role



13.3 Exam Structure

- The exam has approximately 60 questions which you will need to answer in 90 minutes of time
- Questions are scored in full, one wrong answer means the question is marked as incorrect
- Empty responses are marked as incorrect
- Make sure to answer as many questions as you can
- If you get stuck on a question mark it for review and return to it at the end
- It is easy to lose track of time, you have about 1.5 minutes per question so getting stuck on a question can cause you to fail the exam
- There are two types of questions on the exam:
 - ◇ Multiple-choice: Has one correct response and three incorrect responses (distractors)
 - ◇ Multiple-response: Has two correct responses out of five options.



13.4 Knowledge Domains

- Domain 1: Cloud Concepts - 28%
- Domain 2: Security - 24%
- Domain 3: Technology - 36%
- Domain 4: Billing and Pricing - 12%



13.5 Domain 1: Cloud Concepts

- Define the AWS Cloud and its value proposition
- Identify aspects of AWS Cloud economics
- List the different cloud architecture design principles



13.6 Domain 2: Security

- Define the AWS Shared Responsibility model
- Define AWS Cloud security and compliance concepts
- Identify AWS access management capabilities
- Identify resources for security support



13.7 Domain 3: Technology

- Define methods of deploying and operating in the AWS Cloud
- Define the AWS global infrastructure
- Identify the core AWS services
- Identify resources for technology support



13.8 Domain 4: Billing and Pricing

- Compare and contrast the various pricing models for AWS
- Recognize the various account structures in relation to AWS billing and pricing
- Identify resources available for billing support



13.9 Preparing for the exam

- Before taking the exam you should establish your level of preparedness
- Read the AWS FAQ on a certain topic like EC2 or S3 and try to answer the questions without peeking at the answer
 - ◇ If you can answer the FAQ questions you should be able to pass the exam – if not then additional learning is required
 - ◇ For practitioner focus on the basic FAQ for each service
- When preparing for the exam we should consider the following options:
 - ◇ Training
 - ◇ AWS exam-prep literature
 - ◇ AWS test questions
 - ◇ Real world experience



13.10 Additional Exam Resources

- Recommended reading to prepare for the Practitioner exam is as follows:
 - ◇ Overview of Amazon Web Services whitepaper
 - ◇ Architecting for the Cloud: AWS Best Practices whitepaper
 - ◇ How AWS Pricing Works whitepaper
 - ◇ The Total Cost of (Non) Ownership of Web Applications in the Cloud whitepaper
 - ◇ Compare AWS Support Plans Website

Notes:

- ◇ Overview of Amazon Web Services whitepaper - <https://d0.awsstatic.com/whitepapers/aws-overview.pdf>
- ◇ Architecting for the Cloud: AWS Best Practices whitepaper - https://d0.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf
- ◇ How AWS Pricing Works whitepaper- https://d0.awsstatic.com/whitepapers/aws_pricing_overview.pdf
- ◇ The Total Cost of (Non) Ownership of Web Applications in the Cloud whitepaper - <https://d1.awsstatic.com/whitepapers/aws-tco-web-applications.pdf>
- ◇ Compare AWS Support Plans - <https://aws.amazon.com/premiumsupport/plans/>



13.11 Summary

- In this module we have taken an overview of the exam requirements, structure and resources for you to prepare for the exam.