






















AWS Certified Developer Notes










By William Horowitz

Contents

AWS Overview and Terminology	5
Shared Responsibility Model	8
Management & Security	10
Identity Access Management (IAM) 	10
5. Benefits	10
6. IAM Components	10
7. User-Based v Resource Based Policies	12
8. Best Practices	12
9. Other Security measures.....	13
Trusted Advisor 	13
CloudFormation 	13
OpsWorks 	15
Simple Workflow Service (SWF) 	16
CloudWatch 	18
Storage	20
Simple Storage Service (S3) 	20
2. Web Store	20
3. Cross-Origin Resource Sharing (CORS).....	21
4. S3 security	21
5. S3 Performance.....	21
6. S3 Storage Classes	22
7. Multipart Upload.....	22
8. Lifecycle Management	22
9. Versioning	22
10. Cross Region Replication.....	22

Elastic File System (EFS) 	23
2. Advantages of EFS	23
4. Mount Targets.....	24
5. Accessing File System from EC2	24
Snowball 	25
Storage Gateway 	25
Database	26
Relational Database Service (RDS) 	26
Database Migration Service (DMS) 	27
DynamoDB 	28
2. Advantages of DynamoDB	28
4. Tables, Items, Attributes & Primary Keys	28
5. Secondary Indexes	29
6. Query and Scan	29
7. Additional DynamoDB Operations.....	30
8. Atomic Counters vs. Conditional Updates	30
9. Optimistic Locking.....	31
10. Provisioned Throughput	31
11. DynamoDB Streams	31
ElastiCache 	32
2. Memcached	32
3. Redis.....	33
5. Caching Strategies.....	33
Compute.....	34
Elastic Cloud Compute (EC2) 	34
1. Introduction	34
2. Purchasing Options	34
3. Instance Types Overview	35

5. EC2 Storage Options	35
6. EC2 Metadata.....	36
7. Amazon Machine Images (AMI).....	37
8. Cluster Networking	37
9. More information	37
EC2 Container Service (ECS) 	38
Elastic Beanstalk 	38
Elastic Load Balancer (ELB) 	39
1. Introduction	39
2. Types of Load Balancers.....	39
3. Session State	39
Networking.....	40
Virtual Private Cloud (VPC) 	40
1. Introduction	40
3. AWS VPC	41
4. Connecting to a VPC.....	41
6. Route Tables	41
8. Network Address Translation (NAT)	42
9. VPC Security	42
CloudFront 	43
1. Introduction	43
2. CloudFront Origin Server	43
3. CloudFront Distribution	43
4. Delivering Dynamic Content	43
Route 53 	44
1. Quick Overview of Domain Name System (DNS)	44
2. Introduction	44
3. Key Concepts.....	44

4. Policy Routing Types	45
5. Weighted Routing	45
Messaging	46
Simple Queuing Service (SQS) 	46
2. Decoupling Processes	46
3. Queue Types	47
4. Message Lifecycle/Visibility timeout	47
5. Dead Letter Queues	47
6. Delay Queues and Message Timers	47
7. Types of Polling	48
Simple Notification Service (SNS) 	48
2. Transport Protocols.....	48
3. Message Format.....	48
4. More documents.....	50
Analytics & Big Data	50
RedShift 	50
Elastic MapReduce (EMR) 	51
Elasticsearch 	51
QuickSight 	52
Machine Learning 	52
Kinesis 	52
Application Services	53
API Gateway 	53
Deployment in AWS	54

AWS Overview and Terminology

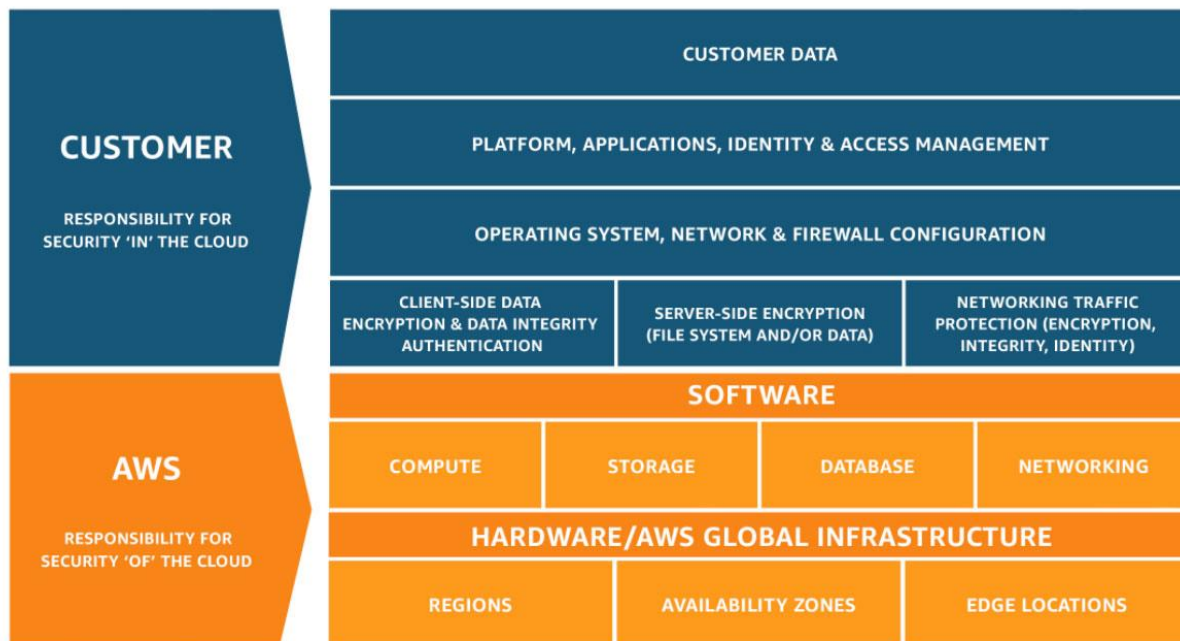
- Regions – service is broken up into geographic areas (e.g. Virginia/us-east-1)
 - Availability Zones (AZ) – 2 or more areas within a region which correspond to a data center
 - Provide redundancy to compensate for when problems occur
 - Edge Locations – a CDN endpoint for cloud front
 - Used to cache files closer to a user's physical location
 - Initial requests are made to the closest region, but subsequent requests can be made from the edge location cache to ensure low latency
 - Find AWS Global Infrastructure at <https://aws.amazon.com/about-aws/global-infrastructure>
- Terminology
 - AMI – Amazon Machine Image
 - Provides information required to launch an instance
 - ARN – Amazon Resource Name
 - Unique identifier for an AWS resource
- Services Provided
 - Networking and Content Delivery
 - VPC (Virtual Private Cloud) – a Virtual data centre in the cloud
 - Allows for Isolation of AWS Resources
 - CloudFront – CDN, used for caching objects
 - Example: websites or video content
 - DirectConnect – Dedicated network connection to AWS
 - Traffic will not go through internet and can expect more consistence speed
 - Route53 – DNS for AWS
 - Map names to numeric IP addresses
 - ELB – Load balancer (also compute)
 - Compute
 - EC2 – Virtual machine in the cloud
 - Give power to run any application and maintain server as if it were your own
 - ECS – Docker containers
 - Don't have to install, update or operate your own cluster
 - Good Use Cases: Microservices and Webapps
 - Lambda – Serverless compute resources
 - Integrates with other AWS services
 - Use Cases: IoT device processing, Dream, file processing

- Elastic Beanstalk – Webapps without infrastructure code
 - Handles auto-scaling and load balancing
 - ELB – Load balancing, distributes requests among instances
 - AutoScaling – scale up or down services based on traffic (add or remove instances)
- Storage
 - S3 – Object storage (highly available and durable)
 - Use Cases: User content, backup and recovery ...
 - Glacier – Archival Storage
 - Storage Gateway – Hybrid Storage
 - EFS – File system for EC2 instances (can be shared b/w instances)
- Migration
 - Snowball – Petabyte scale data solution (very fast)
 - A physical, secure device to transport petabytes of data from on premises cloud to AWS
 - DMS – Database migration <-> AWS, AWS <->AWS
 - Server Migration (SMS) Migrate on premise servers to AWS
- Database
 - RDS – Relational databases
 - AWS manages installation, management, and updates
 - DynamoDB – NoSQL (serverless)
 - ElastiCache – in-memory cache (can take workload away from DB itself)
 - Redshift – Data warehouse (PostgreSQL database engine on petabyte scale)
 - Aurora – Enterprise scale relational database
 - AWS Database Migration Service – Streamline migration between databases
- Management Tools
 - Cloudwatch – Monitoring and alerts in near-real time
 - Metrics like CPU utilization and alarms
 - Can create a custom metric
 - CloudFormation – Infrastructure as code
 - Can easily migrate your app to another region
 - Define architecture in a file (.json, .yaml, .txt, etc..)
 - CloudTrail – Audit all API calls and delivers logs to S3
 - OpsWorks -Chef configuration management
 - Config – Configuration resources rules
 - Chef recipe deployment (learn.chef.io)
 - Trusted Advisor – Reduce costs and improve security
 - Service Catalog – Organization catalogue

- Security, Identity, and Compliance
 - IAM – Access Control
 - Inspector – agents, security
 - Certificate Manager – SSL/TLS certificates
 - Directory Service – Directory store (Active Directory)
 - Cloud HSM – encryption keys on hardware
 - KMS – managed service for encryption keys
 - WAF – firewall for web apps
 - Shield – expanded DDOS protection
 - Compliance Reports (Artifact) – AWS documents
 - Cognito – Authenticate users and save preferences
- Analytics
 - Elastic Map Reduce (EMR) – Hadoop as a service
 - Elasticsearch - Elasticsearch as a service
 - Kinesis – Real time data streams collection and analysis
 - QuickSight – Data Visualization as a service
 - Data Pipeline – Process and move data
- Messaging
 - Simple Queue Service (SQS) – Queue in front of application for holding requests
 - application demand decoupling
 - Simple Notification Service (SNS) – Send messages to the public/subscribers
 - Mobile push notifications
 - Simple Email Service (SES) – Bulk delivery of email
- Developer Tools
 - CodeCommit – Managed Git source control
 - Code Build – Compiles source code, runs test
 - CodeDeploy – automated deployments on premise & AWS
 - Code Pipeline – CI/CD
- Business Productivity
 - WorkMail _ Managed email service
 - WorkDocs – Storage and sharing service
- Mobile
 - Mobile Hub – Build, test and monitor mobile apps
 - Device Farm – Test apps against real devices in the cloud
 - Mobile Analytics – App usage and revenue
 - Pinpoint – Targeted push notifications
 - API Gateway – Build, deploy, & manage APIs
- Artificial Intelligence

- Lex – Speech to text, natural language understanding
 - Polly – Text to speech
 - Rekognition – image recognition
 - Machine Learning – Apply complex algorithms, predictions
- Game Development
 - GameLift – Automated infrastructure deployment and Autoscaling for session based multiplayer games
- Internet of Things
 - IoT – Connect hardware devices to the cloud
- More services available and continue to be rolled out
- [Penetration Testing](#)
 - Needs to be approved by AWS beforehand
 - Third parties cannot be approved. You must make the request
- Command Line Interface [Reference Sheet](#)

Shared Responsibility Model



1. Introduction

- a. Defines what AWS is responsible for and what you as a customer are responsible for
- b. White Paper: [“AWS Security Best Practices”](#)
- c. White Paper: [“Overview of Security Processes”](#)

- d. Responsibilities will vary based on level of service management (IaaS vs PaaS vs SaaS)

2. AWS Responsibilities

- a. Decommissions storage devices as per industry standards
- b. Physical server level
- c. Customer instances have no access to raw disk devices, but instead are presented with virtualized disks
- d. Facility (data centre) security and protection
 - i. Fire detection and protection
 - ii. Climate and temperature
 - iii. Power
 - iv. Management (electrical, mechanical, and life support systems monitoring)
 - v. Physical access is monitored by security staff utilizing video surveillance, intrusion detection systems, and other electronic means
 - vi. Authorised staff must pass two-factor authentication a minimum of two times to access data center floors
 - vii. All visitors and contractors are escorted by authorised staff
- e. Network security:
 - i. Secure network architecture (E.g. Firewalls to control communication)
 - ii. Transmission protection (E.g. SSL)
 - iii. Amazon corporate segregation
 - iv. Fault-tolerant design (think regions, AZs, etc.)
 - v. DDOS protection
 - vi. Man-in-the-middle attacks protection
 - vii. EC2 instance can't send spoofed traffic
 - viii. Port scanning not allowed even if you own the environment → make a request to AWS before attempting to do penetration testing
 - ix. Packet sniffing by other tenants → cannot listen to one another's traffic even if on the same host → recommended to encrypt sensitive traffic

3. User Responsibilities

- a. Amazon Machine Images (AMIs)
- b. Operating Systems (patching and installing software)
- c. Applications
- d. Data in transit
- e. Data at rest
- f. Data stores
- g. Credentials (Console or programmatic access)
- h. Policies and configuration
- i. Multi-factor authentication
- j. Using trusted advisor to find vulnerabilities
- k. VPC
- l. Network Access Control List
- m. Security Groups
- n. Security logs with CloudTrail

Management & Security

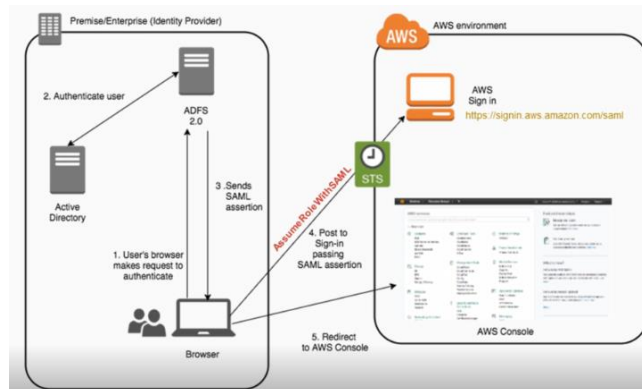
Identity Access Management (IAM)

4. Introduction

- a. Gives you **control over** who can **access** your **AWS** environment
- b. Authorization over resources by controlling what users can do
- c. Applies **Globally** to your AWS environment

5. Benefits

- a. Shared Access to your AWS account
- b. Central control of your account
- c. Granular API Level Permissions
 - i. Example: only allow use to see certain items in S3 or have read-only access
- d. Identity Federation
 - i. First verify their credentials with a 3rd party (i.e. Facebook, google)



- ii.
- e. Temporary credentials to users/applications
- f. Multifactor Authentication (MFA)
- g. PCI/DSS Compliance for card payments
- h. Eventually Consistent
- i. Free to use

6. IAM Components

- a. IAM lets you create individual users within your AWS account and give them each their own user name, password, and access keys
- b. Users (think of as people)
 - i. Programmatic access and console access creds
 - 1. Can use APIs to control the environment and see console
 - ii. Best Practice: Use IAM user instead of root credentials
 - iii. Can use pre-existing policy templates to assign permissions

- iv. No permissions by default
- c. Groups
 - i. Assign the same permissions to a set of users (users inherit permission of group they are in)
 - ii. Groups can only contain users, cannot be nested
 - iii. 100 groups per account
 - iv. Costs cannot be tracked by user, group or role (use cost allocation tags and view in Billing and Cost Management)
- d. Roles
 - i. Permissions that can be assumed by users or resources
 - ii. User can only be assigned to one role
 - iii. Recommended way of specifying what resources are allowed to do with other resources
 - 1. Example: Allow an EC2 instance to be able to write to a DynamoDB NoSQL database table, then create a role for EC2 and specify it can write to the table
 - iv. Best Practice: Never use creds on an instance (don't keep keys)
 - v. Users can assume a role to gain temporary access (e.g. Using FB credentials)
 - vi. Can be used to allow users to temporarily assume a role with least privilege access
- e. Identity Federation
 - i. An IAM role can be used to specify permissions for externally identified (federated) users
 - ii. Max 5000 IAM users per account. Identity federation enables unlimited temporary credentials
 - iii. Identified by your organization or a third-party identity provider
 - iv. Methods of federating users:
 - 1. Amazon Cognito (developer authenticated IDs, guest access, or public identity service provider)
 - 2. Public Identity Service Providers or OpenID Connect
 - 3. Identity provider software package that supports SAML 2.0
 - 4. Creating a custom identity broker application that authenticates users (e.g. with enterprise's LDAP or Active Directory service)
 - 5. AWS Directory Service for Active Directory and use this for enterprise AWS access
- f. Policies
 - i. Documents that specify a set of one or more permissions
 - ii. Written in JSON
 - iii. Two Types: User policies & Resource Policies

7. User-Based v Resource Based Policies

- a. IAM policies (resource-level) are attached to a user, group, or role and specify that actions that are permitted and the resource that can be accessed
- b. Resource-based policies are attached to a resource and only available for:
 - i. S3 (bucket policies and ACLs), Glacier (vault access policies), SNS topics, SQS queues, and KMS encryption keys

8. Best Practices

- a. Enable MFA on reduce root account
 - i. Virtual or hardware security tokens
 - ii. MFA for root account
 - iii. Create Admin users group and restrict root access
 - iv. MFA for admin users
- b. Grant least privilege
 - i. Default deny
 - ii. Identify required permissions
 - iii. Avoid assigning wildcard policies (*:*)
 - iv. Use policy templates
- c. Create individual users
- d. Manage permissions with groups
- e. IAM roles to share access and for EC2
 - i. Where possible, do not use security credentials
 - ii. Never share credentials
 - iii. Assign an EC2 instance a role instead of putting credentials in code
 - iv. Use cases:
 - 1. Cross account access
 - 2. Inter-account delegation
 - 3. Federated Users
 - v. Add external ID condition in policy for 3rd party users
- f. Restrict further with conditions
 - i. E.g. check MFA was used
 - 1. "Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : true } }
- g. Configure strong password policy
 - i. Password Expiration
 - ii. Password Strength
 - iii. Passwords are not reused
- h. Rotate security credentials
 - i. Credentials report to see aged or unused credentials
 - 1. i.e. someone leave the company, so their credentials should be deleted
 - ii. Grant users permissions to rotate their own credentials

9. Other Security measures

- a. Enable AWS Cloud Trail
 - i. Monitor and log API calls to services
- b. AWS Key Management Service (KMS)
- c. VPC security
 - i. Store data in private subnet with no direct access to the internet

10. More Information

- a. [Setting up credentials](#)

Trusted Advisor

1. *An online resource to help you reduce cost, increase performance, and improve security by optimising your AWS environment, Trusted Advisor provides real time guidance to help you provision your resources following AWS best practices. – AWS*
2. Types of optimisations:
 - a. Cost
 - b. Performance
 - c. Security
 - d. Fault Tolerance
 - e. Service Limits

CloudFormation

```
{
  "AWSTemplateFormatVersion": "2010-09-09",

  "Mappings": {
    "RegionMap": {
      "us-east-1": { "32": "ami-6411e20d", "64": "ami-7a11e213" },
      "us-west-1": { "32": "ami-c9c7978c", "64": "ami-cfc7978a" },
      "eu-west-1": { "32": "ami-37c2f643", "64": "ami-31c2f645" },
      "ap-southeast-1": { "32": "ami-66f28c34", "64": "ami-60f28c32" },
      "ap-northeast-1": { "32": "ami-9c03a89d", "64": "ami-a003a8a1" }
    }
  },

  "Resources": {
    "myEC2Instance": {
      "Type": "AWS::EC2::Instance",
      "Properties": {
        "ImageId": { "Fn::FindInMap": [ "RegionMap", { "Ref": "AWS::Region" }, "32"] },
        "InstanceType": "m1.small"
      }
    }
  }
}
```

1. Introduction

- a. Infrastructure as code (JSON or YAML templates)
- b. Version control capability

- c. Template describes all the AWS resources and CloudFormation takes care of provisioning and configuring
- d. Unlimited templates but limited to maximum of 200 stacks per account
 - i. Can request a higher limit

2. Template Sections

- a. **Format Version** template conforms to
- b. **Description** must always follow Format Version
- c. **Metadata** JSON objects and keys that provide additional info
- d. **Parameters** allow values to be passed at stack creation. Default parameter can be defined

```
"Parameters" : {
  "InstanceTypeParameter" : {
    "Type" : "String",
    "Default" : "t2.micro",
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"],
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro"
  }
}
```

- e. **Mappings** match keys to corresponding name value pairs
 - i. [Mappings Structure](#)
- f. **Conditions** define when a resource is created or a property is defined

```
"Parameters" : {
  "EnvType" : {
    "Description" : "Environment type.",
    "Default" : "test",
    "Type" : "String",
    "AllowedValues" : ["prod", "test"],
    "ConstraintDescription" : "must specify prod or test."
  }
},

"Conditions" : {
  "CreateProdResources" : {"Fn::Equals": [{"Ref" :
"EnvType"}, "prod"]}
},

"Resources" : {
  "EC2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "ImageId" : {"Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "AMI" ]}
    }
  },

  "MountPoint" : {
    "Type" : "AWS::EC2::VolumeAttachment",
    "Condition" : "CreateProdResources",
    "Properties" : {
      "InstanceId" : { "Ref" : "EC2Instance" },
      "VolumeId" : { "Ref" : "NewVolume" },
      "Device" : "/dev/sdh"
    }
  }
},
```

- g. **Resources** declares the resources to be included in the stack. It is the only mandatory section of a template
- h. **Outputs** declares output values that can be:
 - i. Imported to other stacks
 - ii. Returned to describe stack calls
 - iii. Displayed on the console

```

"Outputs" : {
  "BackupLoadBalancerDNSName" : {
    "Description": "The DNSName of the backup load balancer",
    "Value" : { "Fn::GetAtt" : [ "BackupLoadBalancer", "DNSName" ] },
    "Condition" : "CreateProdResources"
  },
  "InstanceID" : {
    "Description": "The Instance ID",
    "Value" : { "Ref" : "EC2Instance" }
  }
}

```

3. CloudFormer

- Creates an AWS CloudFormation template from existing AWS resources in your account
- You select resources from your account
- CloudFormation template created in an S3 bucket
- CloudFormer is itself a CloudFormation stack
- Does not support YAML (just use a converter)

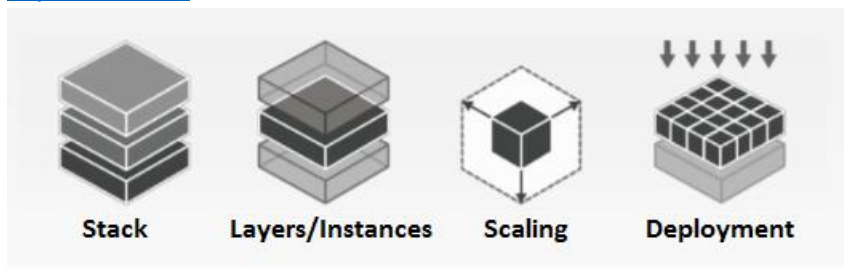
4. CloudFormation Designer

- Visual tool that provides a drag-and-drop interface for adding resources to templates
- Currently does not support YAML

5. Additional Information

- CloudFormation [FAQs](#)
- Intrinsic Function Reference [Guides](#)
- [Format Version Structure](#)
- API [Reference Sheet](#)
- cfn-helper scripts [documentation](#)
- [Template Snippets](#)
- [Template Anatomy](#)

[OpsWorks](#)



1. Introduction

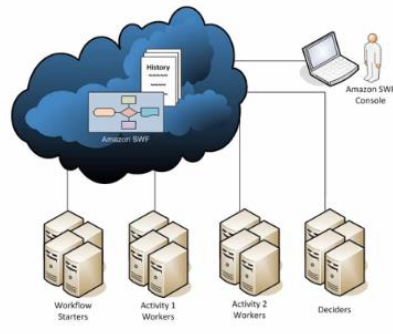
- A Configuration management platform
- Provides more control over infrastructure design and management than EB
- Infrastructure as code using Chef recipes for fine-grained control

- d. Consists of a CM model based upon Stacks, Layers, and Recipes
- 2. Stack
 - a. Top-level OpsWorks entity
 - b. Represents a set of instances and applications that you want to manage collectively
 - c. E.g. Web server stack may contain a load balancer, server instances, and a database
- 3. Layers, Instances, and Apps
 - a. Defines how to set up and configure instances and resources
 - b. Stacks must contain one or more layers
 - c. Layers must contain at least one instance
 - d. Instances can be a member of multiple layers
 - e. Apps represent code to run on your server
- 4. Scaling
 - a. 24/7 instances added to a layer can manually start, stop, or reboot the corresponding EC2 instances
 - b. Automatic Scaling
 - i. Time based instances based upon a schedule
 - ii. Load based instances based upon several load metrics (network traffic, CPU utilization, etc.)
 - c. Combination of all 3 types is an effective strategy
- 5. Deployment and Customization
 - a. App and associated infrastructure is deployed automatically
 - b. Chef recipes define infrastructure as code
 - i. Customization
 - ii. Redeployment
 - iii. Version control
 - iv. Code reuse

Simple Workflow Service ([SWF](#))

- 1. Introduction
 - a. Coordinates work across distributed application components
 - b. Helps implement complex business processes and application workflows
 - c. Long running execution
 - d. Enables complex interactions between different applications on different platforms, AWS and on-premise infrastructure, and different users
 - e. Maximum number of SWF domains per AWS account: 100
 - f. Workflow task can live up to 1 year
- 2. SWF Features

- a. Tasks executed with no duplicates
 - b. Routing and queueing tasks
 - c. Timeout and execution status
 - d. Workflows can have child workflows
 - e. User data input and execution results output
 - f. Guarantees delivery order of messages/tasks
3. SWF Components
- a. A **Workflow** is the control flow logic for the execution of tasks
 - b. A **Domain** contains a workflow or workflows
 - c. **Tasks** can be performed by executable code, a web service call or end user input. They can be performed in parallel or serially
 - d. **Actors** interact directly with SWF to coordinate tasks
4. SWF Actors
- a. **Actors** can be workflow starters, deciders or activity works
 - b. **Starters** can initiate execution of a workflow
 - c. **Deciders** implement workflow logic and notify SWF of changes during workflow execution
 - d. **Activity Workers** perform activity tasks of the workflow. Poll SWF to perform new tasks

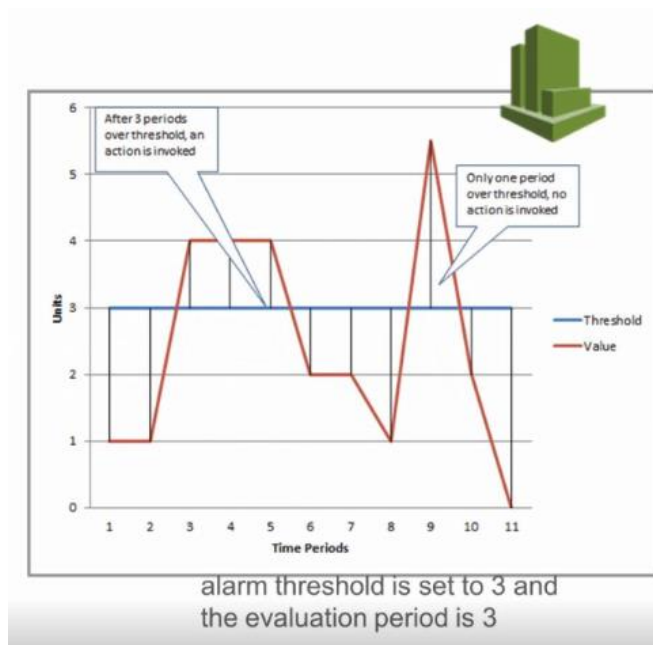


- e.
5. SWF Tasks
- a. Must be registered using either the console or RegisterActivityType action (API/CLI)
 - b. When scheduled you can specify a task list (queue)
 - c. Decision and Activity tasks have separate lists (queues)
 - d. Particular tasks can be assigned to particular activity workers through **task routing** if required
 - e. **Types of Tasks:**
 - i. **Activity Task** – tells an activity worker to perform its function, such as to check inventory or charge a credit card. Activity tasks contain all the information that the activity worker needs to perform its function
 - ii. **Lambda Task** – similar to an Activity task, but executes a Lambda function instead of a traditional SWF activity

- iii. **Decision Task** – tells a decider that the state of the workflow execution has changed so that the decider can determine the next activity that needs to be performed. Contains the current workflow history. Can be performed by humans
- 6. SWF Implementation
 - a. Application Communication to SWF
 - i. SDKs
 - ii. SWF API HTTP POST calls
 - iii. Flow Framework (Java or Ruby)
 - b. SWF setup can also be achieved using console or CLI
- 7. Additional information
 - a. SWF [FAQ](#)

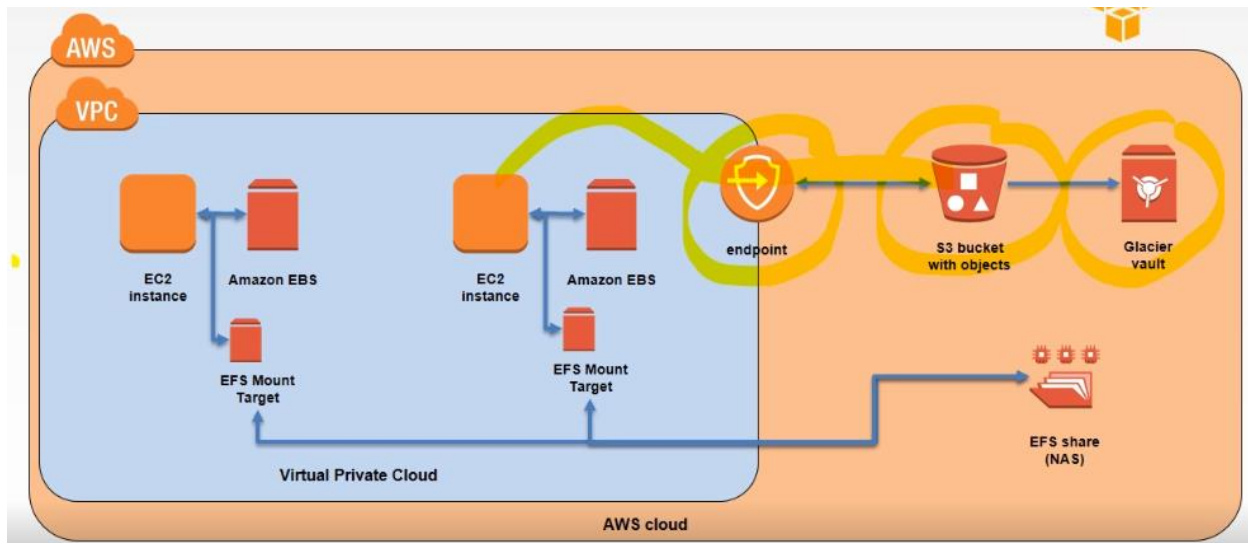
CloudWatch

- 1. CloudWatch Metrics
 - a. Example (not available in all regions)
 - i. Billing
 - ii. DynamoDB
 - iii. EC2, EBS
 - iv. Elastic Beanstalk
 - v. OpsWorks
 - vi. Kinesis Streams
 - vii. [Full List](#)
- 2. CloudWatch Statistics
 - a. Average, Max, Min, etc.
 - b. CLI – get-metric-statistics, API – GetMetricStatistics
 - i. *Maximum number of data points that can be queried: 50,850*
 - ii. *Maximum number of data points returned from a single request: 1,440*
 - c. View the console and create dashboards
- 3. Alarms
 - a. Billing alarms as well as resource alarms
 - b. Integrates with SNS
 - c. Three states: OK, ALARM & INSUFFICIENT_DATA
 - d. If a metric is above the alarm threshold for the number of time periods defined by the evaluation period, an alarm is invoked



- e.
- 4. CloudWatch Logs
 - a. Monitor, store, and access your log files from EC2 instance, CloudTrail, or other sources
 - b. Real time monitoring of log information
 - c. Log Streams – sequence of log events from a source
 - d. Log Groups – streams the same retention, monitoring, and access control settings
 - e. Metric filters – define how information is extracted to create data points
 - f. Retention settings – how long log events are kept in CloudWatch logs
- 5. CloudWatch Events
 - a. Events:
 - i. Occur when resources change state. For example:
 - 1. EC2 state change
 - 2. AutoScaling instance launch
 - ii. CloudTrail integration e.g.
 - 1. API calls
 - 2. Log into console
 - b. Rules:
 - i. Match incoming events and route them to one or more targets for processing
 - c. Targets:
 - i. Can invoke AWS Lambda functions, Amazon SNS topics, Amazon SQS queues, Amazon Kinesis Streams, or built-in targets

Storage



Simple Storage Service (S3)

1. Terminology

a. Bucket

- Uniquely named container for object storage – [Naming Requirements](#)
- Total volume of data and number of objects are unlimited
- 100 buckets per account (can be increased by request to AWS)

b. Objects

- Entities (data and metadata) stored in a bucket
- 0 bytes to a maximum of 5 TB
- Largest object upload is 5 GB
- Multi-part upload for 100 MB to 5 TB

1. [Multi-part upload documentation](#)

c. Keys

- Identifier of object in bucket. Object uniquely identified by bucket, key and version ID.

d. Error Responses

- [List of S3 Error Codes](#)

2. Web Store

- Eventual Consistency - Index updated after data changes synchronized across multiple AZ

- S3 is a Web store not a file system

- Read-after-write consistency with new objects synchronized across multiple AZ before indexed and success returned

- i. Upload new object ⇒ Synchronized ⇒ S3 Index Updated ⇒ Success Returned
- c. Updates and deletes are not RaW/delete consistent, they are eventually consistent
 - i. Update/Delete Object ⇒ Success Returned ⇒ Synchronized ⇒ S3 Index Updated

3. [Cross-Origin Resource Sharing \(CORS\)](#)

- a. Defines a way for client web applications that are loaded in one domain to interact with resources in a different domain
 - i. Example, request page that is hosted in a different AWS region

4. S3 security

- a. S3 is secure by default but can be modified by:
 - i. **IAM** roles, users, and groups (fine grained control)
 - ii. [Bucket policies](#) applied at the bucket level (fine grained control)
 - 1. Resources – ARN to identify the resource (bucket/object)
 - 2. Actions – Action to allow or deny
 - 3. Effect – Allow or deny?
 - 4. Principle – Account or user allowed to access the actions and resources in the statement
 - 5. [Bucket Policy Examples](#)
 - iii. **Access Control Lists (ACL)** applied at the bucket and/or object level
- b. [Server-Side Encryption](#)
 - i. Uses 256-bit Advance Encryption Standard (AES-256)
 - ii. REST API header for encryption: x-amz-server-side-encryption
 - iii. [Using Server-Side Encryption](#) on S3

5. S3 Performance

- a. Key prefixes affect [performance](#)
 - i. Partitions are based upon key prefixes
 - ii. Speed can be increased by ensuring key prefixes do not all start with the same character
 - 1. Numbers at start
 - 2. Reverse object names
 - 3. Random names
 - 4. Hash values
 - iii.

Slow	Faster
folder0/	0folder/
folder1/	1folder/
folder2/	2folder/

folder3/	3folder/
----------	----------

6. [S3 Storage Classes](#)

- a. Standard
 - i. 99.999999999999% durability (replication)
 - ii. 99.99% availability
 - iii. Supports SSL encryption of data in transit and at rest
 - iv. Lifecycle management - old objects can be deleted or archived
- b. Standard - Infrequent Access
 - i. Same features as S3 standard
 - ii. Lower per GB storage price but a per GB retrieval fee
 - iii. 99.9999999999% durability
 - iv. 99.9% availability
- c. Reduced Redundancy
 - i. Lower cost than S3 standard
 - ii. noncritical, reproducible data
 - iii. 99.99% durability
 - iv. 99.99% availability
- d. Archive to Glacier
 - i. Low cost storage
 - ii. Retrieval time of several hours
 - iii. Vault Lock feature enforces compliance via a lockable policy

7. [Multipart Upload](#)

- a. Recommended by AWS to upload in 100MB chunks
- b. Improve throughput with parallelized upload
- c. Upload can be stopped and started again
- d. Can upload data as it is being produced (before fully written)
- e. [Uploading Objects Using Multipart Upload API](#)

8. Lifecycle Management

- a. Object deletion after expiry time
- b. Object Archiving to Glacier after expiry time
- c. Can be restored from Glacier back to S3

9. [Versioning](#)

- a. Preserves copies of objects inside a bucket
- b. Individual objects can be restored to previous versions
- c. Deleted objects can be recovered

10. [Cross Region Replication](#)

- a. Reduced latency for end users
- b. Both source and destination buckets need versioning enabled if using versioning

- c. ACL details updated
 - i. S3 enabled encryption replicated
 - ii. KMS encryption not replicated
- d. Need to copy existing objects to new region

11. [Transfer Acceleration](#)

- a. Enables fast and secure transfer of files between your S3 bucket and your users over long distances
- b. Uses Edge Locations (CloudFront)
- c. Use cases: Global users upload to the same S3 bucket, regular GB or TB data transfer over continents, and underutilization of Internet bandwidth for S3 uploads

12. Events

- a. Can trigger an action when an object is Created/Modified/Deleted from S3
 - i. Example: run a Lambda function

13. Additional Information

- a. S3 [FAQs](#)
- b. S3 [Error Responses](#)
- c. S3 Events to [Push SNS Notifications](#)
- d. [Website Hosting](#) with S3

Elastic File System ([EFS](#))

File	Object	Block
<ul style="list-style-type: none"> EFS 	<ul style="list-style-type: none"> S3 Glacier 	<ul style="list-style-type: none"> EBS Instance Store

1. Introduction

- a. Simple, scalable file storage for use with EC2 instances
- b. Network Attached Storage (NAS)
- c. Can be accessed by multiple EC2 instances at the same time

2. [Advantages of EFS](#)

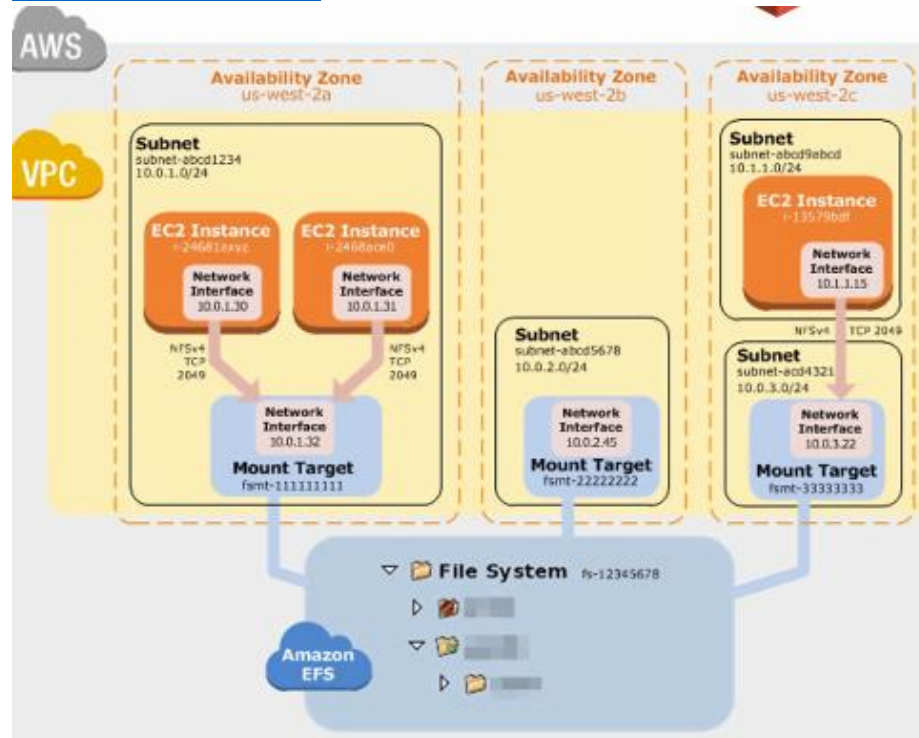
- a. Fully managed service
- b. File system grow and shrinks automatically up to petabyte sizes
- c. Pay only for the storage space you use, with no minimum fee
- d. Throughput scales automatically to ensure consistent low latency
- e. Can support thousands of connections
- f. Multi-AZ replication

3. Disadvantages of EFS

- a. Not available in all regions (subject to change)
- b. No cross-region capability (neither does EBS)
- c. More complicated to provision compared to S3 and EBS

4. Mount Targets

- a. Required to access EFS from a VPC
- b. VPC NFS endpoint
- c. Has an IP address and a DNS name you can use in the Linux mount command
- d. Can be mounted by multiple EC2 instances Can be in different subnet to instance but cannot be in a different AZ
- e. [Creating Mount Targets](#)



f.

5. Accessing File System from EC2

- a. Requires NFS client (standard on current Linux distributions)
- b. Mount file system using the Linux mount command similar to EBS and instance store
- c. File system DNS name (easiest) or mount point DNS name can be used to mount EFS on EC2

6. EFS Security

- a. IAM user permissions for create, update and delete
- b. EC2 security groups can be set as inbound rules for EFS and vice versa
- c. NACLs can be used to control traffic
- d. Linux/Unix file root-only permissions by default (CHOWN, CHMOD)

Snowball

1. Introduction
 - a. Physical device that allows for transfer of large amounts of data (TB/PB) in and out of AWS
 - b. No network infrastructure charges
 - c. Avoid using high-cost/low-speed internet connection for transfer
2. Current version of Snowball
 - a. Petabyte-scale data transportation solution
 - b. Uses secure appliances for transferring data in and out of AWS (S3)
 - c. Secure: tamper-resistant, 256-bit encryption, industry-standard Trusted Platform Module (TPM)
 - d. Fast data transfer (transfer up to 80 terabytes)
 - e. Secure erasure
 - f. Use cases: cloud migration, disaster recover, content distribution
 - g. Create job with AWS and they will send the device
3. Snowball Edge
 - a. 100TB data transfer device
 - b. On-board storage & compute capabilities
 - c. Transfer data in and out of AWS
 - d. Temporary storage tier
 - e. Supports local workloads (lambdas)
 - f. Use Cases: Remote locations, IOT, manufacturing
4. Snowmobile
 - a. Exabyte-scale data transfer service
 - b. Max 100PB
 - c. Dedicated security personnel
 - d. GPS tracking
 - e. 24/7 video surveillance

Storage Gateway

1. Introduction
 - a. Hybrid Cloud Storage Service
 - b. Virtual appliance
 - c. Low-latency performance by caching most recently used data
 - d. Optimises data transfer to AWS storage
 - e. Use cases: hybrid workloads, backup and restore, disaster recover

2. Storage Gateway Types

- a. File Gateway → Store and retrieve objects from S3 (Virtual On-prem File Server)
- b. Volume Gateway → Block storage to you On-prem apps that can be tied into S3
- c. Two existing Modes:
 - i. Cached mode: Whole dataset is stored on S3 and the frequently used data is cached on premise
 - ii. Stored mode: Whole dataset is stored on premise and is asynchronously transferred to S3
- d. Tape Gateway → Provides your backup app with a virtual tape library (VTL) interface:
 - i. A virtual media changer, virtual drives, and virtual tapes
 - ii. The data is stored on S3

Database

Relational Database Service ([RDS](#))

- 1. Introduction
 - a. Managed relational databases as a server
 - b. Amazon Aurora, MySQL, MariaDB, Oracle, Microsoft SQL Server, and PostgreSQL
 - c. Handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair
- 2. RDS Backup
 - a. User initiated DB Snapshots of instance
 - b. Automated DB backups to S3
 - c. Encryption of database and snapshots at rest available
- 3. Multi Availability Zone
 - a. Recommended for production applications
 - b. Application should also be located in multiple AZs
 - c. Available for all database types
- 4. Failover
 - a. In the event of a failover condition:
 - i. Standby instance promoted to master
 - ii. CNAME DNS record is changed to point to the standby instance
 - iii. New standby instance created to replace failed instance
- 5. Read Replicas
 - a. Supported for Aurora, PostgreSQL, MySQL, and MariaDB
 - b. Multiple read replicas (up to 15 for Aurora)
 - c. Cannot be put behind AWS ELB

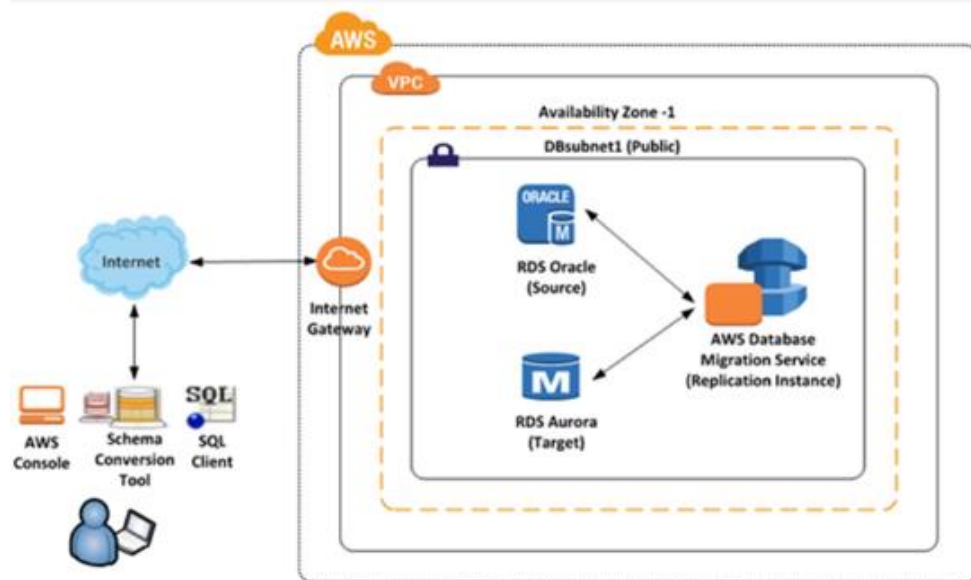
- i. Can get around using Route 53 routing or HaProxy

Database Migration Service ([DMS](#))

1. Introduction

- a. Migrate data from:
 - i. Oracle
 - ii. Microsoft SQL Server
 - iii. MySQL, Amazon Aurora and MariaDB
 - iv. PostgreSQL
 - v. MongoDB
 - vi. SAP Adaptive Server Enterprise (ASE)
- b. Migrate data to all of the above plus:
 - i. Amazon Redshift
 - ii. Amazon S3
 - iii. Amazon DynamoDB

2. How it works:



- a.
- 3. Steps:
 - a. Create your databases to transfer to/from
 - b. Create a data replication instance
 - i. Use the same security group as for the databases
 - c. Create endpoints for data replication instance
 - i. Create one for source and one for target
 - ii. You can run a simple test from the console to check that your replication instance can ping to your databases

DynamoDB



1. Introduction

- a. A Managed NoSQL database service
- b. Support storing, querying, and updating documents
- c. Data is stored in JSON-like format
- d. Consists of Tables, Items, and Attributes
- e. Secondary Indexes
- f. 256 tables per AWS account per region

2. Advantages of DynamoDB

- a. Scales seamlessly including through API & CLI calls (provisioned throughput)
- b. Query on any attributes (column) using secondary indexes
- c. Supports cross-region replication
- d. Schema-less
- e. Supports strong consistency on reads
- f. Atomic counters
- g. Downloadable version available that you can run locally

3. SQL vs NoSQL

- a. SQL optimized for storage, NoSQL optimized for speed
- b. SQL data is normalised (relational). NoSQL is denormalised (hierarchical)
- c. SQL data has joins and rules
- d. SQL language is rich, NoSQL simple queries and scans
- e. SQL scales vertically, NoSQL scales horizontally
- f. SQL suitable for Online Analytical Processing (OLAP), NoSQL suitable for Online Transaction Processing (OLTP)

4. Tables, Items, Attributes & Primary Keys

- a. Tables are a collection of data (e.g. customers)
- b. Tables contain Items (similar to rows or records)
- c. Items have one or more Attributes (eg customerID, firstName, etc.)
- d. Tables must define a Primary key as a unique item identifier (e.g. customerID)
- e. Primary keys can be Partition (hash) or Partition and Sort (hash & range)

Books

```
"ISBN": { "S": "222-2222222222" },
"Title": { "S": "Book 102 Title" },
"Authors": { "SS": [ "Author 1", "Author 2" ] },
"Price": { "N": "20" },
"Dimensions": { "S": "8.5 x 11.0 x 0.8" },
"PageCount": { "N": "600" },
"Fiction": { "BOOL": true }
```

```
"ISBN": { "S": "333-3333333333" },
"Title": { "S": "Book 103 Title" },
"Authors": { "SS": [ "Author 1", "Author 2", "Author 3" ] },
"Price": { "N": "200" },
"Dimensions": { "S": "8.5 x 11.0 x 1.5" },
"PageCount": { "N": "700" },
"Fiction": { "BOOL": false }
```

f.

5. Secondary Indexes

- a. Let you query the data using an alternate key in addition to primary key
- b. You can define up to 5 global secondary indexes and 5 local secondary indexes per table for a total of 10
- c. You specify which attributes will be copied or projected
 - i. Not essential but can provide improved flexibility and speed of querying
- d. All scalar data types (Number, String, Binary, and Boolean) can be used for sort key element of the local secondary index key. Set, list, and map types cannot be indexed
- e. Global and Local [Secondary Indexes](#)
 - i. Local Secondary Indexes
 1. Same partition (hash) key as the table
 2. Different sort (range) key as the table
 3. Can only be created at the same time the table is created
 - ii. Global Secondary Indexes
 1. Partition and sort keys can be different from the table
 2. Can be created when table is created or added later

6. Query and Scan

- a. Query
 - i. Generally, more efficient than scan operations
 - ii. Searches for a specific range of keys that satisfy a given set of key conditions
 1. Specify a Partition Key + Search Value
 2. Optionally you can specify a Sort key (use comparisons)
 - iii. Returns all attributes of the matched values (use ProjectionsExpression of subset)
 - iv. Can also filter the query results using filters on non-key attributes (FilterExpression)

- v. Results can be filtered
- vi. Eventually consistent, but you can request a strongly consistent read
- vii. Maximum size returned is 1MB + LastEvaluatedKey
- viii. Results are by default sorted in ascending order (REMEMBER FOR EXAM)
 - 1. To get results sorted in descending order by the sort key → ScanIndexForward=False

b. Scan

- i. Scans the entire table or secondary index then filters results
 - 1. Returns max 1MB of data + LastEvaluatedKey to continue search in new operation
- ii. Eventually consistent, but ConsistentRead set to true will return results at the time the scan started. This will consume twice the read capacity units
- iii. Larger the dataset, slower the response
- iv. Read Spikes:
 - 1. Reduce Page Size (Limit = # items < 1MB)
 - 2. Isolate Scan Operations (prod table + shadow table)
- v. Can use parallel scans. *TotalSegments* = number of workers that perform parallel scans
 - 1. May throttle your table, requiring high throughput

c. GetItem and BatchGetItem are also efficient APIs

7. Additional DynamoDB Operations

- a. CreateTable/UpdateTable/DeleteTable (max. 10 concurrently)
- b. If adding LSI/GSI to a table max 5 concurrently
- c. BatchGetItem max 100 items, max 16MB
- d. BatchWriteItem max 25 PutItem or DeleteItem requests, max 16MB
 - i. Responsible for inserting, replacing, and deleting multiple items across several tables in one command but not as one transaction
- e. DescribeLimits for a region to obtain the current account limits on the provisioned capacity
 - i. Call DescribeLimits periodically if used programmatically → throttling errors if called > 1 per min

8. Atomic Counters vs. Conditional Updates

a. Atomic Counters

- i. UpdateItem operation increments or decrements the value of an attribute without interfering with other write requests
- ii. Good for non-critical counter applications like site visits counter
- iii. Writes are applied in the order they are received
- iv. Counter will increment each time you call UpdateItem whether or not call was successful

b. Conditional Update

- i. Use ReturnValues parameter of UpdateItem if you want to get the item attributes as they appeared either before or after they were updated
- ii. Good for critical applications e.g. finance, ERP
- iii. Can use ConditionalWrites to only write if the item has not been changed since it was last read

9. Optimistic Locking

- a. A strategy to ensure that the client-side item that you are updating (or deleting) is the same as the item in DynamoDB
- b. Protects your DB writes from being overwritten by the writes of others and vice-versa
- c. With optimistic locking, each item has an attribute that acts as a version number and you can only update an item if the server-side version number is the same
- d. If you have a stale version number you simply try again by retrieving the item and reattempting to update it

10. Provisioned Throughput

- a. Provisioned throughput specified separately for tables and its indexes
- b. Customers can purchase *reserved capacity*
- c. Read Capacity units
 - i. One read capacity unit represents one strongly consistent read per second, or two eventually consistent reads per second, for items up to 4KB in size
- d. Write Capacity units
 - i. One write capacity unit represents one write per second for items up to 1KB in size
 - ii. Secondary indexes require additional capacity units i.e. one for writing to the table and another for writing to the index
- e. Examples:

5KB item size, 50 required eventually consistent read/second = $\lceil (5 / 4) \rceil \times (50 / 2) = 2 \times 25 = 50$

Explanation: 5KB requires two read units (4KB), 2 eventually consistent reads per read capacity unit

2KB item size, 50 required strongly consistent read/second = $\lceil (2 / 4) \rceil \times (50 / 1) = 1 \times 50 = 50$

Explanation: 2KB requires one read unit, 1 strongly consistent read per read capacity unit

10.5KB item size, 50 required writes/second = $\lceil (10.5 / 1) \rceil \times 50 = 11 \times 50 = 550$

Explanation: 10.5KB requires 11 write units

11. DynamoDB Streams

- a. Ordered flow of information about changes to items in an Amazon DynamoDB table

- b. Records are generated in near real-time
- c. Stream Record created whenever an application creates, updates, or deletes items in the table. Assigned a sequence number to be unique
- d. Stream Records are organized into groups (shards)
- e. The stream records within a shard are removed automatically after 24 hours
- f. Endpoint: streams.dynamodb.<region>.amazonaws.com

12. DynamoDB Triggers

- a. Triggered by item-level updates on DynamoDB table
- b. Can be used to invoke a Lambda function
- c. Associate an AWS Lambda function to the stream on a DynamoDB table
- d. Example usages: SNS notification, record in S3 bucket

13. Additional Information

- a. DynamoDB [FAQ](#)
- b. API [Reference List](#)
- c. [Working with Items](#)
- d. [Supported Data Types](#)



1. Introduction

- a. Managed in-memory cache service
- b. Key value stores
- c. Provides ultra-fast (sub-millisecond latency) access to cached data
- d. Redis and Memcached data store options
- e. Reduces load on database
- f. Multi-AZ capability
- g. Use:
 - i. High request rates to DB required
 - ii. Low volume of regularly accessed data
 - iii. Low latency

2. Memcached

- a. Free & open source project, high-performance, distributed memory object caching system
- b. Simple data types. Data structure is a string or object up to 1MB
- c. Max data volume 4.7 TiB

- d. No persistence. Lost data cannot be recovered (can be refetched from backend database although with latency penalty)
- e. Simple scaling by adding nodes. Data is distributed across nodes
- f. Suitable for:
 - i. Simple data structures
 - ii. Large nodes with multiple cores or threads
 - iii. Scaling in/out by adding/removing nodes according to demand (elasticity)
 - iv. Partitioning data across multiple shards
 - v. Database caching

3. Redis

- a. Free & open source project, in-memory data structure store
- b. Supports data structures such as strings, hashes list, sets, sorted sets with range queries, bitmaps, hyperlogs and geospatial indexes with radius queries
- c. Key/value size up to 512MB. Max data volume 3.5 TiB
- d. Persistence. Lost data can be recovered. Read replicas.
- e. Very large command set available
- f. Notifications from the Redis PUB/SUB channel
- g. Use over Memcached for:
 - i. Advanced data types
 - ii. Auto sorting of data
 - iii. Pub/sub capabilities
 - iv. High availability and failover
 - v. Persistence

4. Updating Cache

- a. Database triggers (e.g. DynamoDB, mysql.lambda_async procedure, MongoDB etc.) can be used to update ElastiCache using a Lambda function or EC2 instance
- b. Application can be used to update ElastiCache

5. Caching Strategies

- a. Lazy Loading
 - i. Loads data into the cache on a cache miss
 - ii. Requires TTL for stale data
 - iii. Requires 3 trips on cache miss
- b. Write Through
 - i. Application or trigger event updates cache when data written to the database
 - ii. Requires TTL for stale data
 - iii. Caches infrequent accessed data
- c. Adding TTL
 - i. Memcached set command expire parameter (seconds)
 - ii. Redis set command

1. EX seconds – Set the specified expire time, in seconds
2. PX milliseconds – Set the specified expire time, in milliseconds

Compute

Elastic Cloud Compute ([EC2](#))

EC2 Classic	EC2 VPC
EIP is disassociated when instance stopped	EIP remains associated when instance stopped
Unlimited number of security groups	Up to 5 security groups
Can't change the SGs of a running instance	Can change the SGs of a running instance
SG rules for inbound traffic only	SG rules for inbound and outbound traffic.
Instance can access the Internet by default	Requires IGW and route to IGW
DNS hostnames are enabled by default	DNS hostnames are disabled in default VPC
Runs on shared hardware only	Shared hardware or single-tenant hardware

1. Introduction

- a. Introduced in 2006, instances run in a single, flat (classic) network that you share with other customers
- b. VPC service introduced in 2009. Instances run in VPC that is logically isolated to your AWS account
- c. All new AWS account must have EC2 instances launched inside a VPC
 - i. New accounts come with a default VPC
- d. Old accounts still allow EC2 instances to be launch outside a VPC but only in previously used regions

2. [Purchasing Options](#)

- a. On-Demand
 - i. Pay for compute services by the hour with no long-term commitments or upfront payments
- b. Reserved Instances
 - i. Reserve an instance for a period of time
 - ii. Significant discount (up to 75%) compared to on-demand instance pricing
- c. Scheduled Instances
- d. Spot Instances
 - i. Purchase compute capacity with no upfront commitment and at hourly rates. Usually cheaper than on-demand prices

- ii. Spot instances are not always available and will fluctuate in cost/availability
- e. Dedicated Hosts
- f. Dedicated Instances

3. [Instance Types](#) Overview

- a. General Purpose (T2, T3, M4, M5)
 - i. Small and midsize databases, data processing tasks that require additional memory, caching fleets, running backend servers for SAP, MS SharePoint, cluster computing, and other enterprise applications
- b. Compute Optimised (C5, C4)
 - i. High performance front-end fleets, web-services, batch processing, distributed analytics, high performance science and engineering computing applications, ad serving, MMO gaming, and video encoding
- c. Memory Optimised (X1e, X1, R5, R4, z1d)
 - i. High performance databases, distributed memory caches, in-memory analytics, genome assembly and analytics, larger deployments of SAP, MS SharePoint, other enterprise applications
- d. Accelerated Computing/GPU (P3, P2, G3, F1)
 - i. 3D application streaming, Machine Learning, video encoding, FPGAs, other server-side graphics or GPU compatible workloads
- e. Storage Optimised (H1, I3, D2)
 - i. NoSQL databases such as Cassandra or MongoDB, scale out transactional databases, data warehousing, Hadoop, and other cluster file systems
 - ii. Massively parallel processing data warehousing, MapReduce and Hadoop distributed computing, distributed file systems, network file systems, log or data-processing applications

4. Instance Types Detail

- a. T2/T3 Burstable Performance Instances
 - i. Baseline performance and ability to burst are governed by CPU credits
 - ii. Credits are built up and stored for up to 24 hours while instance is operating below the baseline performance
 - iii. Credits are used to burst above baseline capacity when needed
 - iv. Consider upgrading if instance does not maintain positive CPU credit balance (use CloudWatch to monitor balance)

5. [EC2 Storage Options](#)

- a. Elastic Block Store ([EBS](#)) – Preferred
 - i. Replicated within an Availability Zone
 - ii. EBS optimised instances provide dedicated throughput between EC2 and EBS volume
 - iii. EBS volumes attached to the instance at launch are **deleted** when the EC2 instance is terminated

- iv. EBS volumes attached to a running instance are **not deleted** when the instance is terminated but are detached with data intact
- v. Type of EBS storage:
 - 1. General Purpose SSD (gp2) – Default Choice
 - 2. Provisioned IOPS SSD (io1)
 - a. Consistent, low-latency performance
 - b. I/O intensive applications use as large relational or NoSQL databases
 - 3. Throughput Optimized HDD (st1)
 - a. Low cost HDD
 - b. Frequently accessed, throughput intensive apps
 - c. Data warehouses, big data, log processing
 - d. Cannot be a boot volume
 - 4. Cold HDD (sc1)
 - a. Lowest cost HDD
 - b. Less frequently accessed workload
 - c. Cannot be a boot volume
 - 5. Magnetic – Lowest cost per gigabyte
- vi. EBS Snapshots
 - 1. Point in time backup of EBS volume to S3
 - 2. Incremental backups
 - 3. Can be copied to other regions or AWS accounts
- vii. EBS Encryption
 - 1. AWS KMS master keys or Customer Master Key (CMK)
 - 2. Data stored at rest (including snapshots) as well as data in transit b/w EBS and EC2

b. Instance Store

- i. Physically attached to host server
- ii. Data **Not Lost** when OS is rebooted
- iii. Data **Lost** when:
 - 1. Underlying drive fails
 - 2. Instance is stopped
 - 3. Instance is terminated
- iv. Do not rely on for valuable, long-term data
- v. Cannot detach and attach another instance (physical connection)

6. EC2 Metadata

- a. Data about your instance such as:
 - i. Local IP, Instance ID, AMI ID, and Availability Zone
- b. Access **from within your instance** at: <http://169.254.169.254/latest/meta-data>
- c. Useful for scripts from within the instance
- d. You are not billed for HTTP requests used to retrieve instance metadata and user data

7. Amazon Machine Images ([AMI](#))

- a. Provides information required to launch an instance
 - i. Template for the root volume for the instance
 - 1. E.g. An operating system, application server, and applications
 - ii. Launch permissions that control which AWS accounts can use the AMI to launch instances
 - iii. A block device mapping that specifies the volumes to attach to the instance when it is launched
 - iv. [Components of an AMI](#)
- b. [Creating a Custom AMI](#)
- c. [Copying an AMI](#)
 - i. AMI are attached to a region
 - 1. To make them available in other regions you must copy them over
 - ii. AMIs can be shared
 - 1. Public AMIs
 - 2. Share with specific AWS users
 - a. Need user's ID

8. Cluster Networking

- a. [Enhanced Networking](#)
 - i. Uses single root I/O virtualization (SR-IOV)
 - ii. Provides higher I/O performance and lower CPU utilization at no additional charge
 - iii. Supported in R4, X1, M4, C4, C3, I2, G3, and D2 instances (may change which updates)
- b. EBS optimised instances
 - i. Designed to deliver the provisioned IOPS performance 99.9% of the time
- c. [Placement Groups](#)
 - i. Provides low network latency, high network throughput
 - ii. Available for instances that support Enhanced networking
 - iii. Can't span multiple AZs
 - iv. Can span peered VPC but will not get full bandwidth
 - v. Instances must be added at launch only
 - vi. Can't merge placement groups

9. More information

- a. Mask instance failures using [Elastic IPs](#)
- b. Troubleshooting [Connection Issues](#)
- c. API [Reference Sheets](#)
- d. Launch AMIs from the CLI using [run-instances](#)

EC2 Container Service ([ECS](#))



1. Introduction

- Use Docker containers to define and launch EC2 instances
- Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same regardless of the environment it is running in. – Docker*
- Can deploy docker instances to pull code instead of running scripts to pull git repository (don't need to reboot or resolve dependencies on your instance)

[Elastic Beanstalk](#)

1. Introduction

- Automated deployment and scaling of web applications
- Supports the following languages and development stacks
 - Apache Tomcat* for *Java* applications
 - Microsoft IIS* 7.5, 8.0, and 8.5 for *.NET* applications
 - Apache HTTP Server* for *PHP* and *Python* applications
 - Nginx* or *Apache HTTP Server* for *Node.js* applications
 - Passenger* or *Puma* for *Ruby* applications
 - Java SE
 - Go
 - Docker
- Simply upload your code and Beanstalk handles the deployment, capacity-provisioning, load balancing, auto-scaling, and application health monitoring
- You retain access and full control over the underlying AWS resources
- No additional charge. Just pay for the resources used

2. Additional Information

- Elastic Beanstalk [FAQs](#)

Elastic Load Balancer ([ELB](#))



1. Introduction

- a. Automatically distributes traffic across multiple EC2 instances and availability zones
- b. Stops serving traffic to unhealthy instances
- c. Integrated with AutoScaling
- d. Can act as internal load balancer within a VPC
- e. Store SSL/TLS certificates -> reduce compute needs of instances
- f. ELBs have their own DNS name (No IP given!)

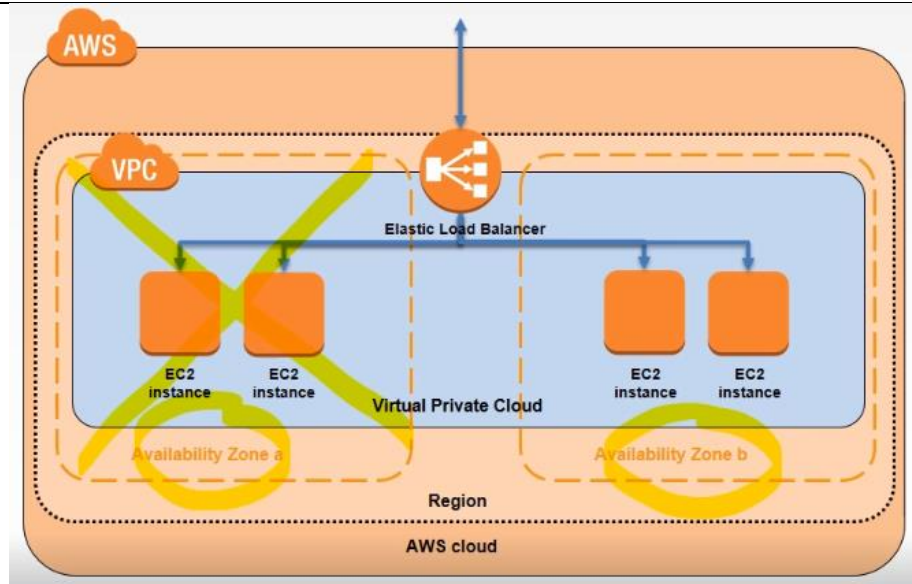
2. Types of Load Balancers

- a. Application Load Balancer
 - i. Makes routing decisions at the application layer
 - ii. Supports path-based routing
 - iii. Can route requests to one or more ports on each EC2 instance or container
 - iv. Use cases: apps that need advanced routing capabilities, container-based architectures, and microservices
- b. Classic Load Balancer
 - i. Routes traffic based on either application or network level information
 - ii. Use cases: simple load balancing of traffic across multiple EC2 instances

3. Session State

- a. Disabled
 - i. Each time a user refreshes their page interacts with the application, the load balancer could send the request to another EC2 instance with no tracking of the session
- b. Session Stickiness
 - i. Route request to the same target in the target group
 - ii. Clients must support cookies
 - iii. Types:
 - 1. Load Balancer Cookie Stickiness (duration based)
 - 2. Application Generated Cookie Stickiness (app controls the duration)
 - iv. Session stickiness may break balancing if many hosts are stuck to few instances

Networking



Virtual Private Cloud (VPC)



1. Introduction

- a. Within a region, a VPC is a virtual network within a region which can span multiple availability zones
- b. Limit of 5 per region by default

2. Quick Review of TCP/IP Subnet Addressing

a. Private Network Ranges

- i. Private address ranges are used within your private network as opposed to public IP addresses which are visible to the wider internet
- ii. Class A Private Address = 10.0.0.0/8 (starts with 10)
- iii. Class B Private Address = 172.16.0.0/12 (172.16.x.x to 172.32.x.x)
- iv. Class C Private Address = 192.168.0.0/16 (starts with 192.168)

b. Subnet Mask

- i. Defines the IP range of a network
- ii. Amazon reserves the first 4 IP addresses and the last 1 IP address of every subnet for IP networking purposes
- iii. Examples: IP 192.168.1.0 with Subnet Mask 255.255.255.0
 1. 11000000.10101000.00000001.00000000 (IP Address)
 2. 11111111.11111111.11111111.00000000 (Subnet Mask)
 3. -----
 4. 11000000.10101000.00000001.00000100 (First Usable Address)
 - a. 192.168.1.4
 5. 11000000.10101000.00000001.11111110 (Last Usable Address)
 - a. 192.168.1.254

- c. Classless Inter-Domain Routing (CIDR) Notation
 - i. Shorthand notation defines the number of Subnet Mask bits
 - ii. The larger the number, the less addresses available for hosts
 - 1. Example: 255.255.255.0/24 (first 24 bits are reserved)

3. AWS VPC

- a. Default VPC is assigned a CIDR range of 172.31.0.0/16
 - i. You are free to use other private address ranges
- b. Amazon VPC supports VPCs between /28 and /16 in size
- c. The minimum size of a subnet is a /28 (or 14 IP addresses.) Subnets cannot be larger than the VPC in which they are created
- d. To change the size of a VPC you must terminate your existing VPC and create a new one

4. Connecting to a VPC

- a. [Internet Gateway](#)
 - i. Scalable, redundant, and highly available VPC component
 - ii. Provides a target in your VPC route tables for Internet-routable traffic
 - iii. Perform network address translation (NAT) for instances that have been assigned public IP addresses
 - iv. Create and attach to a VPC [here](#)
- b. [Virtual private gateway](#) is the VPN concentrator on the Amazon side of the VPN connection
- c. [Customer gateway](#) is a physical device or software application on your side of the VPN connection
- d. Each VPN connection has two tunnels, with each tunnel using a unique virtual private gateway public IP address. Two customer gateways can be used for redundancy

5. Other options include AWS [Direct Connect](#)

6. Route Tables

- a. For an instance to connect to the internet it needs:
 - i. An internet gateway
 - ii. A custom route table to the internet gateway explicitly associated to the subnet containing the instance
 - iii. A public IP addresses
- b. Main route table automatically created when you create a VPC with the VPC wizard
 - i. Allows local traffic within VPC
 - ii. Implicitly associated to all subnets unless another route table has been explicitly associated to the subnet

- c. Custom route table also automatically created when you create a VPC with the VPC wizard
 - i. Allows local traffic within VPC
 - ii. Creates a route to the internet gateway
 - iii. Explicitly associated to the subnet

7. Public and Private Subnets

8. Network Address Translation (NAT)

- a. Allows traffic to flow from a private subnet to an Internet gateway via a public subnet
- b. Can be either a NAT Gateway or a NAT instance (hosted in EC2). Gateways are more resilient although they incur higher costs
- c. Can use **bastion hosts** to SSH into private subnet instances from outside its security group (be careful)

9. VPC Security

- a. Security groups
 - i. Firewall at the instance level
- b. Network access control lists (ACLs)
 - i. Firewall at the subnet level
- c. Flow logs
 - i. Capture VPC information as CloudWatch logs

Security Group	Network ACL
Operates at the instance level (first layer of defense)	Operates at the subnet level (second layer of defense)
Supports allow rules only	Supports allow rules and deny rules
Is stateful : Return traffic is automatically allowed, regardless of any rules	Is stateless : Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic. If there is more than one rule for a specific port, we apply the most permissive rule.	We process rules in number order when deciding whether to allow traffic. Most restrictive deny applies.
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (backup layer of defense, so you don't have to rely on someone specifying the security group)

d.

10. VPC Peering

- a. Networking connection between 2 VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses
- b. Instances in either VPC can communicate with each other as if they are within the same network
- c. Can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account
- d. VPCs must be in the same region
- e. Transitive peering is not allowed

f. [VPC peering basics](#)

CloudFront

1. Introduction

- a. Web service for high performance content delivery
- b. World-wide network of data centres (edge locations)
- c. Reduces number of hops for requests
- d. Can set geographic restrictions (whitelist or blacklist countries)
- e. Can invalidate cached objects (additional costs)

2. CloudFront Origin Server

- a. Location of content to be delivered (Inside or outside AWS)
- b. Origin server is either an S3 bucket or an HTTP server (e.g. EC2 instance)
- c. Can also be a web server you manage
- d. CloudFront must have access to origin server

3. CloudFront Distribution

- a. Distribution tell CloudFront which origin servers to get your files from and the TTL
- b. CloudFront assigns a domain name to your new distribution (can be alternate domain CNAME)
- c. If an edge location contains the requested file within TTL it is delivered. If not, it is fetched from the origin server then delivered and cached at the edge location
- d. Distributions, or parts of, can be invalidated to force new content

4. Delivering Dynamic Content

- a. Options:
 - i. Low Time to Live (TTL)
 - ii. Do not cache dynamic content, only static:
 - 1. Not all HTTP methods are cached by CloudFront, only responses to GET and HEAD requests (although you can also configure CloudFront to cache responses to OPTIONS requests)
 - a. Other HTTPs methods, such as POST; PUT; or DELETE, will not be cached by CloudFront. CloudFront will instead proxy these requests back to the origin server

Route 53



1. Quick Overview of Domain Name System (DNS)

- a. DNS translate human friendly domain names into IP addresses
- b. It is a globally distributed service for the entire Internet
- c. 2 IP types are currently supported → IPv4 (32 bits) and IPv6 (128 bits)
- d. Top Level Domain (TLD)
 - i. Last word after the last '.' In a domain name (e.g. .com)
 - ii. Penultimate word in domain name = second level domain name (not requires (such as – .co.uk, .org.uk)
 - iii. Managed by [IANA](#) (Internet Assigned Numbers Authority)
- e. Domain Registrars
 - i. Service that lets you officially register a domain name so that it is unique to you and nobody else can own it
 - ii. This authority can allow you to register domain names under one or more TLDs
 - iii. Domain name registrars are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN)
- f. Fundamental Record Types:
 - i. **'A' Record (address record)** – is used to find the IP address of a computer connected to the internet from a name. E.g. maps [www.example.com](#) to an IP address (e.g. 10.1.0.0)
 - ii. **CNAME (canonical name record)** – maps a domain name to another one e.g. [www.exampe.com](#) mapped to [www.example.com](#) to redirect in case of typos
 - iii. **NS (name server record)** – indicates which name servers are authoritative for the domain. Mainly used if you want to break your domain into subdomains: delegating a part of a domain name to a different group of name servers
 - iv. **SOA (start of authority record)** – records are used to determine how your zone propagates to the secondary name servers. Contains info such as: name of primary DNS server, e-mail address of person responsible
 - v. **TTL (Time to Live)** – Specifies the maximum amount of time other DNS servers and applications should cache a DNS record. Because of caching, changes to a DNS record will not reach the entire network until the original TTL has expired

2. Introduction

- a. Highly available and scalable Domain Name System (DNS) web service
- b. Domain name registration
- c. Health-checking web services
- d. Not part of the free-tier 😊

3. Key Concepts

- a. Public Hosted Zone – Container that holds information about how it should route traffic on the Internet for a domain and its subdomains
- b. Private Hosted Zone – within VPCs

- c. Resource Record Sets – Tell the DNS how to route traffic for your domain
- d. Traffic Policies – Complex routing configs
- e. Health Checks – monitor the health and performance of your web applications, web servers, and other resources
- f. Domain Registration and Transfers
- g. Alias Records – Provide an Amazon Route 53 specific extension to DNS
 - i. Alias resource record set contains a pointer to CloudFront distributions, S3 buckets, Elastic Beanstalk, ELB, or another Amazon Route 53 resource record set in the same hosted zone. Similar to CNAMEs
 - ii. Remember! You can create an Alias record at the zone apex of a Domain, but you cannot create a CNAME for it. CNAME queries will be charged

4. Policy Routing Types

- a. Simple Routing
 - i. When you have a single resource that performs a specific function for your domain (For example: A web server for the example.com website)
 - ii. Route 53 responds to DNS queries based only on the values in the resource record set (For example: The IP address in an A record)
 - iii. Default policy when you create a Record Set

5. Weighted Routing

- i. When you have multiple resources that perform the same function (e.g. web servers that serve the same website) and you want to route traffic to them in proportions that you specify (For example: ¾ traffic to first server and ¼ traffic to second one)
- ii. Use Cases: Load balancing and testing new versions of software (A/B testing)
- b. Latency Routing
 - i. When you have resources in multiple regions that perform the same function and you want Route 53 to respond to DNS queries with the resources with the lowest latency for you users
 - ii. Latency resource records allowed for EC2 instances and ELBs
 - iii. Latency between hosts on the Internet can change over time as a result of changes in network connectivity and routing
- c. Failover Routing
 - i. When you want to configure active-passive failover, in which traffic is directed to one resource when it's available (primary) and to the other resource when the primary is not available (secondary)
 - ii. Useful for Disaster Recover (DR)
 - iii. Route 53 monitors your primary site using a health check
- d. Geolocation Routing
 - i. When you want Route 53 to respond to DNS queries based on the location of you users
 - ii. Use Cases Include:

1. Localized content to present to some or all of your website in the language of your users
2. Restrict distribution of content to only the locations in which you have distribution rights

Messaging



Simple Queuing Service (SQS)

1. Introduction

- a. Queues for storing messages between apps
- b. Acts as a buffer of data for processing servers. SQS can smooth out the peak demand
- c. Up to 10 attributes (name and values) can be added to a message additional to the message body
- d. Message size can be set from 1KB to 256KB
 - i. To send messages larger than 256KB, use the [Amazon SQS Extended Client Library for Java](#). This library lets you send an Amazon SQS message that contains a reference to a message payload in Amazon S3 that can be as large as 2GB.
- e. Messages can be kept in SQS queues from between 1 minute and 14 days. The default is 4 days. Once the message retention limit is reached, your messages are automatically deleted
- f. Messages can contain text data, including XML, JSON, and unformatted text. Accepted Unicode characters:
 - i. `#x9 | #xA | #xD | [#x20 to #xD7FF] | [#xE000 to #xFFFD] | [#x10000 to #x10FFFF]`
- g. Queues can contain an unlimited number of messages. However, only 120,000 inflight messages allowed for standard queues and only 20,000 for FIFO queues
- h. **Any number** of queues can be created
- i. 1 million requests per month for free

2. Decoupling Processes

- a. If average demand exceeds processing capacity, queue will grow indefinitely
- b. SQS can provide CloudWatch metrics that can be used with Autoscaling
- c. Can provide upper and lower setpoints

- d. **Best-Effort ordering** (messages can be delivered in a different order than how they were sent)
- e. **At-least-once delivery**
- f. If you need ordering to be maintained, you should embed sequencing info into your messages so they can be reordered when received

3. Queue Types

- a. Standard Queue
 - i. Default queue type
 - ii. Nearly-unlimited number of transactions per second
 - iii. Guarantee that message is delivered at least once although rarely duplicates
 - iv. Best-effort ordering
- b. First-In-First-Out (FIFO)
 - i. Complements standard type
 - ii. Limited to 300 transactions per second (TPS)
 - iii. Exactly-once processing
 - iv. Currently not available in all regions

4. Message Lifecycle/[Visibility timeout](#)

- a. Message received by SQS queue
- b. Message received by processing server. Visibility timeout period starts
- c. Message processed by processing server. Processing server sends message delete request. Visibility timeout period ends
- d. **Note:** visibility timeout specifies the time which the message is **invisible** in the queue and has **not** been deleted (default 30 seconds)
 - i. If the message has not been deleted by the visibility timeout period, the message becomes visible and can be received again by a processing server

5. Dead Letter Queues

- a. Queue that can be a target for messages that cannot be processed successfully
- b. This allows them to be analyzed or reprocessed later
- c. Dead letter queues must be in the same region and from the same account as the corresponding SQS queue

6. Delay Queues and Message Timers

- a. Delay Queues
 - i. Postpone the delivery of new messages
 - ii. Message is not visible when it is first added to the queue for a defined period of time
 - iii. Set *DelaySeconds* parameter with `CreateQueue` or `SetQueueAttributes` for existing queue

- iv. 120,000 limit for the number of inflight messages per queue
 - b. Message Timers
 - i. Initial invisibility period for an individual message
 - ii. Can be created using the console or with *DelaySeconds* parameter of *SendMessage*
7. Types of Polling
- a. **Short polling** – send response immediately back whether or not messages in queue
 - b. **Long polling** – waits until a message is available in the queue before sending a response
 - i. Reduces the number of empty responses when there are no messages available to return
 - ii. Set *WaitTimeSeconds* (*ReceiveMessage*) or *ReceiveMessageWaitTimeSeconds* (*CreateQueue* or *SetQueueAttributes*) parameter between 1 to 20
 - iii. [More Info](#) about long polling
8. Additional Documentation
- a. SQS [FAQs](#)
 - b. [Amazon SQS API Reference](#)

Simple Notification Service ([SNS](#))

1. Introduction
- a. Enables the sending messages up to 256KB to subscribing endpoints or clients
 - b. Topic name is unique and describes the endpoint for publishers to post messages
 - c. Subscribers subscribe to a topic name. Subscriber access to topics determined through policies
2. Transport Protocols
- a. HTTP, HTTPS – Subscribers specify a URL as part of the subscription registration; notifications will be delivered through an HTTP POST to the specified URL
 - b. Email, Email-JSON – Messages are sent to registered addresses as email. Email-JSON sends notifications as a JSON object, while Email sends text-based email
 - c. SQS – Users can specify an SQS standard queue as the endpoint. Note that FIFO queues are not currently supported
 - d. SMS – Messages are sent to registered phone numbers as SMS text messages
3. Message Format
- a. Email transport only contain payload (message body)
 - b. Format for HTTP, HTTPS, Email-JSON and SQS:

- i. MessageId: A universally Unique Identifier, unique for each notification published
 - ii. Timestamp: Time (in GMT) at which the notification was published
 - iii. TopicArn: The topic to which this message was published
 - iv. Type: The type of the delivery message set to “Notification” for notification services
 - v. UnsubscribeURL: A link to unsubscribe the end-point from this topic, and prevent receiving any further notification
 - vi. Message: The payload (body) of the message, as received from the publisher
 - vii. Subject: The subject field – if one was included as an optional parameter to the publish API call along with the message
 - viii. Signature: Base64-encoded “SHA1withRSA” signature of the Message, MessageId, Subject (if present), Type, Timestamp, and Topic values
 - ix. SignatureVersion: Version of the Amazon SNS signature used
 - x. Token: A value you can use with the [ConfirmSubscription](#) action to confirm the subscription. Alternatively, you can simply visit the SubscribeURL. Valid 3 days
- c. SNS Mobile Push Notifications
 - i. High level steps:
 1. Request Credentials from Mobile Platforms
 2. Request Token from Mobile Platforms
 - a. ADM, GCM registration ID
 - b. APNS device token
 3. Create SNS Platform Application Object
 4. Create SNS Platform Endpoint Object
 5. Publish SNS Message to Mobile Endpoint
 - ii. Amazon Device Messaging (ADM) Steps:
 1. Create a Kindle Fire App with the ADM Service Enabled
 2. Obtain a Client ID and Client Secret
 3. Obtain an API Key
 4. Obtain a Registration ID
 - iii. Apple Push Notification Service (APNS) Steps:
 1. Create an iOS App
 2. Obtain an APNS SSL Certificate
 3. Obtain the App Private Key
 4. Verify the Certificate and App Private Key
 5. Obtain a Device Token
 - iv. Google Cloud Messaging (GCM) for Android Steps:
 1. Create a google API Project and Enable the GCM
 2. Obtain the Server API Key
 3. Obtain a Registration ID from GCM

4. More documents
 - a. SNS [FAQs](#)
 - b. SNS fanout scenario using SQS [here](#)
 - c. JSON Formats [here](#)
 - d. SNS message attributes [here](#)
 - e. Send SNS Message to HTTP/HTTPS Endpoints [here](#)

Analytics & Big Data

1. Big Data Services
 - a. Database/Storage
 - i. Redshift
 - ii. DynamoDB
 - iii. S3
 - iv. RDS
 - b. Analysis
 - i. Elastic MapReduce (EMR)
 - ii. Elasticsearch
 - iii. QuickSight BI
 - iv. Amazon Machine Learning
 - v. Lambda
 - c. Real Time Data
 - i. Kinesis Streams
 - d. Third-Party applications in EC2 instances
2. Big Data Migration
 - a. Services available
 - i. Database snapshots to S3
 - ii. AWS Database Migration Service
 - iii. AWS Data Pipeline
 - iv. AWS Snowball to S3



1. Introduction
 - a. Petabyte scale data warehousing service
 - b. Based upon PostgreSQL
 - c. Accessed with standard BI reporting tools
 - d. All data replicated between nodes in a cluster
 - e. Continuously backed up to S3 with snapshots (held for 1 to 35 days)
 - f. User initiated snapshots are retained upon cluster deletion
 - g. Quick recovery from snapshots

Elastic MapReduce ([EMR](#))



1. Introduction
 - a. Fully managed Hadoop service
 - b. Apache Hadoop is an open source framework for the distributed processing of large data sets across clusters of compute instances
 - c. Clusters of EC2 instances analyze data
 - d. Clusters can be automatically deleted upon task completion
 - e. Not suitable for small data sets and ACID transaction requirements
2. Data Processing Frameworks:
 - a. Hadoop MapReduce
 - b. Apache Spark
3. Storage options
 - a. Hadoop Distributed File System (HDFS) ephemeral storage
 - b. EMR File System (EMRFS) to access S3 as a file system
 - c. Local File System (EC2 instance store)

[Elasticsearch](#)



1. Introduction
 - a. Elasticsearch is a real-time distributed search and analytics engine. It is the most popular enterprise search engine (Facebook, GitHub, Stack Exchange, Quora, ...)
 - b. AWS Elasticsearch is a fully managed implementation of Elasticsearch
 - c. Analyze data from:
 - i. S3
 - ii. Kinesis Streams
 - iii. DynamoDB
 - iv. CloudWatch Logs
 - v. CloudTrail API call logs
 - d. Suitable for querying and searching large amounts of data
 - e. Not Suitable for:
 - i. Online transaction processing (OLTP)
 - ii. Petabyte storage (consider using self-managed EC2)

QuickSight

1. Introduction

- a. Business Intelligence (BI) reporting service
- b. Super-fast, Parallel, In-memory, Calculation Engine (SPICE)
- c. 1/10 cost of traditional BI software

Machine Learning

1. Introduction

- a. Predictive analytics and machine-learning simplified with visualization tools and wizards
- b. Data sources:
 - i. S3
 - ii. Redshift
 - iii. RDS (MySQL)
- c. Suitable to:
 - i. Flag suspicious transactions
 - ii. Forecast product demand
 - iii. Personalize application content
 - iv. Predict user activity
 - v. Analyze social media
- d. Not Suitable for
 - i. Very large data sets
 - ii. Unsupported learning tasks
- e. Consider EMR to run Spark and MLlib

Kinesis

1. Introduction

- a. Data can be put into streams using
 - i. API calls (HTTP PUT)
 - ii. SDK
 - iii. Amazon Kinesis Producer Library (KPL) C++ application
 - iv. Kinesis Agent java application
- b. Kinesis Client Library (KCL) can be used to process data already in a stream
 - i. Java, Node.js, .NET, Python, Ruby
- c. Kinesis Firehose can capture, transform, and load streaming data into Amazon Kinesis Analytics, Amazon S3, Amazon Redshift, and Amazon Elasticsearch

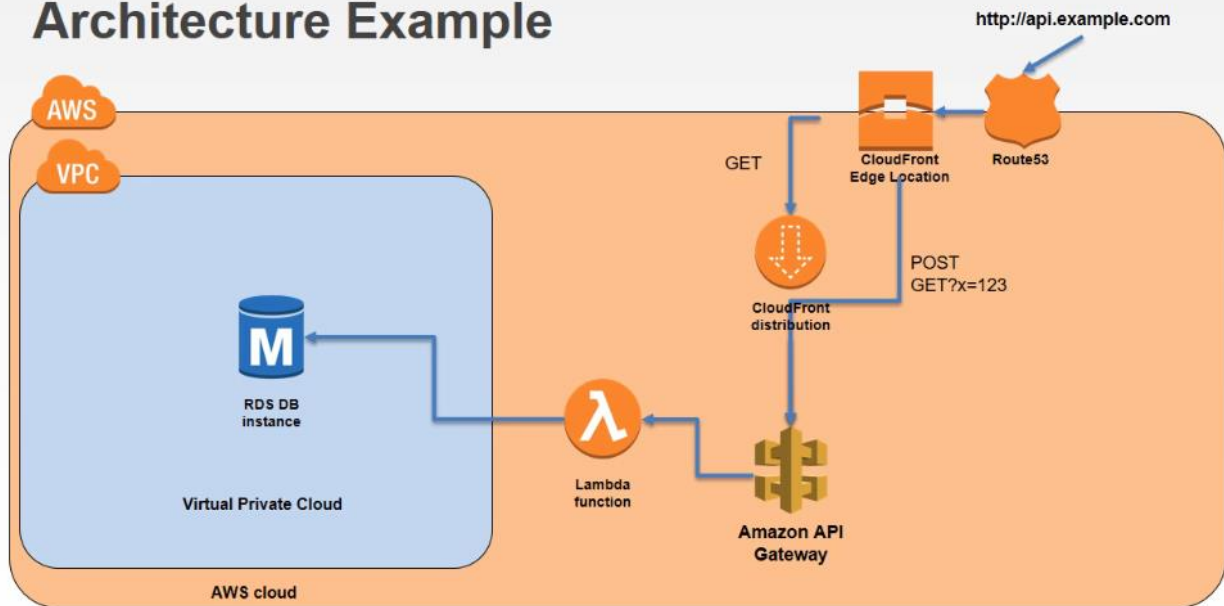
- d. Suitable for:
 - i. Real-time data analytics
 - ii. Log and data feed intake and processing
 - iii. Real-time metrics and reporting
- e. Not Suitable for:
 - i. Small scale consistent throughput
 - ii. Long-term data storage and analytics

Application Services

API Gateway

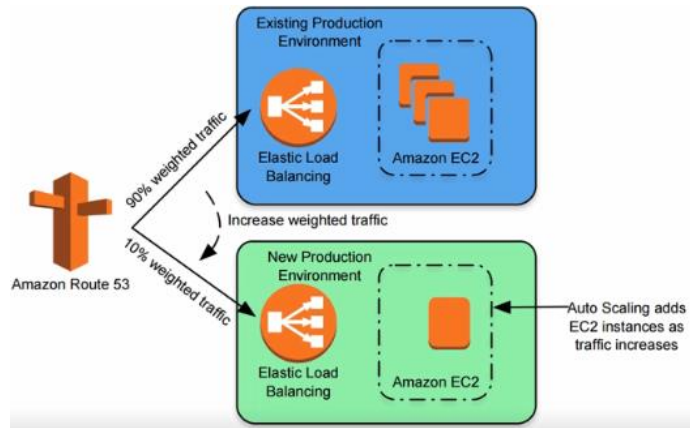
1. Introduction
 - a. Acts as a secure “front door” for applications to access data, business logic, or functionality from your back-end services
 - b. Serverless – A simple, flexible, fully managed, pay-as-you-go service that handles all aspects of creating and operating robust APIs for applications back ends

Architecture Example



Deployment in AWS

1. Infrastructure as Code
 - a. Allows infrastructure to be managed in the same way as software
 - b. Version Control
 - c. Examples:
 - i. CloudFormation Templates
 - ii. CloudFormation Designer
 - iii. AMI
2. Continuous Deployment
 - a. Application
 - i. Automated delivery of production ready code
 - ii. Allows rapid deployment and roll back if necessary
 - b. Application and Infrastructure
 - i. **Hybrid Deployments** e.g. separating application and database, or application and infrastructure deployments
 - c. Examples:
 - i. CodeCommit
 - ii. CodePipeline
 - iii. Elastic Beanstalk
 - iv. OpsWorks
 - v. Elastic Container Service (ECS)
3. Application Updating Options
 - a. Prebaking AMI
 - b. In-place Upgrades
 - i. Application updates on live Amazon EC2 instances
 - c. Disposable Upgrade
 - i. Rolling out new EC2 instances and terminating older instances
 - ii. Allows staged deployment
4. Blue-Green Deployments
 - a. Staged rollout from existing (blue) environment while testing a new (green) one
 - b. Uses domain name services (DNS) to increase traffic to green environment in stages
 - c. Requires doubling up on resources



d.