

# AWS Certified Developer Associate

# Getting Started

In 28  
Minutes



EC2



DynamoDB



AWS Lambda



API Gateway

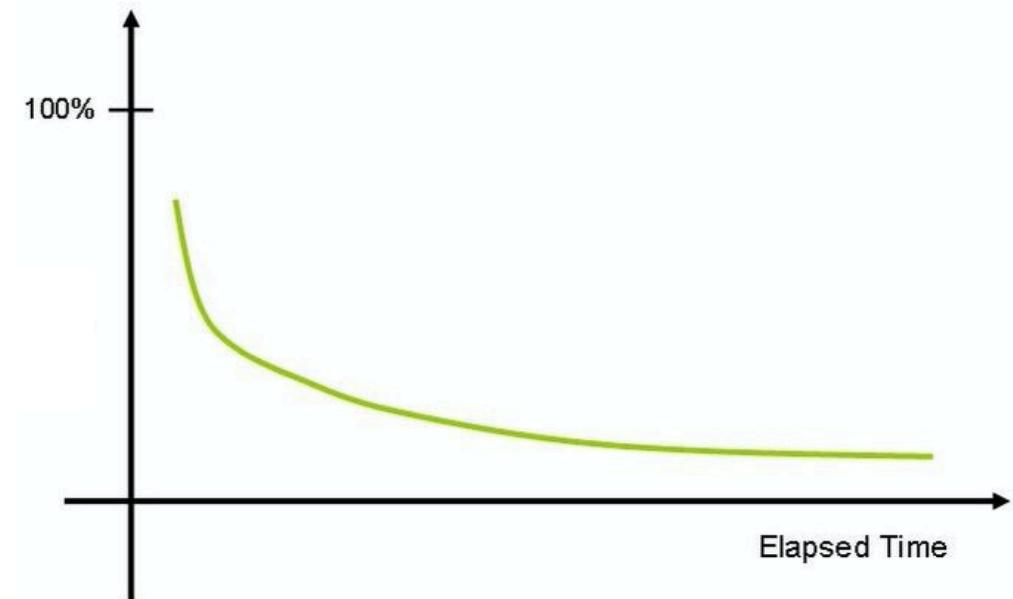


Amazon S3

- AWS has 200+ services. Exam expects knowledge of 40+ services.
- Exam **tests your decision making abilities:**
  - Which service do you choose in which situation?
- Exam expects **in-depth knowledge** for certain services:
  - EC2, Serverless (Lambda, API Gateway), Cognito etc
- This course is **designed** to give you *in-depth* knowledge & *make tough choices*
- **Our Goal :** Enable you to build amazing solutions in AWS

# How do you put your best foot forward?

- **Challenging certification** - Expects you to understand and **REMEMBER** a number of services
- As time passes, humans forget things.
- How do you improve your chances of remembering things?
  - Active learning - think and make notes
  - Review the presentation every once in a while



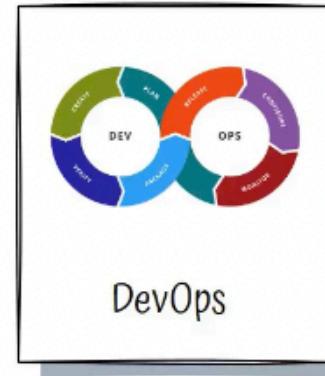
# Our Approach

- Videos with presentations and demos
- **Two kinds of quizzes** to reinforce concepts:
  - Text quizzes : Traditional text quiz
  - Video quizzes : Discuss various scenario questions and the solution in a video
- **Practice test** at the end of the course
- (Recommended) Take your time. Do not hesitate to replay videos!
- (Recommended) Have Fun!



# FASTEST ROADMAPS

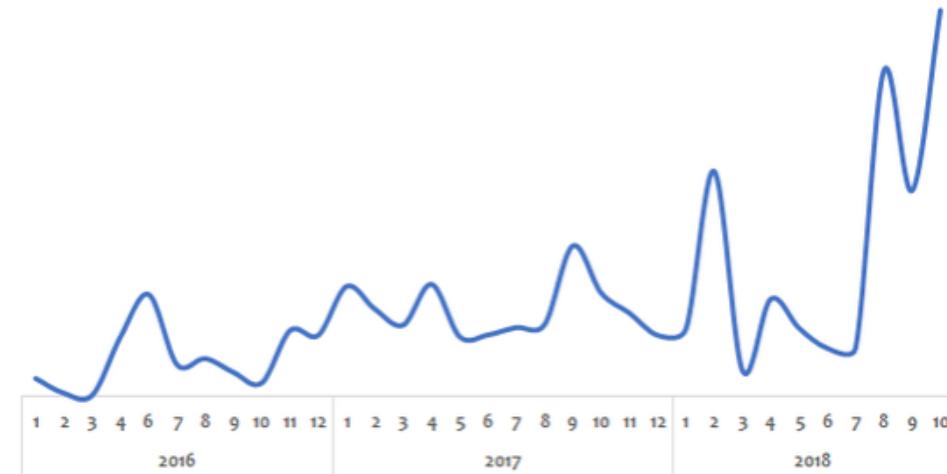
in28minutes.com



# AWS - Getting started

# Before the Cloud

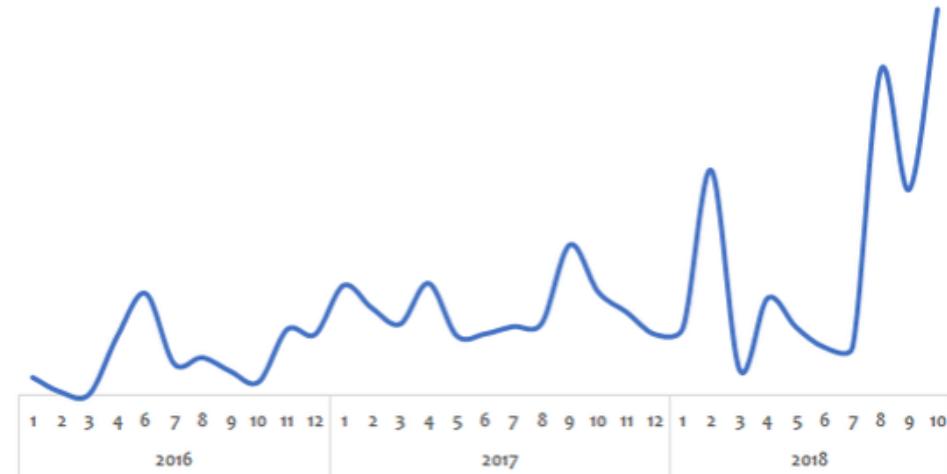
In 28  
Minutes



- Consider a Online Shopping Application:
  - Peak usage during holidays and weekends. Less load during rest of the time.
- A startup suddenly becomes popular:
  - How does it handle the sudden increase in load?
- Enterprises **procured** (bought) infrastructure **for peak load**
  - Startups procured infrastructure assuming they would be successful

# Before the Cloud - Challenges

In 28  
Minutes



- Low infrastructure utilization
- Needs ahead of time planning (**Can you guess the future?**)
- High costs of procuring infrastructure
- Dedicated infrastructure maintenance team (**Can a startup afford it?**)

# Silver Lining in the Cloud

- How about provisioning (renting) resources when you want them and releasing them back when you do not need them?
  - On-demand resource provisioning
- Advantages
  - Lower costs (Pay per use)
  - No upfront planning needed
  - Avoid "undifferentiated heavy lifting"
- Challenge
  - Building cloud enabled applications



# Amazon Web Services (AWS)

- Leading cloud service provider
- Provides most (200+) services
- Reliable, secure and cost-effective
- The entire course is all about AWS. You will learn it as we go further.



# Best path to learn AWS!



Amazon S3



EC2



Amazon EBS



ELB



ECS

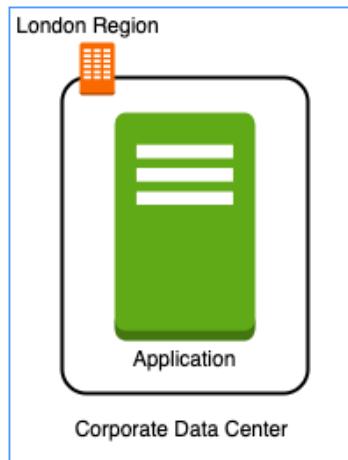
- Cloud applications make use of multiple AWS services.
- There is no **single path** to learn these services independently.
- **HOWEVER**, we've worked out a simple path!

# Setting up AWS Account

- Create AWS Account
- Setup an IAM user

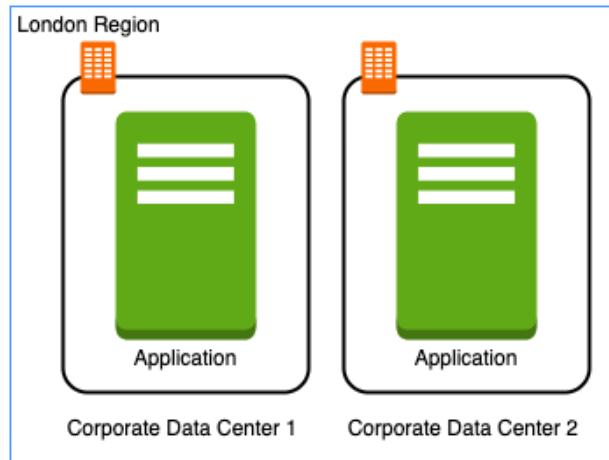
# Regions and Zones

# Regions and Zones



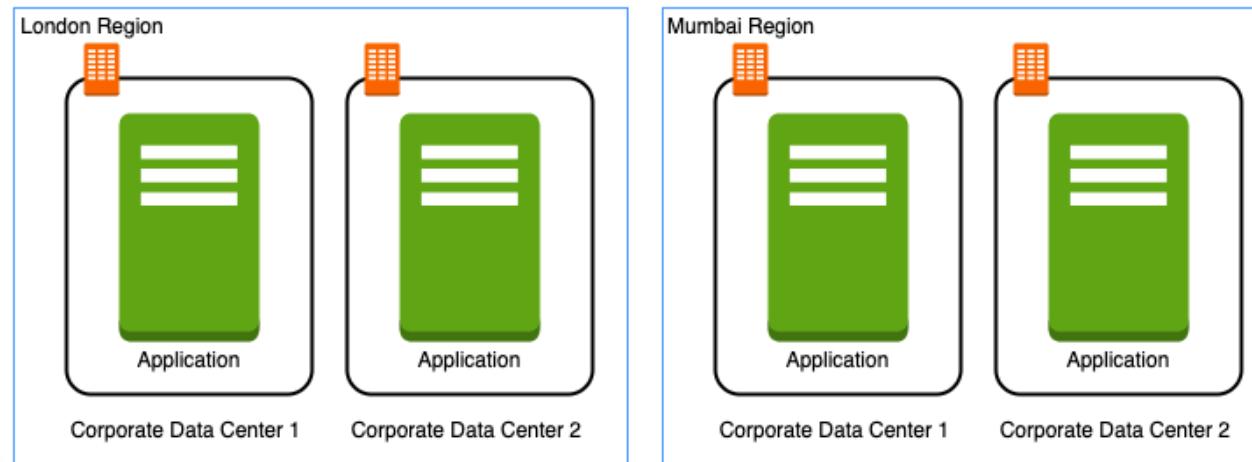
- Imagine that your application is deployed in a data center in London
- What would be the challenges?
  - Challenge 1 : Slow access for users from other parts of the world (**high latency**)
  - Challenge 2 : What if the data center crashes?
    - Your application goes down (**low availability**)

# Multiple data centers



- Let's add in one more data center in London
- What would be the challenges?
  - Challenge 1 : Slow access for users from other parts of the world
  - Challenge 2 (**SOLVED**) : What if one data center crashes?
    - Your application is still available from the other data center
  - Challenge 3 : What if entire region of London is unavailable?
    - Your application goes down

# Multiple regions



- Let's add a new region : Mumbai
- What would be the challenges?
  - Challenge 1 (**PARTLY SOLVED**) : Slow access for users from other parts of the world
    - You can solve this by adding deployments for your applications in other regions
  - Challenge 2 (**SOLVED**) : What if one data center crashes?
    - Your application is still live from the other data centers
  - Challenge 3 (**SOLVED**) : What if entire region of London is unavailable?
    - Your application is served from Mumbai

# Regions

In 28  
Minutes



- Imagine setting up your own data centers in different regions around the world
  - Would that be easy?
- (Solution) AWS provides **20+ regions** around the world (expanding every year)

# Regions - Advantages



- High Availability
- Low Latency
- Adhere to government **regulations**

# Regions

- Choose the right region(s) based on:
  - Where are your users located?
  - Where is your data located?
  - Regulatory and security compliance needs
- AWS Services can be:
  - Regional (OR)
  - Global



# Availability Zones

- Isolated locations in a Region
- Each AWS Region has at least two Availability Zones
- Increase availability of applications in the same region



# Regions and Availability Zones examples

*New Regions and AZs are constantly added*

Region Code	Region	Availability Zones	Availability Zones List
us-east-1	US East (N. Virginia)	6	us-east-1a us-east-1b us-east-1c us-east-1d us-east-1e us-east-1f
eu-west-2	Europe (London)	3	eu-west-2a eu-west-2b eu-west-2c
ap-south-1	Asia Pacific(Mumbai)	3	ap-south-1a ap-south-1b ap-south-1c

# EC2 Fundamentals

# EC2(Elastic Compute Cloud)

- In corporate data centers, applications are deployed to physical servers
- Where do you deploy applications in the cloud?
  - Rent virtual servers
  - **EC2 instances** - Virtual servers in AWS (billed by second)
  - **EC2 service** - Provision EC2 instances or virtual servers

# EC2 Features

In 28  
Minutes



EC2 Instances



ELB



Amazon EBS

- Create and manage lifecycle of EC2 instances
- **Load balancing and auto scaling** for multiple EC2 instances
- **Attach storage** (& network storage) to your EC2 instances
- Manage **network connectivity** for an EC2 instance
- **Our Goal:**
  - Setup EC2 instances as HTTP Server
  - Distribute load with Load Balancers

# EC2 Hands-on

- Let's create a few EC2 instances and play with them
- Let's check out the lifecycle of EC2 instances
- Let's use EC2 Instance Connect to SSH into EC2 instances

# EC2 Instance Types

- Optimized combination of **compute(CPU, GPU), memory, disk (storage) and networking** for specific workloads
- 270+ instances across 40+ instance types for different workloads
- **t2.micro:**
  - t - Instance Family
  - 2 - generation. Improvements with each generation.
  - **micro** - size. (*nano < micro < small < medium < large < xlarge < .....*)
- (Remember) As size increases, compute(CPU, GPU), memory and networking capabilities increase proportionately

# EC2 - Instance Metadata Service and Dynamic Data

## Instance Metadata Service:

- Get details about EC2 instance **from inside** an EC2 instance:
  - AMI ID, storage devices, DNS hostname, instance id, instance type, security groups, IP addresses etc
- URL: *http://169.254.169.254/latest/meta-data/*
- URL Paths: network, ami-id, hostname, local-hostname, local-ipv4 , public-hostname, public-ipv4, security-groups, placement/availability-zone

## Dynamic Data Service:

- Get dynamic information about EC2 instance:
- URL: *http://169.254.169.254/latest/dynamic/*
- Example: *http://169.254.169.254/latest/dynamic/instance-identity/document*

# EC2 Hands-on : Setting up a HTTP server

```
sudo su
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
echo "Getting started with AWS" > /var/www/html/index.html
echo "Welcome to in28minutes $(whoami)" > /var/www/html/index.html
echo "Welcome to in28minutes $(hostname)" > /var/www/html/index.html
echo "Welcome to in28minutes $(hostname -i)" > /var/www/html/index.html
```

# Security Groups



- **Virtual firewall** to control **incoming and outgoing** traffic to/from AWS resources (EC2 instances, databases etc)
- Provides additional layer of security - Defense in Depth

# Security Groups Rules

Inbound rules	Outbound rules	Tags		
<b>Inbound rules</b>				
<a href="#">Edit inbound rules</a>				
Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	::/0	-
HTTPS	TCP	443	183.82.136.27/32	-

- Security groups are **default deny**
  - If there are no rules configured, no outbound/inbound traffic is allowed
- You can specify **allow rules ONLY**
- You can configure **separate rules** for inbound and outbound traffic
- You can assign multiple (upto five) security groups to your EC2 instances

# Security Groups

- You can add and delete security groups to EC2 instances at any time.
  - Changes are immediately effective
- Traffic NOT explicitly allowed by Security Group **will not reach** the EC2 instance
- Security Groups are **stateful**:
  - If an outgoing request is allowed, the incoming response for it is automatically allowed.
  - If an incoming request is allowed, an outgoing response for it is automatically allowed

# Security Group - Trivia

- What if there are no security group rules configured for inbound and outbound?
  - Default DENY. No traffic is allowed in and out of EC2 instance.
- Can I change security groups at runtime?
  - Yes. Changes are immediately effective.

# EC2 IP Addresses

- Public IP addresses are internet addressable.
- Private IP addresses are **internal** to a corporate network
- You CANNOT have two resources with same public IP address.
- HOWEVER, two different corporate networks CAN have resources with same private IP address
- **All EC2 instances** are assigned private IP addresses
- Creation of public IP addresses **can be enabled** for EC2 instances in public subnet
- (Remember) When you stop an EC2 instance, public IP address is lost
- **DEMO:** EC2 public and private addresses

# Elastic IP Addresses

- Scenario : How do you get a **constant public IP address** for a EC2 instance?
  - Quick and dirty way is to **use an Elastic IP!**
- **DEMO:** Using Elastic IP Address with an EC2 instance

# Elastic IP Addresses - Remember

- Elastic IP can be switched to another EC2 instance **within the same region**
- Elastic IP **remains attached** even if you stop the instance. You have to manually detach it.
- Remember : You are charged for an Elastic IP when you are NOT using it! Make sure that you explicitly release an Elastic IP when you are not using it
- You will be charged for Elastic IP when:
  - Elastic IP is NOT associated with an EC2 instance OR
  - EC2 instance associated with Elastic IP is stopped

# Simplify EC2 HTTP server setup

- How do we reduce the number of steps in creating an EC2 instance and setting up a HTTP Server?
- Let's explore a few options:
  - Userdata
  - Launch Template
  - AMI

# Bootstrapping with Userdata

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
curl -s http://169.254.169.254/latest/dynamic/instance-identity/document >
```

- **Bootstrapping:** Install OS patches or software when an EC2 instance is launched.
- In EC2, you can configure **userdata** to bootstrap
- Lookup user data - *http://169.254.169.254/latest/user-data/*
- **DEMO** - Using Userdata

# Launch Templates

- Why do you need to specify all the EC2 instance details (AMI ID, instance type, and network settings) **every time** you launch an instance?
- How about creating a **Launch Template**?
- Allow you to launch Spot instances and Spot fleets as well
- **DEMO** - Launch EC2 instances using Launch Templates

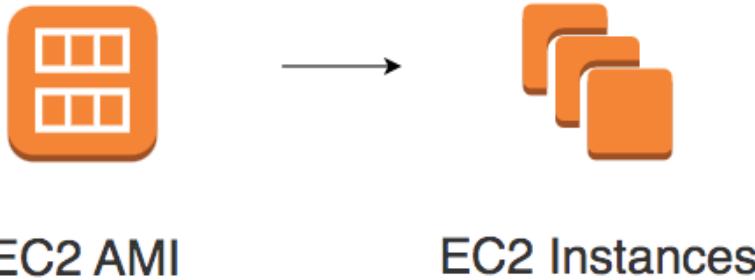
# Reducing Launch Time with Customized AMI

In 28  
Minutes



- Installing OS patches and software using userdata at launch of EC2 instances **increases boot up time**
- How about creating customized AMIs with OS patches and software **pre-installed?**
  - Hardening an Image - Customize EC2 images to your corporate security standards
- **Prefer** using Customized AMI to userdata
- **DEMO** : Create a Customized AMI and using it in Launch Template

# AMI - Amazon Machine Image



- What operating system and what software do you want on the instance?
- Three AMI sources:
  - Provided by AWS
  - **AWS Market Place:** Online store for customized AMIs. Per hour billing
  - **Customized AMIs:** Created by you.

# EC2 Amazon Machine Image - AMI - Remember

- AMIs contain:
  - Root volume block storage (OS and applications)
  - Block device mappings for non-root volumes
- You can configure launch permissions on an AMI
  - Who can use the AMI?
  - You can share your AMIs with other AWS accounts
- AMIs are stored in Amazon S3 (**region specific**).
- **Best Practice:** Backup upto date AMIs in multiple regions
  - Critical for Disaster Recovery

# EC2 Security - Key Pairs

- EC2 uses public key cryptography for protecting login credentials
- Key pair - public key and a private key
  - Public key is stored in EC2 instance
  - Private key is stored by customer

# Connecting to EC2 instance(s) - Troubleshooting

- You need to have the **private key** with you
- Change permissions to **0400** (*chmod 400 /path/my-key-pair.pem*)
  - Default permissions on private key - 0777 (**VERY OPEN**)
- (Windows Instances) In addition to private key, you need admin password
  - (At Launch) Random admin password is generated and encrypted using public key
  - Decrypt the password using the private key and use it to login via RDP
- Security Group should allow inbound SSH or RDP access:
  - Port 22 - Linux EC2 instance (SSH)
  - Port 3389 - RDP (Remote Desktop - Windows)
- Connect to your instance using its Public DNS: `ec2-**-**-**-**.compute.amazonaws.com`

# Important EC2 Scenarios - Quick Review

Scenario	Solution
You want to identify all instances belonging to a project, to an environment or to a specific billing type	Add Tags. Project - A. Environment - Dev
You want to change instance type	Stop the instance. Use "Change Instance Type" to change and restart.
You don't want an EC2 instance to be automatically terminated	Turn on Termination Protection. (Remember) EC2 Termination Protection is not effective for terminations from a) Auto Scaling Groups (ASG) b) Spot Instances c) OS Shutdown
You want to update the EC2 instance to a new AMI updated with latest patches	Relaunch a new instance with an updated AMI
Create EC2 instances based on on-premise Virtual Machine (VM) images	Use VM Import/Export. You are responsible for licenses.

# Important EC2 Scenarios - Quick Review

Scenario	Solution
<b>Change security group on an EC2 instance</b>	Assign at launch or runtime. Security Group changes are immediately effective.
<b>You get a timeout while accessing an EC2 instance</b>	Check your Security Group configuration
<b>You are installing a lot of software using user data slowing down instance launch. How to make it faster?</b>	Create an AMI from the EC2 instance and use it for launching new instances
<b>I've stopped my EC2 instance. Will I be billed for it?</b>	ZERO charge for a stopped instance (If you have storage attached, you have to pay for storage)

## AMI

- What operating system and what software do you want on the instance?
- Reduce boot time and improve security by creating customized hardened AMIs.
- Region specific.
- Backup AMIs in multiple regions.
- You can share AMIs with other AWS accounts.

## EC2 Instance Types

- Optimized combination of compute(CPU, GPU), memory, disk (storage) and networking for specific workloads.

## Security Groups

- Virtual firewall to control incoming and outgoing traffic to/from AWS resources (EC2 instances, databases etc)
- Default deny. Separate allow rules for inbound and outbound traffic
- Stateful and immediately effective

## Key Pairs

- Public key cryptography (Key Pairs) used to protect your EC2 instances
- You need private key with right permissions (chmod 400) to connect to your EC2 instance. (Windows EC2 instances only) You need admin password also.
- Security group should allow SSH(22) or RDP(3389)

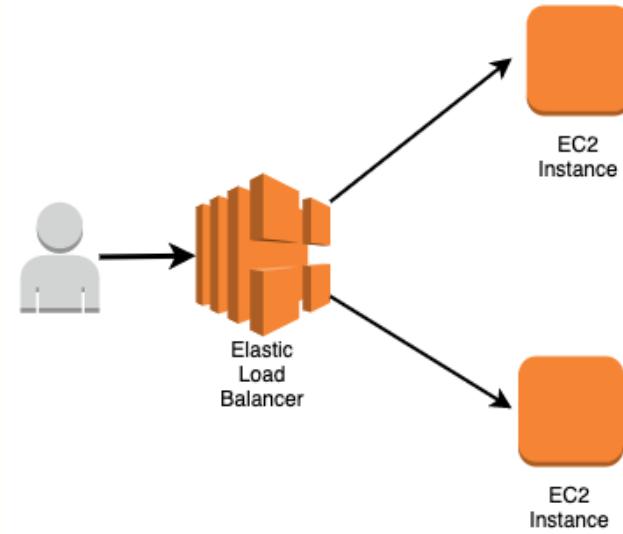
# Quick Review

- **Instance Metadata Service** - Get details about EC2 instance from inside an EC2 instance. <http://169.254.169.254/latest/meta-data/>
- **Userdata** - Used for bootstrapping. Install OS patches or software when an EC2 instance is launched.
- **Elastic IP Addresses** - Static public IP address for EC2 instance.
- **Launch Templates** - Pre-configured templates (AMI ID, instance type, and network settings) simplifying the creation of EC2 instances.

# Load Balancing

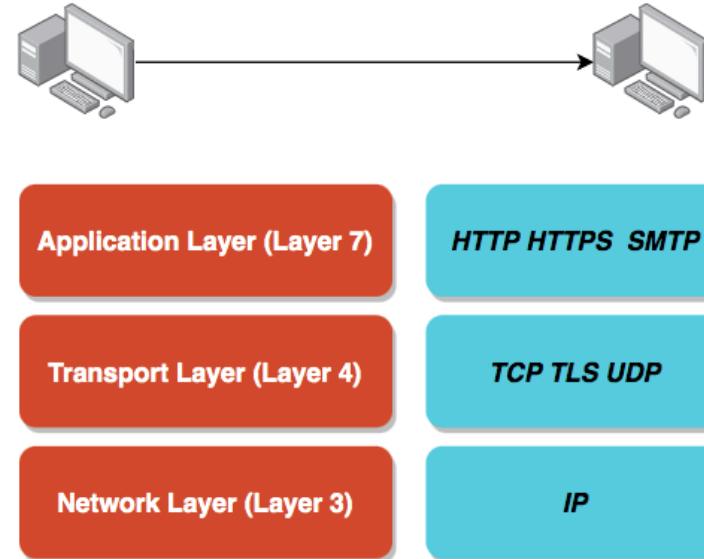
# Elastic Load Balancer

- Distribute traffic across EC2 instances in one or more AZs in a single region
- **Managed service** - AWS ensures that it is highly available
- Auto scales to handle huge loads
- Load Balancers can be **public** or **private**
- **Health checks** - route traffic to healthy instances



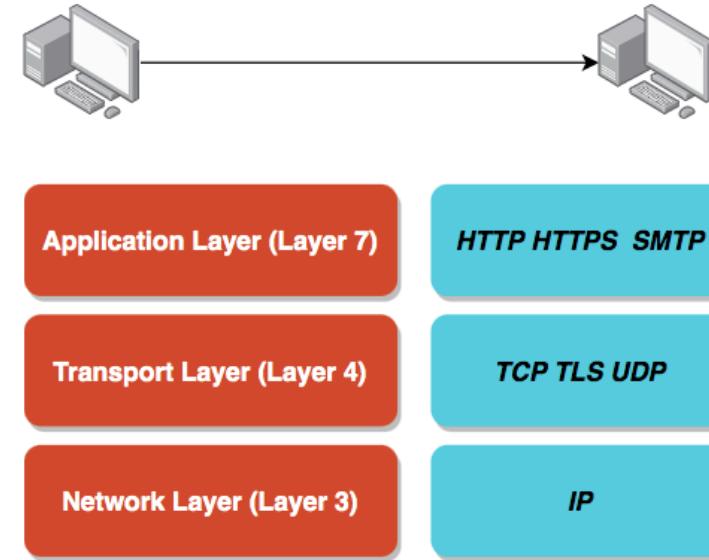
# HTTP vs HTTPS vs TCP vs TLS vs UDP

- Computers use protocols to communicate
- Multiple layers and multiple protocols
- **Network Layer** - Transfer bits and bytes
- **Transport Layer** - Are the bits and bytes transferred properly?
- **Application Layer** - Make REST API calls and Send Emails
- (Remember) Each layer makes use of the layers beneath it
- (Remember) Most applications talk at application layer. BUT some applications talk at transport layer directly (high performance).



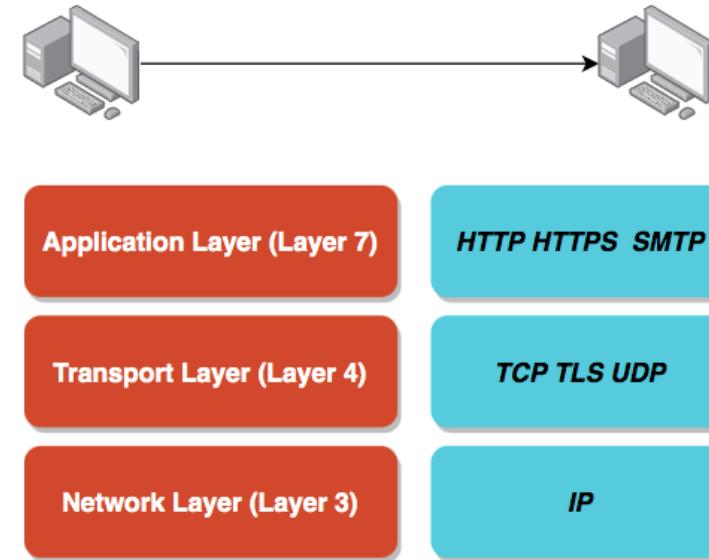
# HTTP vs HTTPS vs TCP vs TLS vs UDP

- Network Layer:
  - IP (Internet Protocol): Transfer bytes. Unreliable.
- Transport Layer:
  - TCP (Transmission Control): Reliability > Performance
  - TLS (Transport Layer Security): Secure TCP
  - UDP (User Datagram Protocol): Performance > Reliability
- Application Layer:
  - HTTP(Hypertext Transfer Protocol): Stateless Request Response Cycle
  - HTTPS: Secure HTTP
  - SMTP: Email Transfer Protocol
  - and a lot of others...



# HTTP vs HTTPS vs TCP vs TLS vs UDP

- Most applications typically communicate at application layer
  - Web apps/REST API(HTTP/HTTPS), Email Servers(SMTP), File Transfers(FTP)
  - All these applications use TCP/TLS at network layer(for reliability)
- **HOWEVER** applications needing high performance directly communicate at transport layer:
  - Gaming applications and live video streaming use UDP (sacrifice reliability for performance)
- **Objective:** Understand Big Picture. Its OK if you do not understand all details.



# Three Types of Elastic Load Balancers

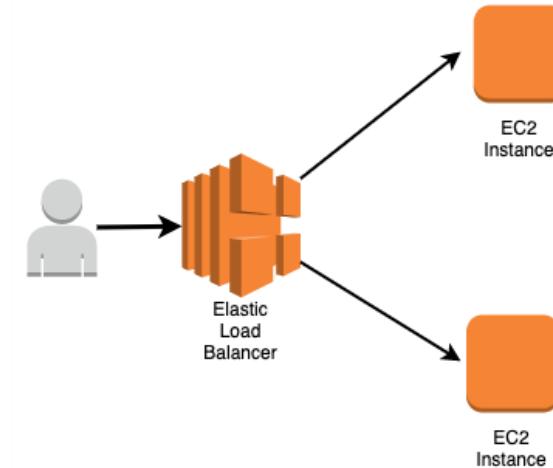
- **Classic Load Balancer (Layer 4 and Layer 7)**
  - Old generation supporting Layer 4(TCP/TLS) and Layer 7(HTTP/HTTPS) protocols
  - Not Recommended by AWS
- **Application Load Balancer (Layer 7)**
  - New generation supporting HTTP/HTTPS and advanced routing approaches
- **Network Load Balancer (Layer 4)**
  - New generation supporting TCP/TLS and UDP
  - Very high performance usecases

# Classic Load Balancer

- Older version of ELB
- Not recommended anymore
- Supports TCP, SSL/TLS and HTTP(S) (Layer 4 and 7)
- Demo: Create a Classic Load Balancer

# Application Load Balancer

- Most popular and frequently used ELB in AWS
- Supports WebSockets and HTTP/HTTPS (Layer 7)
- Supports all important load balancer features
- Scales automatically based on demand (Auto Scaling)
- Can load balance between:
  - EC2 instances (AWS)
  - Containerized applications (Amazon ECS)
  - Web applications (using IP addresses)
  - Lambdas (serverless)
- **Demo : Create an Application Load Balancer**



# Load Balancers - Security Group Best Practice

Load Balancer allow traffic from everywhere!

Inbound rules			
Type	Protocol	Port range	Source
HTTP	TCP	80	0.0.0.0/0

EC2 Security Group **ONLY** allows traffic from Load Balancer Security Group

Inbound rules			
Type	Protocol	Port range	Source
HTTP	TCP	80	sg-03eb042440351fdad (awseb-e-grzepvhpv3-stack-AWSEBLoadBalancerSecurityGroup-1EG2SPQRTIQ02)

(Best Practice) Restrict allowed traffic using Security Groups

# Listeners

The screenshot shows the 'Listeners' tab selected in a navigation bar. Below it is a descriptive text about listeners and their functions. At the bottom are three buttons: 'Add listener' (blue), 'Edit' (grey), and 'Delete' (grey). A table lists a single listener entry:

<input type="checkbox"/>	Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/>	HTTP : 80 arn...6d4fd3790d1b96d9	N/A	N/A	Default: forwarding to awseb-AWSEB-77UXO29Z6IMQ <a href="#">View/edit rules</a>

- Each Load Balancer has **one or more listeners** listening for connection requests from the client
- Each listener has:
  - a protocol
  - a port
  - a set of rules to route requests to targets

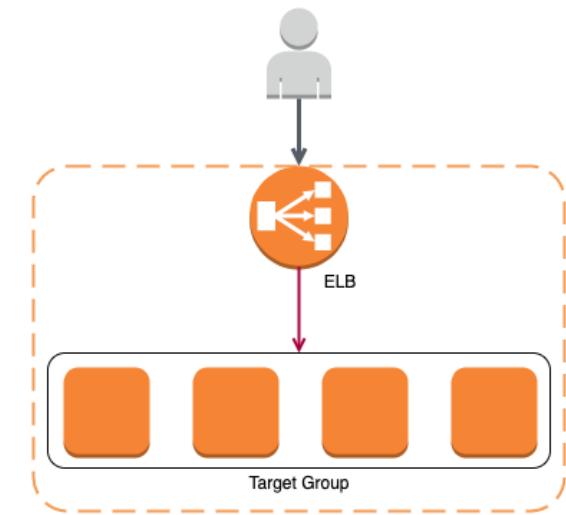
# Multiple Listeners

<input type="checkbox"/>	Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/>	HTTP : 80 arn...6d4fd3790d1b96d9 ▾	N/A	N/A	Default: forwarding to awseb-AWSEB-77UXO29Z6IMQ <a href="#">View/edit rules</a>
<input type="checkbox"/>	HTTP : 443 ▲ arn...770a0d2977599957 ▾	N/A	N/A	Default: redirecting to HTTP://#{host}:80/#{path}?#{query} <a href="#">View/edit rules</a>
<input type="checkbox"/>	HTTP : 8080 arn...8659e53f87d96af9 ▾	N/A	N/A	Default: returning fixed response 400 <a href="#">View/edit rules</a>

- You can have multiple listeners listening for a different protocol or port
- In the above example:
  - HTTP requests on port 80 are routed to the EC2 instances target group
  - HTTPS requests on port 443 are routed to port 80
  - HTTP requests on port 8080 get a fixed response (customized HTML)

# Target Groups

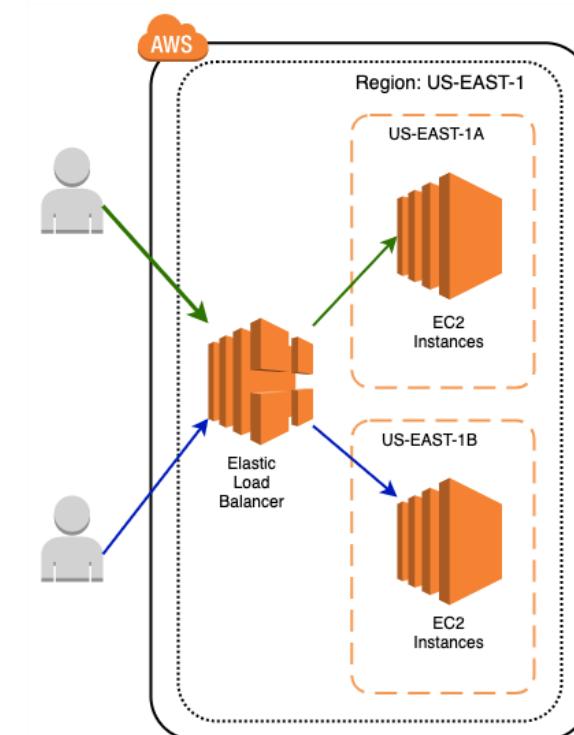
- How to group instances that ALB has to distribute the load between?
  - Create a Target Group
- A target group can be:
  - A set of EC2 instances
  - A lambda function
  - Or a set of IP addresses



# Target Group Configuration - Sticky Session

*Enable sticky user sessions*

- Send all requests from one user to the same instance
- Implemented using a cookie
- Supported by ALB and CLB



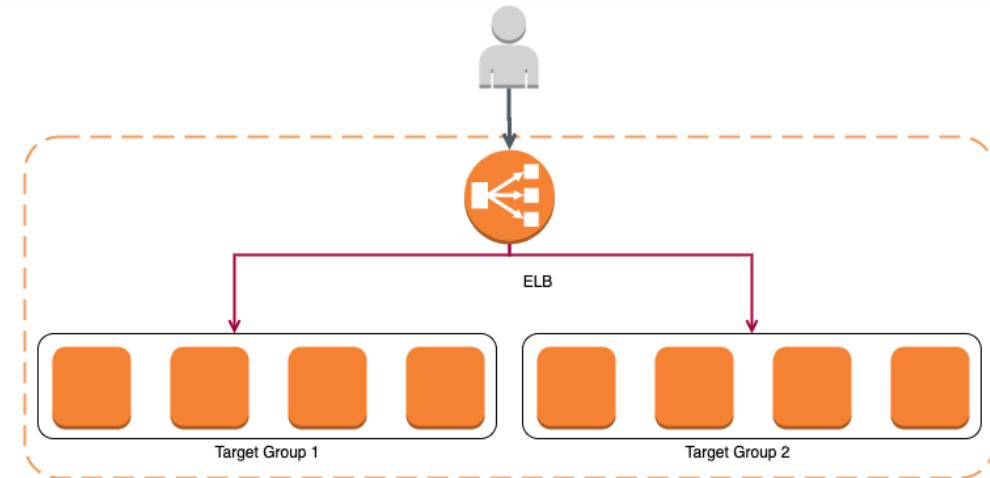
# Target Group Configuration - Deregistration delay

*How long should ELB wait before de-registering a target?*

- Load balancer stops routing new requests to a target when you unregister it
- What about requests that are **already in progress** with that target?
- This setting ensures that load balancer gives **in-flight requests** a chance to complete execution
- 0 to 3600 seconds (default 300 seconds)
- Also called Connection Draining

# Microservices architectures - Multiple Target Group(s)

- Microservices architectures have 1000s of microservices
  - <http://www.xyz.com/microservice-a>
  - <http://www.xyz.com/microservice-b>
- Should we create multiple ALBs?
- **Nope.** One ALB can support multiple microservices!
- Create separate target group for each microservices
- (Remember) Classic Load Balancer, **does NOT** support multiple target groups.



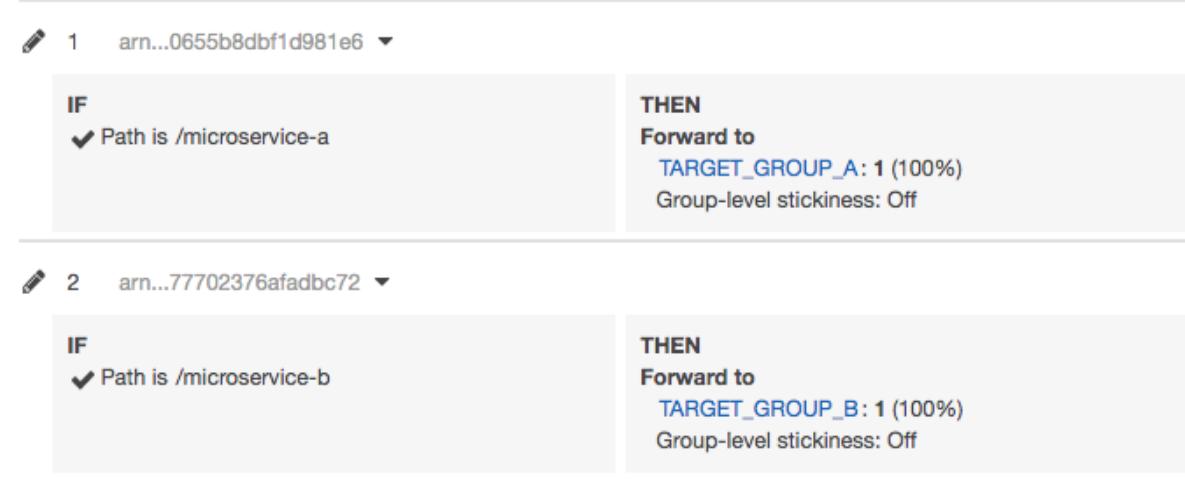
# Listener Rules

The screenshot shows the AWS Lambda Listener Rules configuration interface. It displays two rules:

- Rule 1:** ARN: arn:aws:lambda:us-east-1:0655b8dbf1d981e6. IF Path is /microservice-a, THEN Forward to TARGET\_GROUP\_A: 1 (100%). Group-level stickiness: Off.
- Rule 2:** ARN: arn:aws:lambda:us-east-1:77702376afadbc72. IF Path is /microservice-b, THEN Forward to TARGET\_GROUP\_B: 1 (100%). Group-level stickiness: Off.

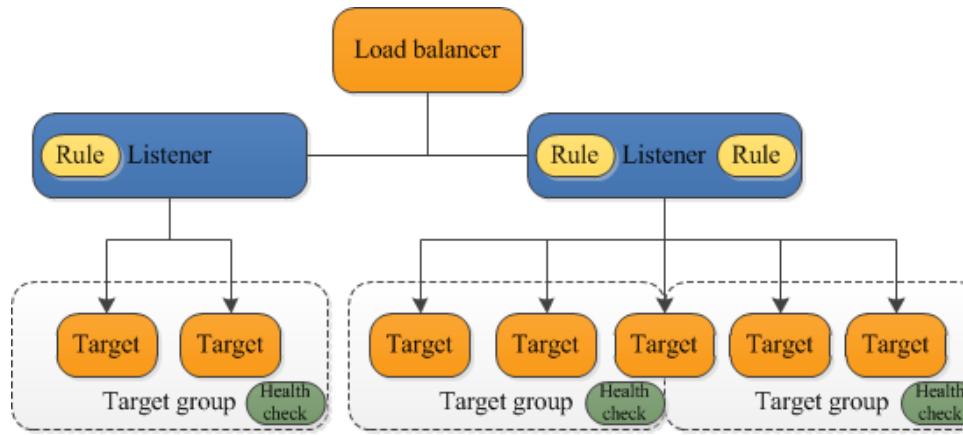
- How do I identify which request should be sent to which target group?
- Configure multiple listener rules for the same listener
- Rules are executed in the order they are configured.
- Default Rule is executed last.

# Listener Rules - Possibilities



- Based on **path** - [in28minutes.com/a](http://in28minutes.com/a) to target group A and [in28minutes.com/b](http://in28minutes.com/b) to target group B
- Based on **Host** - [a.in28minutes.com](http://a.in28minutes.com) to target group A and [b.in28minutes.com](http://b.in28minutes.com) to target group B
- Based on **HTTP headers** (Authorization header) and methods (POST, GET, etc)
- Based on **Query Strings** (/microservice?target=a, /microservice?target=b)
- Based on **IP Address** - all requests from a range of IP address to target group A. Others to target group B

# Architecture Summary

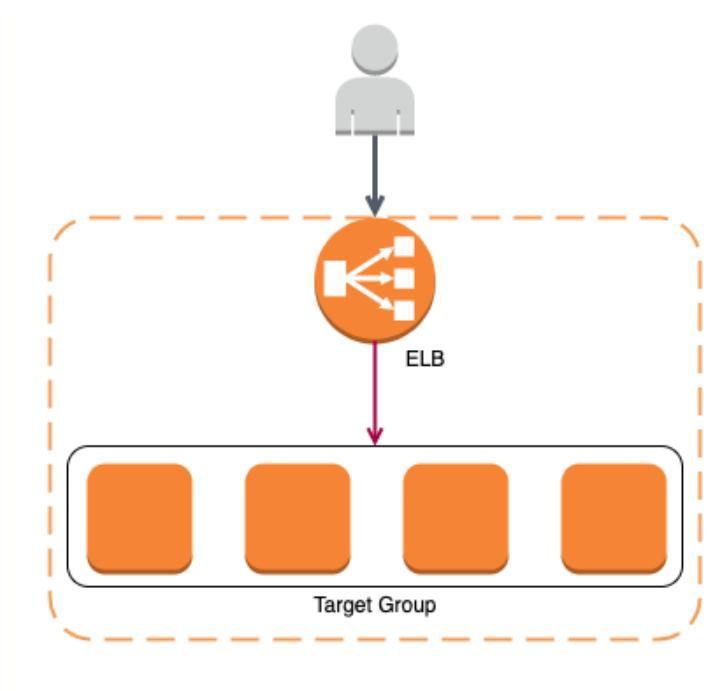


[https://docs.amazonaws.cn/en\\_us/elasticloadbalancing/latest/application/introduction.html](https://docs.amazonaws.cn/en_us/elasticloadbalancing/latest/application/introduction.html)

- Highly decoupled architecture
- Load balancer can have multiple listeners (protocol + port combinations).
- Each listener can have multiple rules each routing to a target group based on request content.
- A target can be part of multiple target groups.

# Introducing Auto Scaling Groups

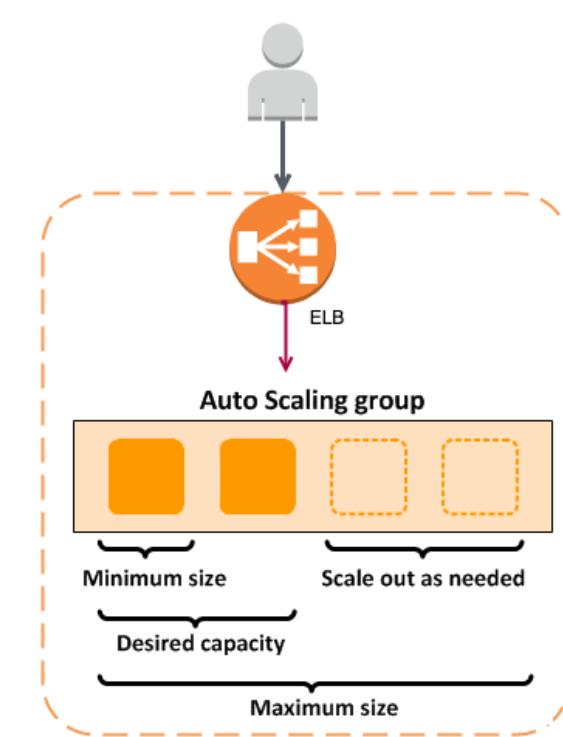
In 28  
Minutes



- Target Groups are configured with a static set of instances. How do you scale out and scale in **automatically**?
  - Configure a Auto Scaling Group

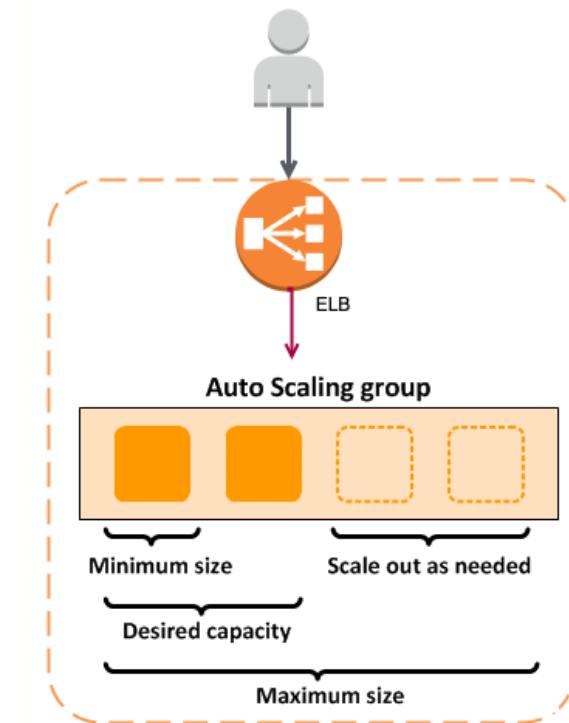
# Auto Scaling Groups

- Auto Scaling Group responsibilities:
  - **Maintain** configured number of instances (using periodic health checks)
    - If an instance goes down, ASG launches replacement instance
  - **Auto scale** to adjust to load (scale-in and scale-out based on auto scaling policies)
- ASG can launch On-Demand Instances, Spot Instances, or both
  - **Best Practice:** Use Launch Template
- An ELB can distribute load to **active instances** as ASG expands and contracts based on the load
- **DEMO:** Creating Auto Scaling Groups



# Auto Scaling Components

- **Launch Configuration/Template**
  - EC2 instance size and AMI
- **Auto Scaling Group**
  - Reference to Launch Configuration/Template
  - Min, max and desired size of ASG
  - EC2 health checks by default. Optionally enable ELB health checks.
  - **Auto Scaling Policies**
    - When and How to execute scaling?



# Auto Scaling Group - Use Cases

In 28  
Minutes



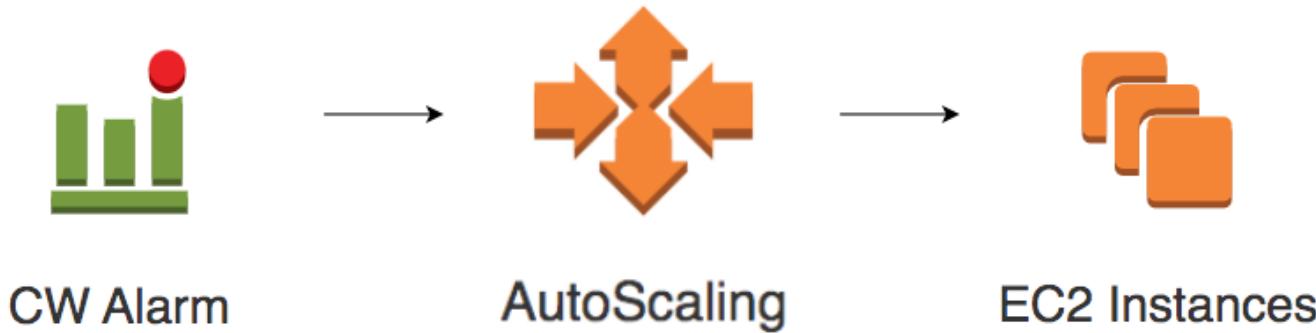
ASG Use case	Description	More details
Maintain current instance levels at all times	min = max = desired = CONSTANT When an instance becomes unhealthy, it is replaced.	Constant load
Scale manually	Change desired capacity as needed	You need complete control over scaling
Scale based on a schedule	Schedule a date and time for scaling up and down.	Batch programs with regular schedules
Scale based on demand (Dynamic/Automatic Scaling)	Create scaling policy (what to monitor?) and scaling action (what action?)	Unpredictable load

# Dynamic Scaling Policy Types



Scaling Policy	Example(s)	Description
<b>Target tracking scaling</b>	Maintain CPU Utilization at 70%.	Modify current capacity based on a target value for a specific metric.
<b>Simple scaling</b>	+5 if CPU utilization > 80% -3 if CPU utilization < 60%	Waits for cooldown period before triggering additional actions.
<b>Step scaling</b>	+1 if CPU utilization between 70% and 80% +3 if CPU utilization between 80% and 100% Similar settings for scale down	Warm up time can be configured for each instance

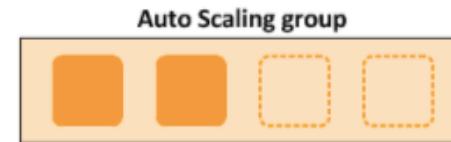
# Scaling Policies - Background



- Two parts:
  - CloudWatch alarm (Is CPU utilization  $>80\%$ ? or  $< 60\%$ ).
  - Scaling action (+5 EC2 instances or -3 EC2 instances)

# Auto Scaling - Scenarios

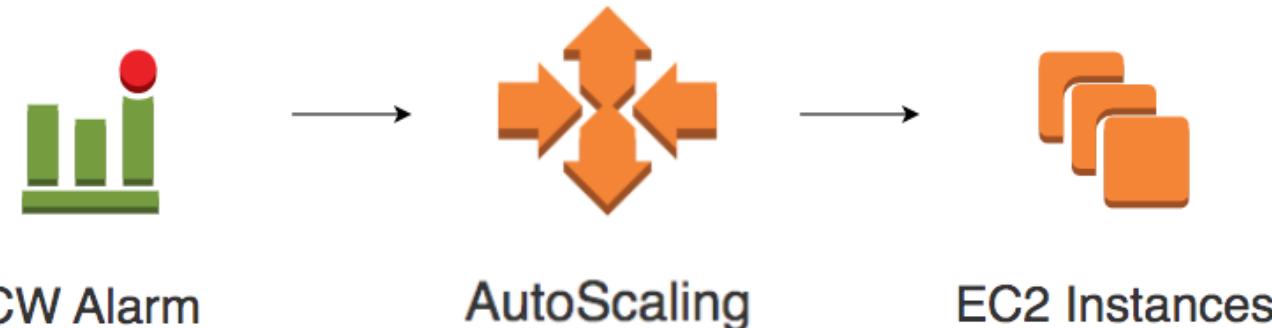
In 28  
Minutes



Scenario	Solution
<b>Change instance type or size of ASG instances</b>	Launch configuration or Launch template cannot be edited. Create a new version and ensure that the ASG is using the new version. Terminate instances in small groups.
<b>Roll out a new security patch (new AMI) to ASG instances</b>	Same as above.
<b>Perform actions before an instance is added or removed</b>	Create a Lifecycle Hook. You can configure CloudWatch to trigger actions based on it.

# Auto Scaling - Scenarios

In 28  
Minutes



Scenario	Solution
Which instance in an ASG is terminated first when a scale-in happens?	(Default Termination Policy) Within constraints, goal is to distribute instances evenly across available AZs. Next priority is to terminate older instances.
Preventing frequent scale up and down	Adjust cooldown period to suit your need (default - 300 seconds). Align CloudWatch monitoring interval
I would want to protect newly launched instances from scale-in	Enable instance scale-in protection

# Network Load Balancer

- Functions at the **Transport Layer** - Layer 4 (Protocols TCP, TLS and UDP)
- For **high performance** use cases (millions of requests per second)
- Can be assigned a **Static IP/Elastic IP**
- Can load balance between:
  - EC2 instances
  - Containerized applications (Amazon ECS)
  - Web applications (using IP addresses)
- Demo

## Elastic Load Balancer

- Distribute traffic across EC2 instances in one or more AZs in a single region
- **Managed Service** - highly available, Auto scales, public or private

## Classic Load Balancer

- Layer 4(TCP/TLS) and Layer 7(HTTP/HTTPS)
- **Old.** Not Recommended by AWS

## Network Load Balancer

- Layer 4(TCP/TLS and UDP)
- **Very high performance usecases**
- Can be assigned a Static IP/Elastic IP

## Application Load Balancer

- Layer 7(HTTP/HTTPS)
- Supports **advanced routing approaches** (path, host, http headers, query strings and origin IP addresses)
- Load balance between EC2 instances, containers, IP addresses and lambdas

## Concepts

- Each Load Balancer has one or more **listeners** (different protocol or port) listening for connection requests from the client
- **Target group** is a group representing the targets (ex: EC2 instances)
- One ALB or NLB can support multiple microservices (multiple target groups)!

## Concepts

- **Auto Scaling Group** - Maintain configured number of instances (using periodic health checks). Auto scale to adjust to load.
- **Dynamic Scaling Policies** - Target tracking scaling, Simple scaling and Step scaling.
- **CloudWatch alarms** track the metric (Is CPU utilization >80%? or < 60%) and trigger the auto scaling action (+5 EC2 instances or -3 EC2 instances)



# Serverless Fundamentals - Lambda and API Gateway



AWS Lambda



API Gateway



DynamoDB

- What are the things we think about when we develop an application?
  - Where do we deploy the application?
  - What kind of server? What OS?
  - How do we take care of scaling the application?
  - How do we ensure that it is always available?
- **What if we do not need to worry about servers and focus on building our application?**
- Enter Serverless

- Remember: **Serverless does NOT mean "No Servers"**
- **Serverless for me:**
  - You don't worry about infrastructure
  - Flexible scaling
  - Automated high availability
  - Pay for use:
    - You don't have to provision servers or capacity!
- **You focus on code** and the cloud managed service takes care of all that is needed to scale your code to serve millions of requests!



AWS Lambda



Lambda Fn

- Write and Scale Your Business Logic
  - Write your business logic in Node.js (JavaScript), Java, Python, Go, C# and more..
  - Don't worry about servers or scaling or availability (only worry about your code)
- Pay for Use
  - Number of requests
  - Duration of requests
  - Memory

# AWS Lambda Function

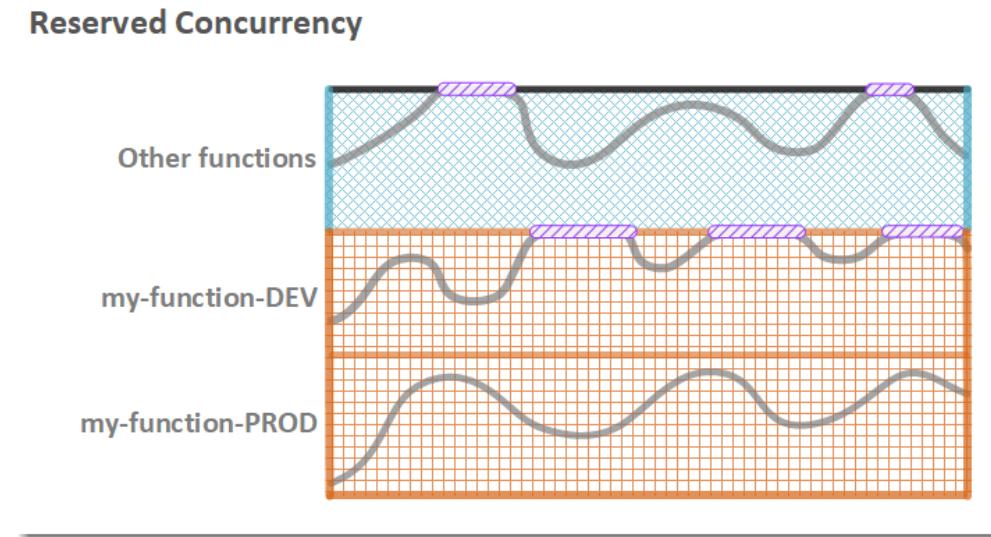
- Stateless - store data to Amazon S3 or Amazon DynamoDB
- 500MB of non-persistent disk space (/tmp directory)
- Allocate memory in 64MB increments from 128MB to 3GB
  - Lambda cost increases with memory
  - CPU Power increases with memory allocated
  - Inexpensive - <https://aws.amazon.com/lambda/pricing/>
    - Free tier - 1M free requests per month
- Monitor function executions through Amazon CloudWatch
- Maximum allowed time for lambda execution is 900 seconds (default - 3 seconds)
- Integrates with AWS X-Ray(tracing), AWS CloudWatch (monitoring and logs)



AWS Lambda

# AWS Lambda Concurrency - Reserved Concurrency

- **Function concurrency** - no of Lambda function instances serving requests (at a given time)
- How to control Function concurrency?
  - Regional quota is shared by all functions in a Region
    - Default 1,000 (Raise by creating support request)
  - How to ensure that a critical lambda function can always run?
    - Use **Reserved Concurrency**
    - Example: my-function-PROD & my-function-DEV have Reserved Concurrency configured
    - Other functions can use remaining concurrency from regional quota



<https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html>

# AWS Lambda Execution Context

```
const AWS = require('aws-sdk');
const dynamo = new AWS.DynamoDB.DocumentClient();

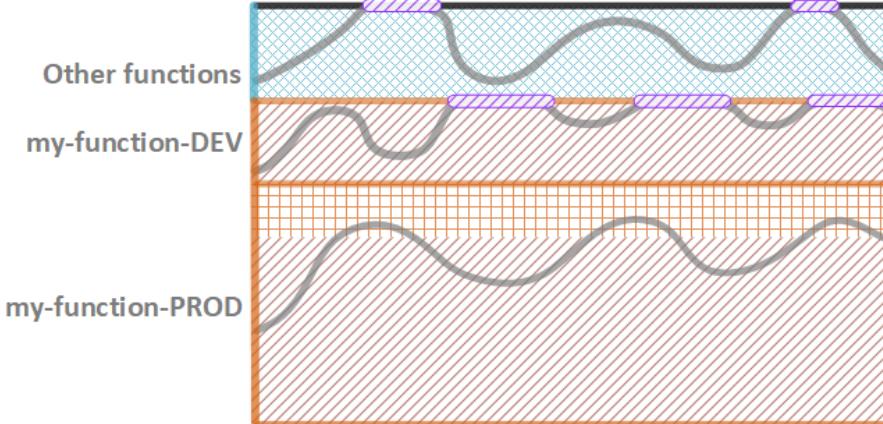
exports.handler = async (event) => {
    //Other logic
}
```

- **Execution Context** - Temporary runtime environment created to execute lambda functions
  - Lambda tries to reuse execution context when possible
  - When Lambda reuses execution context, objects declared outside handler functions remain initialized (AWS and dynamo in above example)
  - Each execution context has /tmp directory with 512 MB disk space. This cache is reused across invocations using same execution context.
- **Cold Start** is a common problem for the first request to a Lambda function (and subsequent requests involving creation of new execution contexts)

# AWS Lambda Concurrency - Provisioned Concurrency

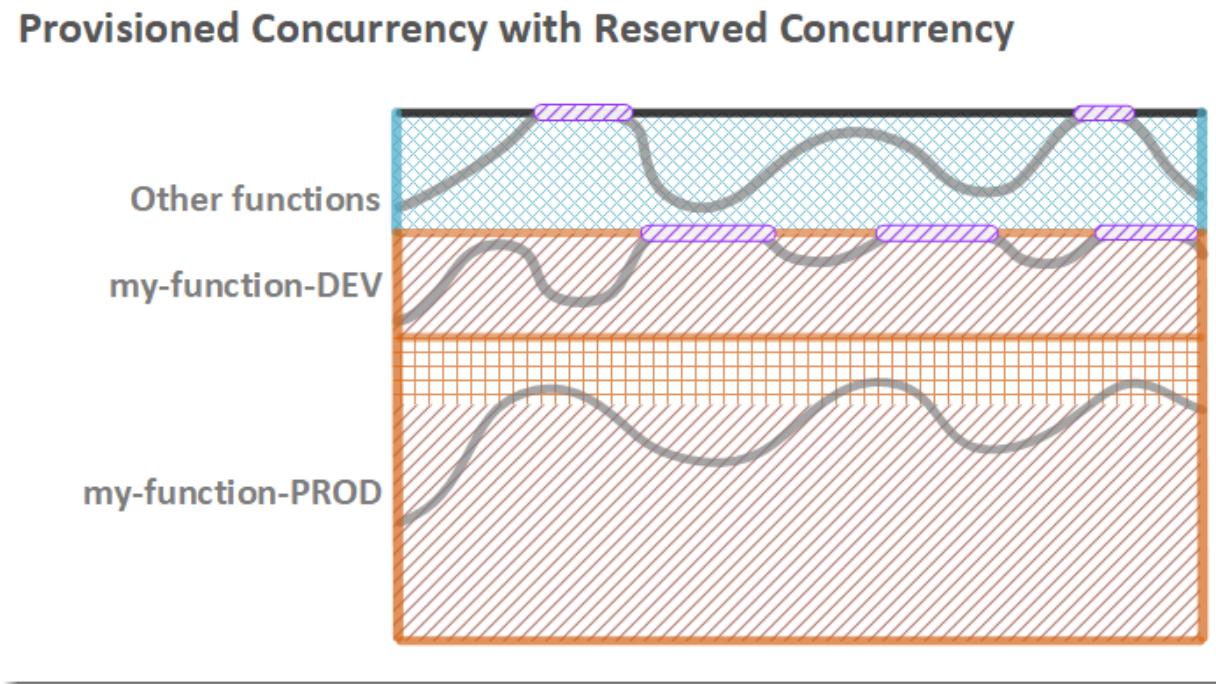
- Initialization of Lambda function takes time
- How to get a consistent performance from Lambda Functions?
  - Use **Provisioned Concurrency**
  - Provisioned concurrency runs continually (More expensive)
  - Provisioned concurrency can be configured on a Lambda function version or an alias
  - In the example, my-function-PROD and my-function-DEV are configured with Provisioned concurrency

Provisioned Concurrency with Reserved Concurrency



<https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html>

# AWS Lambda Concurrency - Throttling



<https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html>

- What happens if more than allowed requests are made?
  - Throttling error (429 status code)

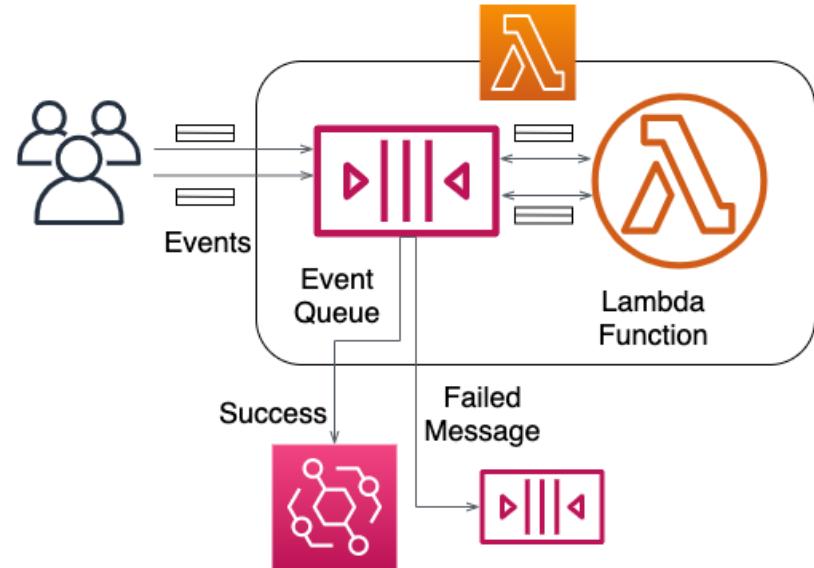
# AWS Lambda - Synchronous Invocation

```
aws lambda invoke --function-name my-function --payload '{ "key": "value" }' response.json
{
    "ExecutedVersion": "$LATEST",
    "StatusCode": 200
}
```

- Lambda runs the function and waits for response
- Lambda returns the response with additional data such as the version of the function that was executed
- Sample services using synchronous invocation
  - AWS API Gateway
  - Amazon CloudFront
  - Amazon Lex

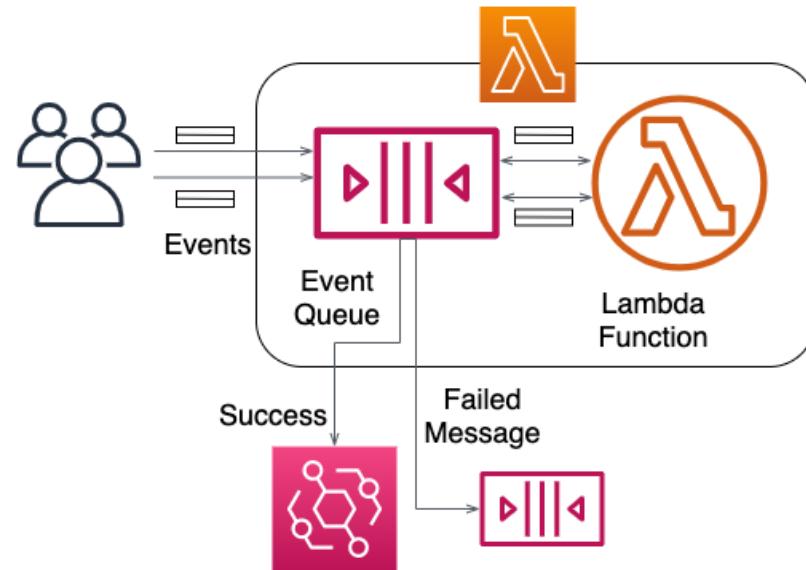
# AWS Lambda - Asynchronous Invocation (Events)

- When using asynchronous invocation (**--invocation-type Event**), AWS services do NOT wait for a response from Lambda function
  - Example: Processing events from Amazon S3, Amazon SNS
  - Lambda places the event on an **event queue**
  - On successful execution, an invocation record (JSON with request and response details) can be sent other AWS services (SQS queue, SNS topic or another Lambda function)



# AWS Lambda - Asynchronous Invocation - Errors

- Lambda retries failed events two more times
- If an event is throttled, Lambda retries upto 6 hours (with exponential backoff)
- Failed events can be sent to a Dead Letter Queue (SQS queue or SNS topic)
  - ONLY request details are sent
- You can configure:
  - Maximum age of event (default - 6 hours)
  - Retry attempts (default - 2)
  - Dead-letter queue service (default - none)



# Lambda Request Context

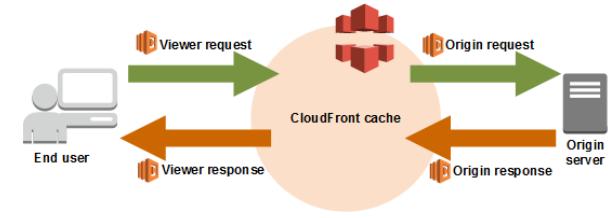
```
exports.handler = async (event, context) => {
    console.log(context);
}
```

- Context object provides information about
  - Lambda function invocation
    - awsRequestId (unique identifier)
    - identity > cognitoidentityId, cognitoidentityPoolId (Which Amazon Cognito identity?)
  - Lambda function & Execution Environment
    - functionName, functionVersion, invokedFunctionArn
    - memoryLimitInMB, logGroupName, logStreamName

# AWS Lambda@Edge

In 28  
Minutes

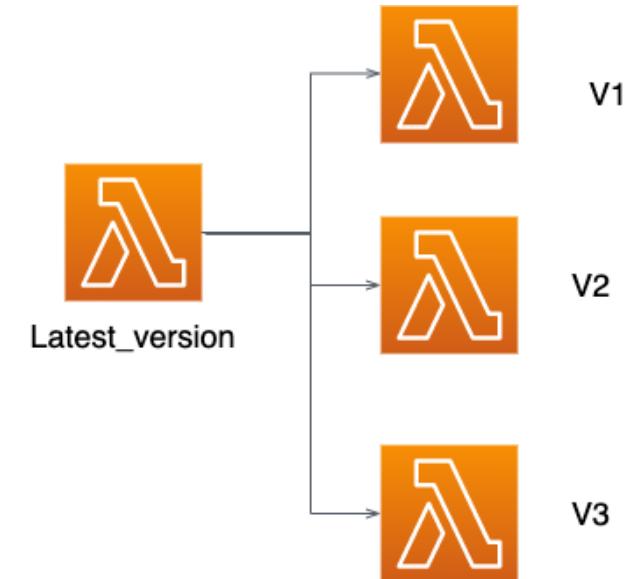
- Run Lambda functions to customize CloudFront content
  - (RESTRICTION) ONLY Python or Node JS supported
- Lambda functions can be run at different points in processing a request in CloudFront :
  - After CF receives a request from a viewer (Viewer Request)
  - Before CF forwards the request to Origin (Origin Request)
  - After CF receives response from Origin (Origin Response)
  - Before sending response to the viewer (Viewer Response)
- Use Cases:
  - **A/B Testing** - URL rewrite to different versions of a site
  - **Multi device support** - Based on User-Agent header, send pictures of different resolution
  - **Generate new HTTP responses** - redirect unauthenticated users to Login page



<https://docs.aws.amazon.com/lambda/latest/dg/lambdaledge.html>

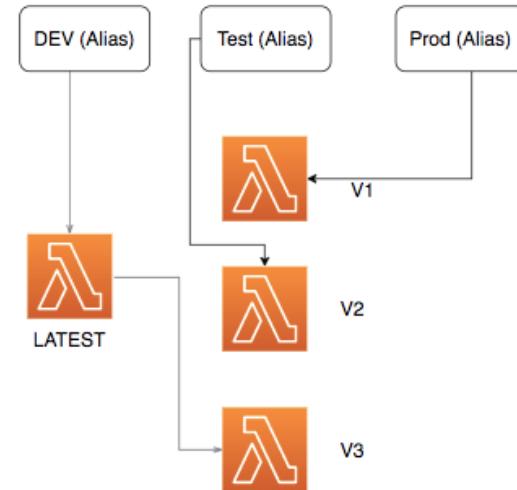
# AWS Lambda - Versioning

- How to move a tested lambda function to production and avoid anyone changing it accidentally?
  - Create a **version**
  - Creates an immutable copy of your lambda function
- A version includes:
  - Function code and all the dependencies
  - The Lambda runtime
  - All function settings including environment variables
  - Unique ARN for the version
- (NOTE) `$LATEST` points to latest version



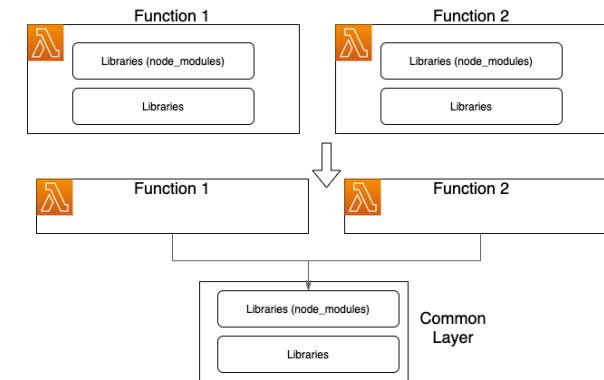
# AWS Lambda - Alias

- How to ensure that consumers of lambda functions are not affected when you release new versions?
  - Use an Alias
- **Alias** - pointer to specific version of Lambda function
- Example:
  - Currently : Dev => latest version, Test => V2, Prod => V1
  - After V2 is tested: Switch Alias Prod => V2
  - Consumers can always refer to the Prod alias and use the fully tested version
- Features:
  - Can be used to define permissions in resource-based policies
  - Alias routing configuration can be used to send a portion of traffic to a second function version (Blue/Green Deployment)



# AWS Lambda Layers

- Lambda code is typically dependent on other libraries
- How to share libraries among Lambda functions?
  - Create Layers
- Layer - ZIP with libraries & other dependencies
  - (ADVANTAGE) Keep deployment package small
  - (ADVANTAGE) Develop function code in the Lambda console (Package size < 3 MB)
  - (CONSTRAINT) Max 5 Layers
- Layers are extracted to the /opt directory and made available to your Lambda functions
- Use AWS Serverless Application Model (AWS SAM) to automate creation and mapping of layers



# AWS Lambda Layers using SAM

```
Transform: 'AWS::Serverless-2016-10-31'
Resources:
  function:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs12.x
      Policies:
        - AWSLambdaBasicExecutionRole
        - AWSLambdaReadOnlyAccess
        - AWSXrayWriteOnlyAccess
      Layers:
        - !Ref libs
  libs:
    Type: AWS::Serverless::LayerVersion
    Properties:
      LayerName: blank-nodejs-lib
      ContentUri: lib/
      CompatibleRuntimes:
        - nodejs12.x
```

# Lambda Best Practices - Recommended by AWS

- Take advantage of **execution context reuse** to improve the performance of your function
  - Initialize SDK clients and database connections outside of the function handler
  - Cache static assets locally in the /tmp directory
- Use **environment variables** to pass operational parameters
- Minimize your deployment package size to its runtime necessities
- **Avoid using recursive code** (Save \$\$\$)
- Reduce the time it takes Lambda to unpack deployment packages authored in Java by putting your dependency .jar files in a separate /lib directory.
  - This is faster than putting all your function's code in a single jar



AWS Lambda

# AWS Lambda - Scenario Questions

In 28  
Minutes

Scenario	Solution
<b>Does Lambda scale up or out when it receives multiple requests?</b>	Lambda scales out - NOT up. No of instances are increased
<b>How do you enable logging in Lambda functions?</b>	Lambda make logging very easy Call the appropriate method ( <code>System.out</code> or <code>console.log</code> ) Lambda automatically sends it to CloudWatch logs Make sure that Lambda function execution role has the right permissions to write to CloudWatch logs
<b>How do you enable tracing in Lambda functions?</b>	Lambda makes tracing very easy. 1. Give Permissions to Execution Role 2. Enable Tracing with X-Ray
<b>How can you make a Lambda function run faster?</b>	Increase memory

# AWS Lambda - Scenario Questions - 2

In 28  
Minutes

Scenario	Solution
I would want to create a temporary file of 100 MB in a Lambda. Where can I store the file?	Use /tmp directory
Send request headers with multiple values as an array from Application Load Balancer to a Lambda Function	Enable Multi-value headers on ALB
You are creating thumbnails for images in Lambda functions based on event notifications from an S3 bucket. You are creating a number of versions of your Lambda function. How do you avoid configuring the Lambda function version in S3 event notification every time there is a new version?	Create an Alias for your Lambda function and use it from the S3 event notification
How can you increase the CPU available to a Lambda function?	Increase available memory

# API GATEWAY

# REST API Challenges

In 28  
Minutes



- Most applications today are built around REST API:
  - Resources (/todos, /todos/{id}, etc.)
  - Actions - HTTP Methods - GET, PUT, POST, DELETE etc.
- Management of REST API is not easy:
  - You've to take care of authentication and authorization
  - You've to be able to set limits (rate limiting, quotas) for your API consumers
  - You've to take care of implementing multiple versions of your API
  - You would want to implement monitoring, caching and a lot of other features..

# Amazon API Gateway

In 28  
Minutes



- How about a **fully managed service** with auto scaling that can act as a "front door" to your APIs?
- Welcome "**Amazon API Gateway**"

# Amazon API Gateway

In 28  
Minutes



- **"publish, maintain, monitor, and secure APIs at any scale"**
- Integrates with AWS Lambda or any web application
- Supports HTTP(S) and WebSockets (two way communication - chat apps and streaming dashboards)
- Serverless. Pay for use (API calls and connection duration)

# API Gateway - API Types

- REST API
  - feature-rich RESTful API
- HTTP API
  - Also used to build RESTful API
  - Newer approach
- WebSocket API
  - Persistent connections with clients
  - Allows full-duplex communication
- Names are little confusing



API Gateway

# API Gateway RESTful API approaches

In 28  
Minutes



## REST API

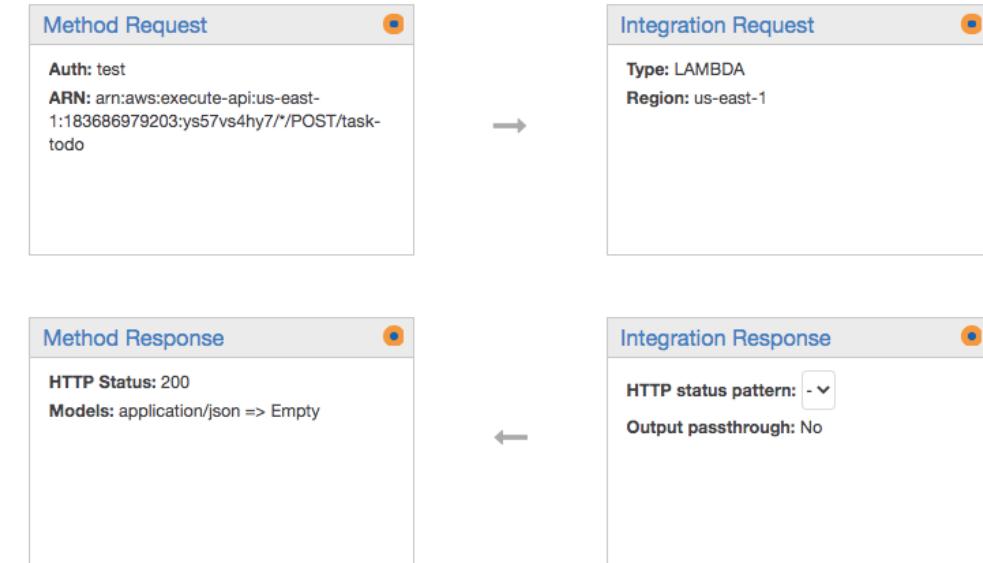
- Fully Featured (API caching, Request/Response Validations, Test invocations)
- Custom Request/Response Transformations
- Better Integration with AWS Services (AWS X-Ray, AWS WAF etc)

## HTTP API

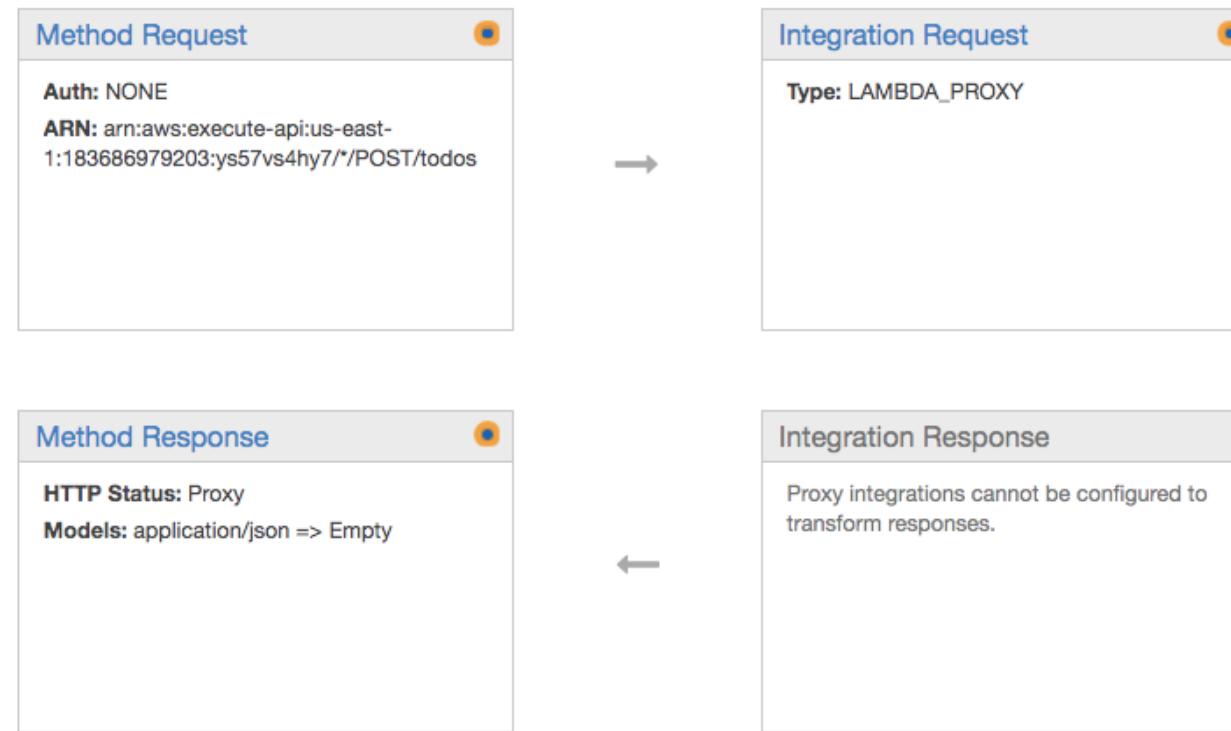
- Newer, Simpler, Cheaper and Low Latency
- Automatic Deployments

# REST API Gateway - Custom Integration (Default)

- Integrations define request/response transformation to/from lambda
- The default is Custom Integration
- Request can be transformed by configuring Mapping Template in Integration Request
- Response can be transformed by configuring Mapping Template in Integration Response



# REST API Gateway - Proxy Integration



- How about defining a standard transformation?

# REST API Gateway - Proxy Integration - Request

## Request to API Gateway

```
//Headers: header1:header-value
//queryString: ?queryparam=queryparamvalue
{
    "message": "Welcome"
}
```

## Standard event sent to Lambda Function

```
{
  resource: '/todos',
  path: '/todos',
  httpMethod: 'POST',
  headers: {"header1": "header-value"},
  multiValueHeaders: {"header1": ["header-value"]},
  queryStringParameters: {"queryparam": "queryparamvalue"},
  multiValueQueryStringParameters: {"queryparam": ["queryparamvalue"]},
  pathParameters: null,
  stageVariables: null,
  requestContext: {},
  body: '{\n    "message" : "Welcome"\n}',
  isBase64Encoded: false
},
```

# REST API Gateway - Proxy Integration - Response

## Response from Lambda Function

```
{  
  statusCode: 200,                      // a valid HTTP status code  
  headers: {  
    custom-header: "xyz"                // any API-specific custom header  
  },  
  body: "{\"message\": \"Welcome\"}"    // a JSON string.  
}
```

## Response from API Gateway

Status: 200

Latency: 1401 ms

Response Body

```
{  
  "message": "Welcome"  
}
```

Response Headers

```
{"custom-header": "xyz", "X-Amzn-Trace-Id": "Root=1-5f59d83b-3b4dd22a1  
6e7f84a7c495ac4;Sampled=0"}
```

# HTTP API - API Gateway

- REST API - API Gateway has a lot of features very few AWS customers made use of
- REST API - API Gateway is a little complex to setup (transformations etc)
- How about creating a simpler API Gateway?
- Enter "HTTP API"
  - The naming is confusing
  - Newer, Cheaper and Low Latency
  - Simpler
    - Lesser features
    - Easier to setup
    - Example: Makes OAuth Authentication simple



API Gateway

# HTTP API - API Gateway - Payload

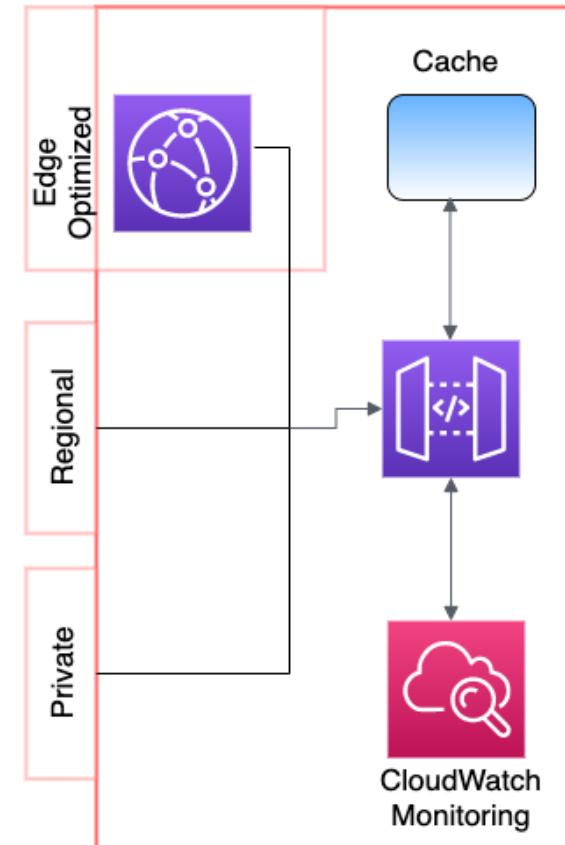
- Two Versions - 1.0 and 2.0
- (Recommendation) Use 1.0 for migration from REST API and 2.0 for newer APIs
- Request Structure
  - Almost same as REST API - Proxy Integration
  - 2.0 offers support for cookies and has minor changes
- Response Structure
  - Same as REST API - Proxy Integration (with statusCode, body, headers)
  - In addition, 2.0 supports a simple structure:
    - Just return a valid response JSON return {"message": "Welcome"}



API Gateway

# API Gateway - Endpoint Types

- Edge Optimized (default)
  - Recommended for geographically distributed clients
  - API requests are routed to the nearest CloudFront Edge Location
- Regional
  - Recommended for clients in a single region
- Private
  - Can only be accessed from your VPC using an interface VPC endpoint



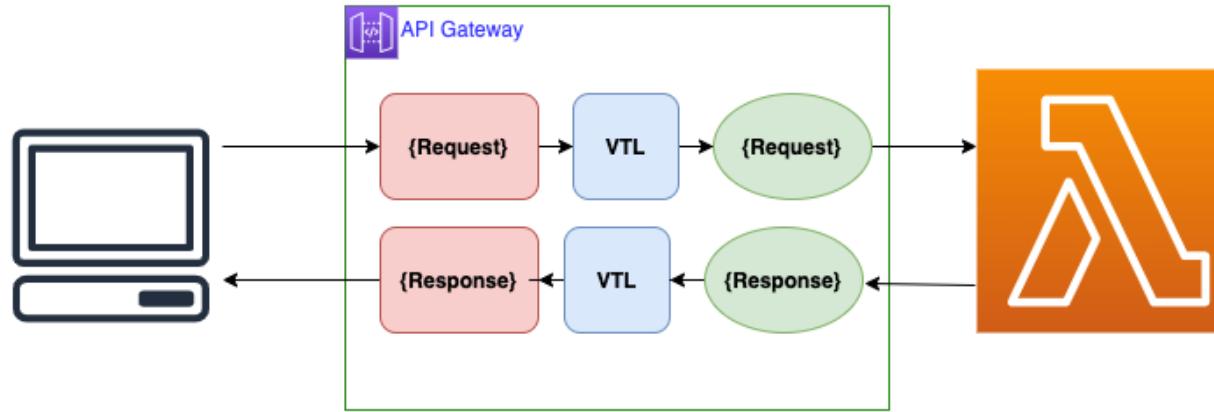
# API Gateway - Integration Types

- API Gateway acts as a front-end for different backend systems
- Supported Integration Types:
  - **Lambda function** - Connect via proxy or direct integration
  - **HTTP** - Connect to an HTTP/HTTPS end point inside or outside of AWS
  - **Mock** - Create a mock backend service
  - **AWS service** - Connect to 100+ service endpoints inside of AWS (DynamoDB, Kinesis etc)
  - **VPC Link** - Connect to AWS resources inside a VPC



API Gateway

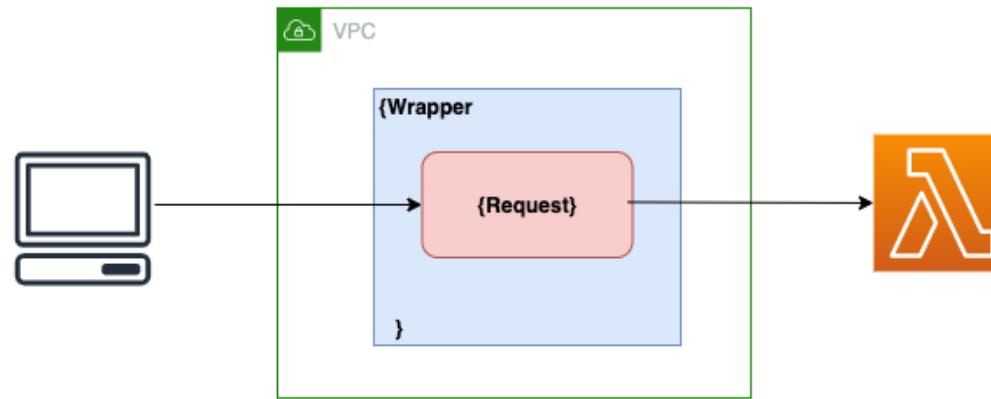
# REST API Gateway - Lambda Integration - Custom Integration



- Configure mappings(using VTL) to transform request and response

# REST API Gateway - Lambda Integration - Proxy Integration

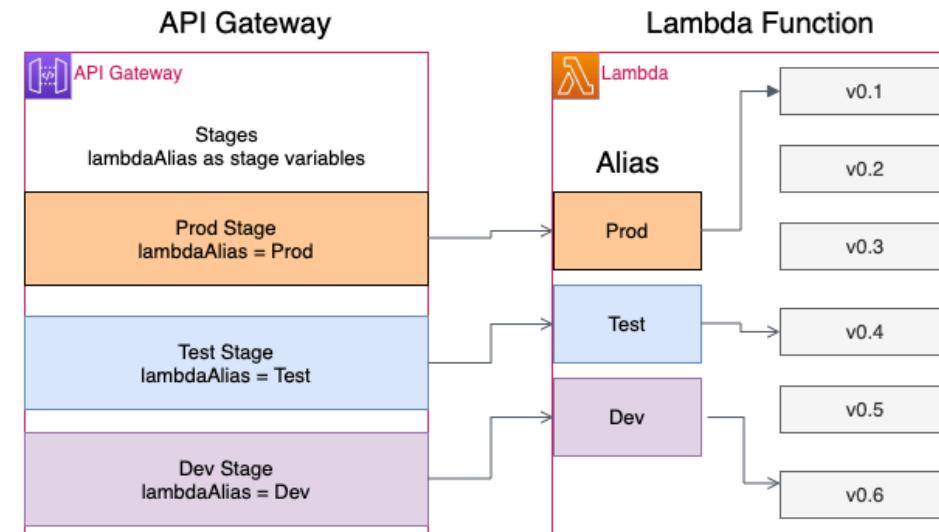
In 28  
Minutes



- Predefined structure for request and response transformations

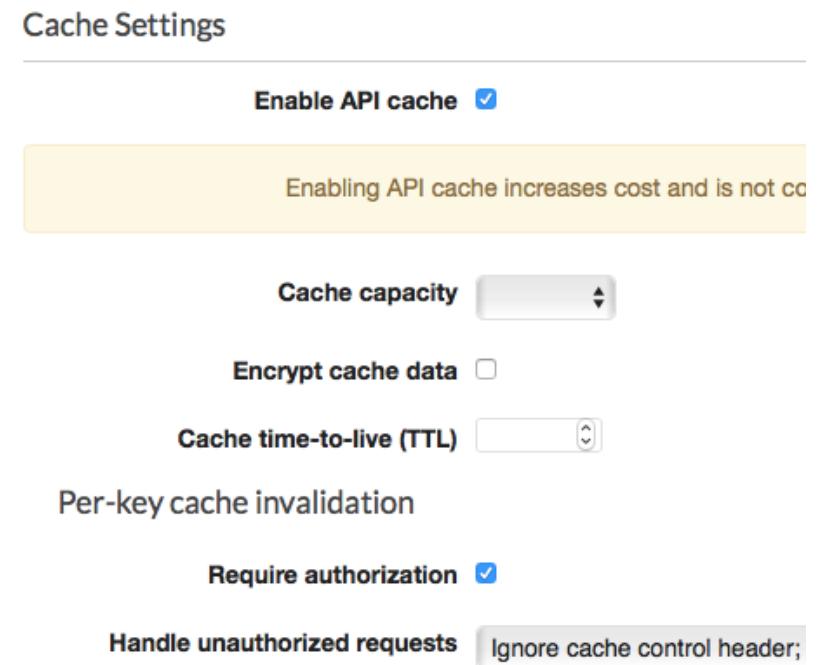
# API Gateway - Deployment Stages

- How do we deploy API Gateway to different environments?
  - Create different **Stages**
- You can create more than one stage
  - Dev, Test, UAT, Prod etc.
- Use **Stage variables** for changing configuration values for different environments:
  - Example: Connect to different Lambda aliases in different stages



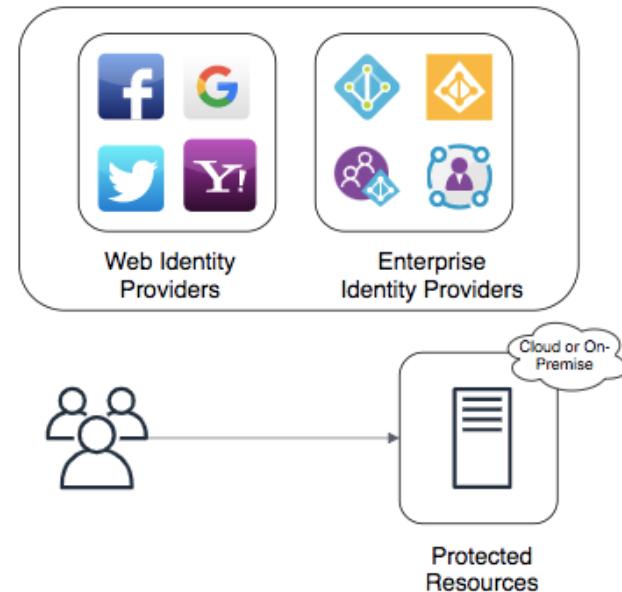
# API Gateway - Caching

- **Caching** helps you provide quick responses (low latency) and minimize load on the backend systems (save \$\$\$)
- Supported for API Gateway - REST API
- How to enable Caching?
  - Enable API cache for the specific stage
    - You can override stage settings for specific methods
  - Configure time-to-live (TTL)
    - default - 300 seconds (max - 3600 seconds, TTL=0 to disable caching)
  - Verify CacheHitCount and CacheMissCount metrics in CloudWatch
  - Cache keys can be formed using custom headers, URL paths, and/or query strings



# Identity Federation

- Authenticate users with an external authentication system and provide them access to resources on the cloud
- **Corporate Identity Federation :**
  - Federate with an Enterprise Authentication System
  - SAML (XML Based) is the most popular protocol
- **Web Identity Federation :**
  - Provide access to your application to users based on their Social IDs
  - OpenID (Supported by Facebook, Microsoft, Google etc) is the most popular protocol



# Amazon Cognito

- Want to quickly add a **sign-up page and authentication** for your mobile and web apps?
- Want to **integrate with web identity providers** (example: Google, Facebook, Amazon) and provide a social sign-in?
- Do you want security features such as **multi-factor authentication (MFA)**, phone and email verification?
- Want to create **your own user database** without worrying about scaling or operations?
- Let's go : Amazon Cognito
- (Feature) Sync user data across devices, platforms, and applications



Amazon Cognito

# Amazon Cognito - User Pools

- Do you want to create your own secure and scalable user directory?
- Do you want to create sign-up (or registration) pages?
- Do you want a built-in, customizable web UI to sign in users (with option to social sign-in )?
- Create a user pool
- Cognito User Pool can be integrated with Application Load Balancer and API Gateway



Amazon Cognito

# Amazon Cognito - Identity pools

Cognito    Amazon    Apple    Facebook    Google+    Twitter / Digits    OpenID    **SAML**    Custom

- Identity pools provide **AWS credentials** to grant your users access to other AWS resources
- Connect identity pools with **authentication (identity) providers**:
  - Your own user pool OR
  - Amazon, Apple, Facebook, Google+, Twitter OR
  - OpenID Connect provider OR
  - SAML identity providers (SAML 2.0)
- Configure **multiple** authentication (identity) providers for each identity pool

# Customize Userpool workflow - Important Triggers

Userpool flow	Operation	Description
Authentication Events	Pre Authentication Lambda Trigger	Custom validation - Accept or deny sign-in request
	Post Auth Lambda Trigger	Event logging - Generate custom analytics
	Pre Token Gen. Lambda Trigger	Customize token claims
Sign-Up	Pre Sign-up Lambda Trigger	Custom validation - Accept or deny sign-up request
	Post Confirmation Lambda Trigger	Customize welcome messages. Logging for custom analytics.
User Migration	Migrate User Lambda Trigger	Migrating a user from an existing user directory to user pools

# Amazon Cognito - User Pools vs Identity Pools

Scenario	Solution
Maintain Your Own Registry of Hundreds of Users for a Web Application	User Pool
Maintain Your Own Registry of Thousands of Users for a Mobile Application	User Pool
Create Sign Up Pages or Sign In Pages	User Pool
Create Password Reset Page	User Pool
Guest Access or Anonymous Access	Identity Pool
Support authentication for your mobile/web app without needing to maintain your own users	Identity Pool
Give access to AWS resources based on Social IDs (OpenID/OIDC)	Identity Pool
Give access to AWS resources based on Corporate Directory (SAML)	Identity Pool

# API Gateway - Authorization

- Open - No authentication or authorization
- IAM Permissions - Use IAM Policies and AWS credentials to grant access
- Amazon Cognito authorizer - Connect to Amazon Cognito User Pool (possible to use OAuth authorization)
- Lambda authorizers - Write custom lambda function to validate the bearer token (OAuth or SAML for example), or request parameters



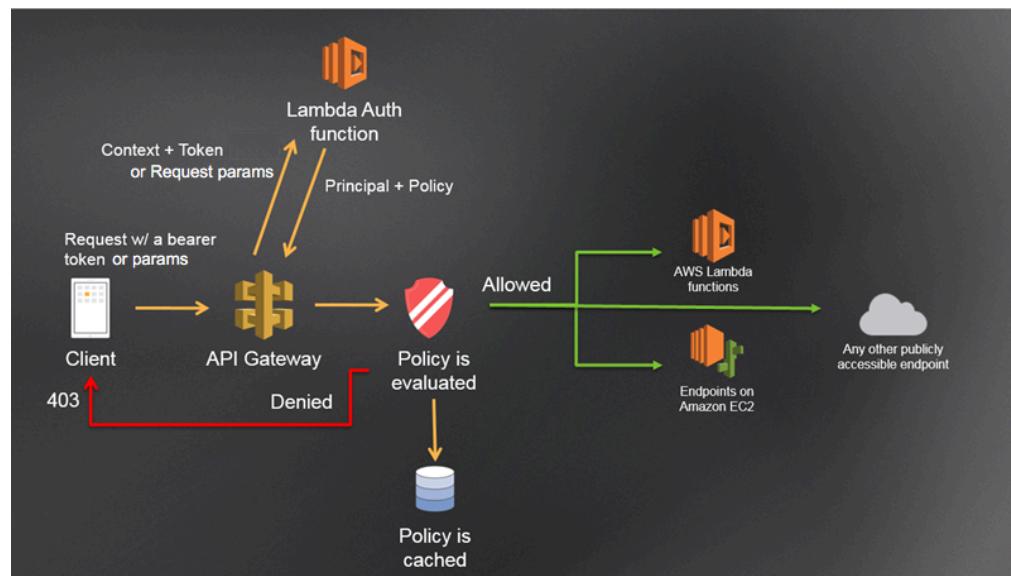
# Authorization - IAM Authorization

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["execute-api:Invoke"],  
            "Resource": [  
                "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets"  
            ]  
        }  
    ]  
}
```

- When using IAM Authorization (authorization type set to AWS\_IAM):
  - API Gateway checks whether the IAM user has the right permissions attached

# Lambda Authorizer

- Use a Lambda function to control access to your API:
  - Input: bearer token (token-based) or request parameters (request parameter-based)
  - Implement custom authorization strategy (call OAuth or SAML provider) in Lambda
  - Output: Object containing at least an IAM policy and a principal identifier
- When API Gateway receives a request:
  - API Gateway calls the authorizer Lambda function
  - Lambda function returns the IAM policy
  - API Gateway evaluates the policy document and grants/denies access



<https://docs.aws.amazon.com/apigateway/latest/developerguide/images/cu-auth-workflow.png>

auth-workflow.png

# Lambda Authorizer - Policy Response Example

```
//Grant Access
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "execute-api:Invoke",
            "Effect": "Allow",
            "Resource": "arn:aws:execute-api:us-east-1:123:7b5/ESTestInvoke-stage/GET/"
        }
    ]
}

//Deny Access
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "execute-api:Invoke",
            "Effect": "Deny",
            "Resource": "arn:aws:execute-api:us-east-1:123:7b5/ESTestInvoke-stage/GET/"
        }
    ]
}
```

# Cognito User Pool Authorizer

- Use an Amazon Cognito user pool to control access to your API
- Configuring a Cognito User Pool Authorizer:
  - Step I: Create a User Pool in Cognito
  - Step II: Configure API Gateway to use the Amazon Cognito user pool as authorizer
- To call an API integrated with user pool:
  - Step I: User signs up for the user pool
  - Step II: User signs in
  - Step III: Call the API method passing the user's identity token in Request Authorization header

Create Authorizer

Name \*

Type \*  Cognito  Lambda

Cognito User Pool \*  us-east-1

Token Source \*  Token Validation

**Create** **Cancel**

# API Gateway - Scenario Questions

In 28  
Minutes

Scenario	Solution
Create separate dev, test, qa and prod environments for API Gateway and Lambda	Create multiple stages for API Gateway. Use Lambda Aliases as Stage Variables to map to the different Lambda versions
Expose API around a backend SOAP web service	Use Mapping Templates to convert JSON to XML
You would need to release a breaking change to an API. However, you would NOT want to impact existing clients of API.	Deploy new version to a new stage
An API Gateway is integrating with a Lambda. What would happen if Lambda take 5 minutes to process the request	Timeout after 30 seconds (Max allowed for API Gateway)
Can an API Gateway client invalidate a cache entry?	Yes. By using header <i>Cache-Control:max-age=0</i> . Policy allowing <i>execute-api:InvalidateCache</i> should be attached with the user.

# API Gateway - Scenario Questions - 2

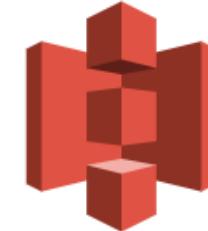
In 28  
Minutes

Scenario	Solution
Create customized plans for your API Consumers - Basic, Premium, Full	Use Usage Plans
You are creating an API to be exposed to existing AWS users in your account. Which Authorization approach would you recommend?	IAM Based Authorization
You want to enable creation of your API specific users. You want to provide them with sign-up, sign-in and password reset features. Which Authorization approach would you recommend?	Cognito User Pool

# Amazon S3 Fundamentals

# Amazon S3 (Simple Storage Service)

- Most popular, very flexible & inexpensive storage service
- Store large objects using a **key-value** approach
- Also called **Object Storage**
- Provides REST API to access and modify objects
- Provides **unlimited storage**:
  - (S3 storage class) 99.99% availability & (11 9's - 99.999999999) durability
  - Objects are replicated in a single region (across multiple AZs)
- **Store all file types** - text, binary, backup & archives:
  - Media files and archives
  - Application packages and logs
  - Backups of your databases or storage devices
  - Staging data during on-premise to cloud database migration



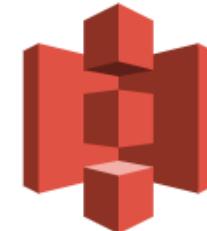
Amazon S3

# Amazon S3 Demo

- DEMO

# Amazon S3 - Objects and Buckets

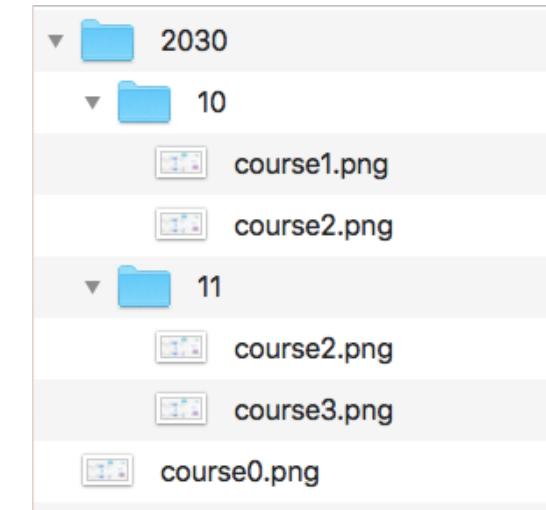
- Amazon S3 is a **global service**. NOT associated with a region.
  - HOWEVER a bucket is created in a specific AWS region
- Objects are stored in buckets
  - Bucket names are **globally unique**
  - Bucket names are used as part of object URLs => Can contain ONLY lower case letters, numbers, hyphens and periods.
  - Unlimited objects in a bucket
- Each object is identified by a **key value pair**
  - Key is **unique** in a bucket
  - Max object size is 5 TB
- (Remember) **No hierarchy** of buckets, sub-buckets or folders



Amazon S3

# Amazon S3 Key Value Example

Key	Value
2030/course0.png	image-binary-content
2030/10/course1.png	image-binary-content
2030/10/course2.png	image-binary-content
2030/11/course2.png	image-binary-content
2030/11/course3.png	image-binary-content



# Amazon S3 Versioning

- Protects against accidental deletion
- Versioning is **optional** and is enabled at bucket level
- You can turn on versioning on a non versioned bucket
  - All old objects will have a version of null
- You cannot turn off versioning on a versioned bucket
  - You can only **suspend** versioning



S3 Bucket

# Amazon S3 Static Website Hosting

- Use S3 to host a static website using a bucket
- Step 1 : Upload website content
- Step 2 : Enable **Static website hosting**
- Step 3 : Disable "Block public access"
- Step 4 : Configure "Bucket policy" to enable public read access

# Resource-based policies - Bucket policies

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PublicRead",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": ["s3:GetObject"],  
            "Resource": ["arn:aws:s3:::mybucket/*"]  
        }  
    ]  
}
```

- Control access to your bucket and objects
- Can grant **cross account** and **public** access

# Amazon S3 - Tags

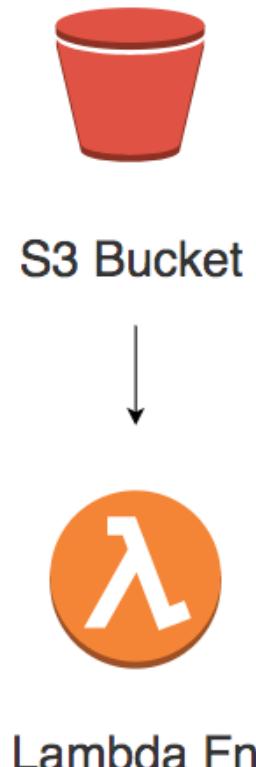
- Tags can be assigned to most AWS resources
- Can be used for **automation, security (policies), cost tracking** etc
- **Key-value pairs** applied to S3 objects:
  - Environment=Dev
  - Classification=Secure
  - Project=A
- Can be used in creating **lifecycle policies**
- Can be updated continuously during the lifetime of an object



S3 Bucket

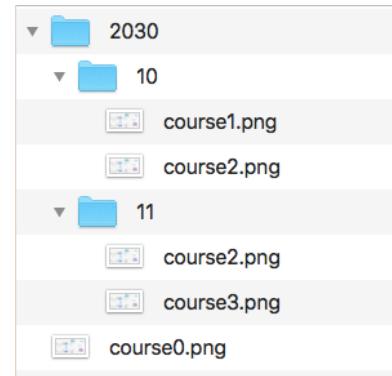
# Amazon S3 Event Notifications

- Configure **notifications** when **certain events** happen in your bucket
- **Event Sources**
  - New object created events
  - Object removal events
  - Reduced Redundancy Storage (RRS) object lost events
  - Replication events
- **Event Destinations**
  - Amazon SNS topic
  - Amazon SQS queue
  - AWS Lambda function



# Amazon S3 - Prefix

- Allows you to search for keys **starting with a certain prefix**
- Searching with prefix **2030/10** returns
  - `2030/10/course1.png` & `2030/10/course2.png`
- URL - `http://s3.amazonaws.com/my-bucket-ranga?prefix=2030/10/`
  - Above URL would work only when public access is allowed
- Supported by REST API, AWS SDK, AWS CLI and AWS Management Console
- Used in IAM and Bucket Policies to restrict access to specific files or group of files



# Bucket ACLs and Object ACLs

- **Bucket/Object ACLs**
  - Access for bucket/object owner
  - Access for other AWS accounts
  - Public access
- **Use object ACLs (object level access)**
  - When bucket owner is not the object owner
  - When you need different permissions for different objects in the same bucket
- **(Remember) Bucket/Object ACLs**
  - CANNOT have conditions while policies can have conditions
  - CANNOT explicitly DENY access
  - CANNOT grant permissions to other individual users
- **(Remember) ACLs are primarily used to grant permissions to public or other AWS accounts**

# Amazon S3 Storage Classes - Introduction

- Different kinds of data can be stored in Amazon S3
  - Media files and archives
  - Application packages and logs
  - Backups of your databases or storage devices
  - Long term archives
- Huge variations in access patterns
- Trade-off between access time and cost
- S3 storage classes help to optimize your costs while meeting access time needs
  - Designed for durability of 99.999999999%(11 9's)



Amazon S3



Amazon Glacier

# Amazon S3 Storage Classes

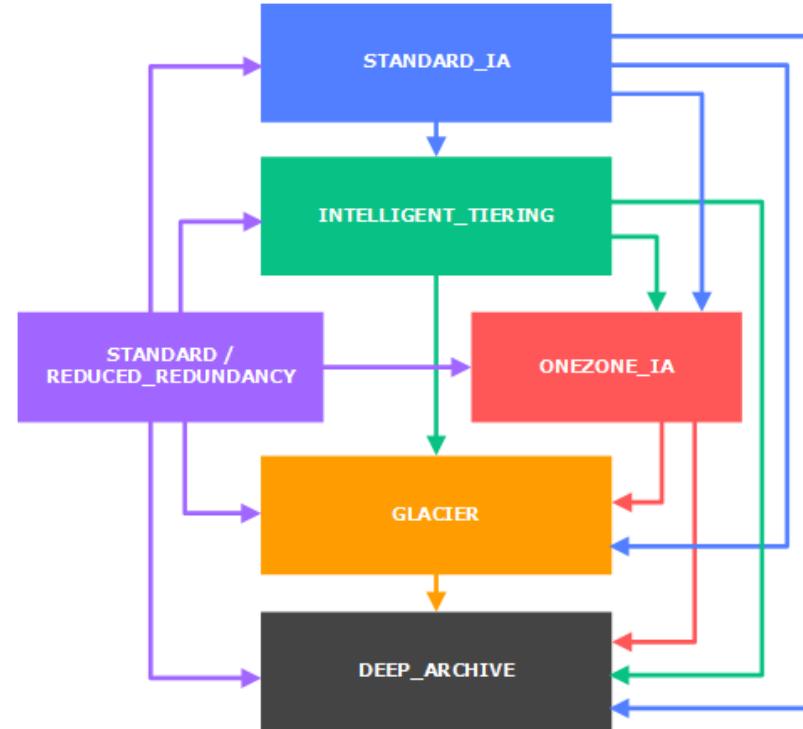
Storage Class	Scenario	AZs
Standard	Frequently accessed data	>=3
Standard-IA	Long-lived, infrequently accessed data (backups for disaster recovery)	>=3
One Zone-IA	Long-lived, infrequently accessed, non-critical data (Easily re-creatable data - thumbnails for images)	1
Intelligent-Tiering	Long-lived data with changing or unknown access patterns	>=3
Glacier	Archive data with retrieval times ranging from minutes to hours	>=3
Glacier Deep Archive	Archive data that rarely, if ever, needs to be accessed with retrieval times in hours	>=3
Reduced Redundancy (Not recommended)	Frequently accessed, non-critical data	>=3

# Amazon S3 Storage Classes - Comparison

Feature	Standard	Intelligent Tiering	Standard IA	One Zone IA	Glacier	Glacier Deep Archive
Availability (Designed)	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability (SLA)	99.9%	99%	99%	99%	99.9%	99.9%
Replication AZs	>=3	>=3	>=3	1	>=3	>=3
First byte: ms (milliseconds)	ms	ms	ms	ms	minutes or hours	few hours
Min object size (for billing)	NA	NA	128KB	128KB	40KB	40KB
Min storage days (for billing)	NA	30	30	30	90	180
Per GB Cost (varies)	\$0.025	varies	\$0.018	\$0.0144	\$0.005	\$0.002

# S3 Lifecycle configuration

- Files are frequently accessed when they are created
- Generally **usage reduces with time**
- How do you save costs and **move files automatically between storage classes?**
  - Solution: S3 Lifecycle configuration
- Two kinds of actions:
  - **transition** actions (one storage class to another)
  - **expiration** actions (delete objects)
- Object can be identified by tags or prefix.



<https://docs.aws.amazon.com/AmazonS3/latest/dev/lifecycle-transition-general-considerations.html>

# Amazon S3 Replication - Same Region and Multiple Region

- Replicate objects between buckets in same or different regions
  - Could be cross account
  - Can be configured at bucket level, a shared prefix level, or an object level using S3 object tags
  - Access to destination bucket is provided using IAM Policy
- Versioning should be enabled on BOTH source and destination
- ONLY new objects are replicated (Explicitly copy existing objects)
- (Advantage) Reduces latency and helps you meet regulations
- (USECASE) Object replication between dev & test environments



S3 Bucket



S3 Bucket

# Amazon S3 - Object level configuration



Amazon S3



S3 Bucket

- You can configure these at **individual object level** (overriding bucket level configuration):
  - Storage class
  - Encryption
  - Objects ACLs

# Amazon S3 Consistency

- S3 is distributed - maintains **multiple copies** of your data in a Region to ensure durability
- Distributing data **presents a challenge**
  - How do you ensure data is consistent?
- **S3 Consistency Model**
  - READ AFTER WRITE for PUTS of new objects
  - Eventual Consistency for Overwrites PUTS and DELETES
- (In simplified words) S3 Data is highly distributed across multiple AZs and (possibly) multiple regions:
  - When you create a new object, it is immediately available
  - You might get a previous version of data immediately after an object update using PUT/DELETE
  - You will never get partial or inconsistent data

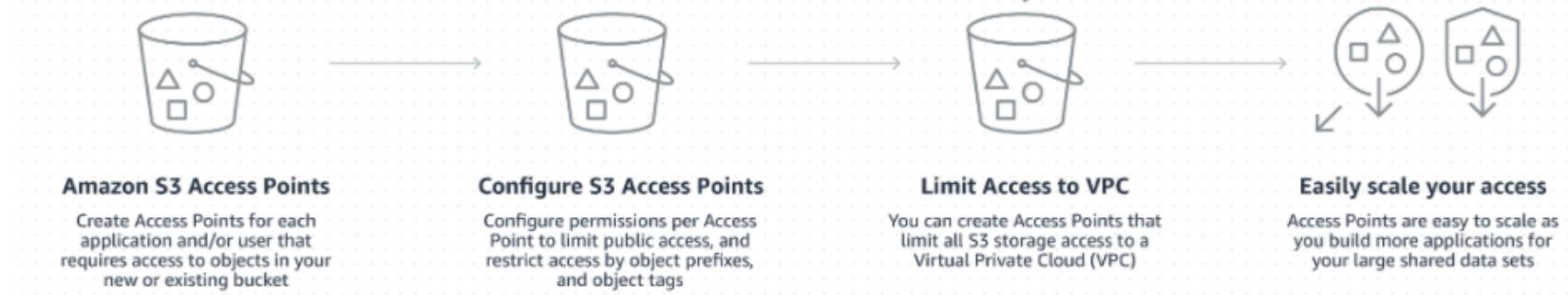


Amazon S3

# Amazon S3 Presigned URL

- Grant **time-limited permission** (few hours to 7 days) to download objects
- **Avoid** web site scraping and unintended access
- Specify:
  - Your security credentials
  - Bucket name
  - Object key
  - HTTP method and
  - Expiration date and time
- Created using AWS SDK API
  - Java code
    - `GeneratePresignedUrlRequest(bucketName, objectKey).withMethod(HttpMethod.GET).withExpiration(expiration);`

# Amazon S3 Access Points



- Simplifies bucket policy configuration
- Create application specific access points with an application specific policy
- Provide multiple customized paths with unique hostname and access policy for each bucket
- “dual-stack” endpoint supports IPv4 and IPv6 access

# Amazon S3 Scenarios - Security

In 28  
Minutes

Scenario	Solution
Prevent objects from being deleted or overwritten for a few days or for ever	Use Amazon S3 Object Lock. Can be enabled only on new buckets. Automatically enables versioning. Prevents deletion of objects. Allows you to meet regulatory requirements.
Protect against accidental deletion	Use Versioning
Protect from changing versioning state of a bucket	Use MFA Delete. You need to be an owner of the bucket AND Versioning should be enabled.
Avoid content scraping. Provide secure access.	Pre Signed URLs. Also called Query String Authentication.
Enable cross domain requests to S3 hosted website (from www.abc.com to www.xyz.com)	Use Cross-origin resource sharing (CORS)

# Amazon S3 Cost

- **Important pricing elements:**
  - Cost of Storage (per GB)
  - (If Applicable) Retrieval Charge (per GB)
  - Monthly tiering fee (Only for Intelligent Tiering)
  - Data transfer fee
- **FREE of cost:**
  - Data transfer into Amazon S3
  - Data transfer from Amazon S3 to Amazon CloudFront
  - Data transfer from Amazon S3 to services in the same region



Amazon S3

# Amazon S3 Scenarios - Costs

In 28  
Minutes

Scenario	Solution
Reduce Costs	Use proper storage classes. Configure lifecycle management.
Analyze storage access patterns and decide the right storage class for my data	Use Intelligent Tiering. Use Storage Class Analysis reports to get an analysis.
Move data automatically between storage classes	Use Lifecycle Rules
Remove objects from buckets after a specified time period	Use Lifecycle Rules and configure Expiration policy

# Amazon S3 Performance

- Amazon S3 is serverless
- Recommended for large objects
- Amazon S3 supports upto
  - 3,500 requests per second to add data
  - 5,500 requests per second to retrieve data
  - Zero additional cost
  - With each S3 prefix
- **Transfer acceleration**
  - Enable fast, easy and secure transfers of files to and from your bucket



Amazon S3

# Amazon S3 Scenarios - Performance

Scenario	Solution
Improve S3 bucket performance	Use <b>Prefixes</b> . Supports upto 3,500 RPS to add data and 5,500 RPS to retrieve data with each S3 prefix.
Upload large objects to S3	Use <b>Multipart Upload API</b> . <b>Advantages:</b> 1. Quick recovery from any network issues 2. Pause and resume object uploads 3. Begin an upload before you know the final object size. <b>Recommended</b> for files >100 MB and <b>mandatory</b> for files >4 GB
Get part of the object	Use <b>Byte-Range Fetches</b> - Range HTTP header in GET Object request <b>Recommended:</b> GET them in the same part sizes used in multipart upload
Is this recommended: EC2 (Region A) <-> S3 bucket (Region B)	No. <b>Same region recommended</b> . Reduce network latency and data transfer costs

# Amazon S3 Scenarios - Features

Scenario	Solution
Make user pay for S3 storage	<b>Requester pays</b> - The requester (instead of the bucket owner) will pay for requests and data transfer.
Create an inventory of objects in S3 bucket	<b>Use S3 inventory report</b>
I want to change object metadata or manage tags or ACL or invoke Lambda function for billions of objects stored in a single S3 bucket	<b>Generate S3 inventory report</b> <b>Perform S3 Batch Operations</b> using the report
Need S3 Bucket (or Object) Access Logs	<b>Enable S3 Server Access Logs</b> (default: off). Configure the bucket to use and a prefix (logs/).

# S3 Glacier

# Amazon S3 Glacier

- In addition to existing as a S3 Storage Class, S3 Glacier is a separate AWS Service on its own!
- **Extremely low cost storage** for archives and long-term backups:
  - Old media content
  - Archives to meet regulatory requirements (old patient records etc)
  - As a replacement for magnetic tapes
- High durability (11 9s - 99.99999999%)
- High scalability (unlimited storage)
- High security (**encrypted** at rest and in transfer)
- Cannot upload objects to Glacier using Management Console
  - Use REST API, AWS CLI, AWS SDK



Amazon Glacier

# Amazon S3 vs S3 Glacier

Feature	Amazon S3	S3 Glacier
Terminology	Objects (files) are stored in Buckets (containers)	Archives (files) are stored in Vaults (containers)
Keys	Objects keys are user defined	Archive keys are system generated identifiers
Mutability	(Default) Allows uploading new content to object	After an archive is created, it cannot be updated (Perfect for regulatory compliance)
Max size	Each object can be upto 5TB	Each archive can be upto 40TB
Management Console	Almost all bucket and object operations supported	Only vault operations are supported. You cannot upload/delete/update archives.
Encryption	Optional	Mandatory using AWS managed keys and AES-256. You can use client side encryption on top of this.
WORM Write Once Read	Enable Object Lock Policy	Enable Vault lock policy

More To...

# Retrieving archives from S3 Glacier

- Asynchronous two step process (Use REST API, AWS CLI or SDK)
  - Initiate a archive retrieval
  - (After archive is available) Download the archive
- Reduce costs by **optionally specify a range, or portion, of the archive to retrieve**
- Reduce costs by **requesting longer access times**
  - Amazon S3 Glacier:
    - Expedited (1 – 5 minutes)
    - Standard (3 – 5 hours)
    - Bulk retrievals (5–12 hours)
  - Amazon S3 Glacier Deep Archive:
    - Standard retrieval (upto 12 hours)
    - Bulk retrieval (upto 48 hours)



Amazon Glacier

# IAM - Fundamentals

# Typical identity management in the cloud

- You have **resources** in the cloud (examples - a virtual server, a database etc)
- You have **identities (human and non-human)** that need to access those resources and perform actions
  - For example: launch (stop, start or terminate) a virtual server
- How do you **identify users** in the cloud?
- How do you configure resources they can access?
- How can you configure what actions to allow?
- In AWS, *Identity and Access Management (IAM)* provides this service



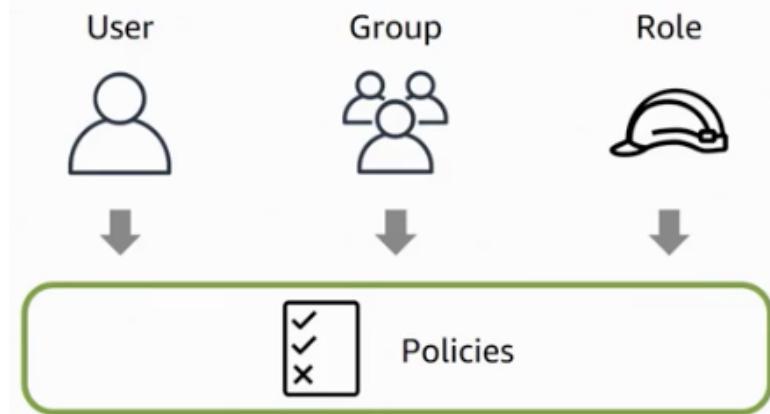
# AWS Identity and Access Management (IAM)

- **Authentication** (is it the right user?) and
- **Authorization** (do they have the right access?)
- **Identities** can be
  - AWS users or
  - Federated users (externally authenticated users)
- Provides very **granular** control
  - Limit a single user:
    - to perform single action
    - on a specific AWS resource
    - from a specific IP address
    - during a specific time window



# Important IAM Concepts

- **IAM users:** Users created in an AWS account
  - Has credentials attached (name/password or access keys)
- **IAM groups:** Collection of IAM users
- **Roles:** Temporary identities
  - Does NOT have credentials attached
  - (Advantage) Expire after a set period of time
- **Policies:** Define permissions
  - **AWS managed policies** - Standalone policy predefined by AWS
  - **Customer managed policies** - Standalone policy created by you
  - **Inline policies** - Directly embedded into a user, group or role



# AWS IAM Policies - Authorization

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "*",  
            "Resource": "*"  
        }  
    ]  
}
```

- Policy is a JSON document with one or more permissions
  - Effect - Allow or Deny
  - Resource - Which resource are you providing access to?
  - Action - What actions are allowed on the resource?
  - Condition - Are there any restrictions on IP address ranges or time intervals?
- Example above: AWS Managed Policy : AdministratorAccess

# AWS Managed Policy : AmazonS3ReadOnlyAccess

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:Get*",  
                "s3>List*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

# Customer Managed Policy : ReadSpecificS3Bucket

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:Get*",  
                "s3>List*"  
            ],  
            "Resource": "arn:aws:s3:::mybucket/somefolder/*"  
        }  
    ]  
}
```

# IAM Scenarios

In 28  
Minutes

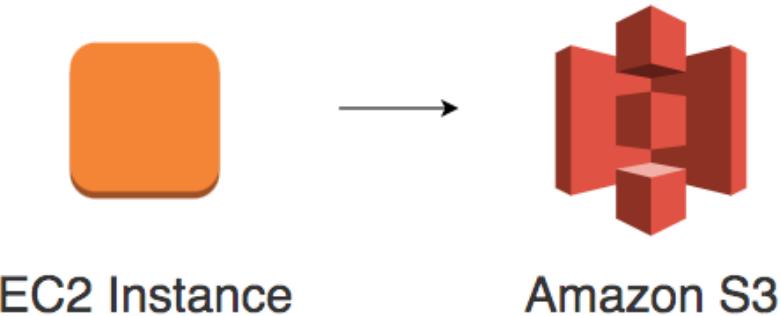
Scenario	User/Role	Recommendation
You're the only one in your account	IAM user	Do not use ROOT user
Your team needs access to your AWS account and there is no other identity mechanism	IAM users	Use IAM Groups to manage policies
EC2 instance talks with Amazon S3 or a database	IAM role	
Cross Account Access	IAM role	

# Instance Profile

- A Container (A Box) for an IAM role
- Used to pass role information to an EC2 instance
- AWS Management Console:
  - An instance profile is automatically created when you create a role for EC2 instance
- From CLI or API
  - Explicitly manage Instance Profiles - CreateInstanceProfile etc
- (REMEMBER) Instance profile is a simple container for IAM Role



# IAM Role Use case 1 : EC2 talking with S3



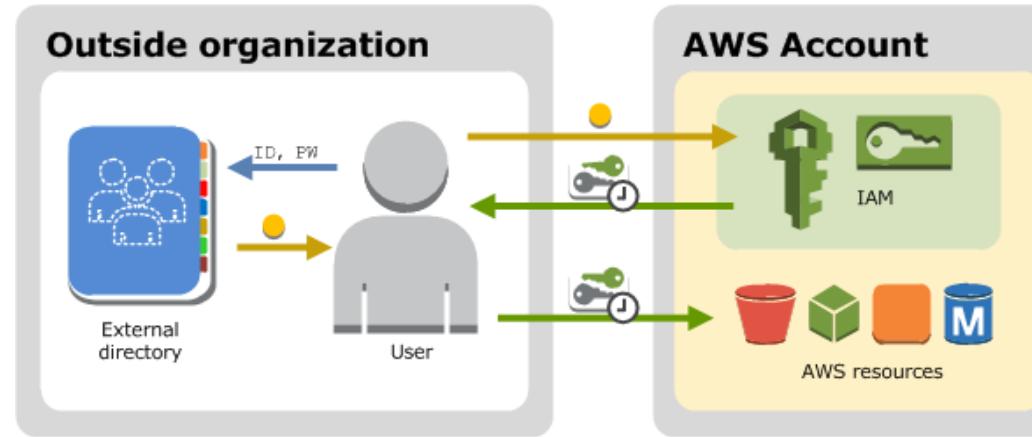
- Create IAM role with access to S3 bucket
- Assign IAM role to EC2 instance
- No need to store credentials in config files
- No need for rotation of keys

# IAM Role Use case 2: Cross Account Access

- PROD Account (111111111111)
  - Create IAM role (ProdS3AccessRole) with right permissions and establish trust relationship with AWS account 222222222222
- DEV Account (222222222222)
  - Grant users (Operations Group) permissions to assume the ProdS3AccessRole in PROD Account
    - Create a customer managed policy ProdS3AccessPolicy allowing access to call STS AssumeRole API for ProdS3AccessRole(arn:aws:iam::111111111111:role/ProdS3AccessRole)
    - Assign the policy to users (Operations Group)
    - (Optional) Enable MFA for assuming the role
- Operations user requests access to the role
  - Background: Call is made to AWS Security Token Service (AWS STS) AssumeRole API for the ProdS3AccessRole role
  - Gets access!



# IAM - Corporate Directory Federation

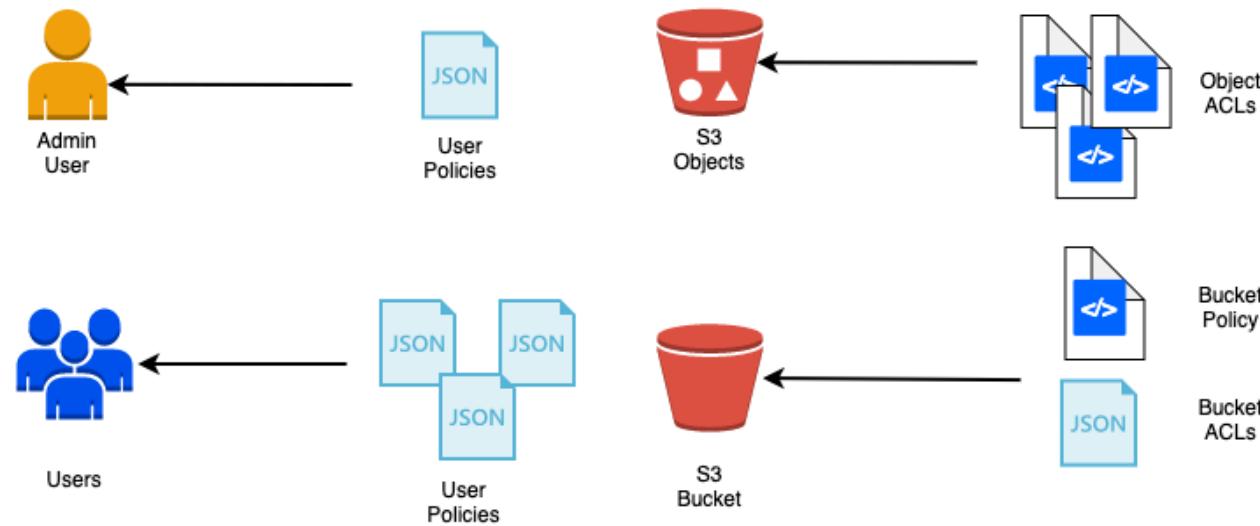


- Users authenticated with a **corporate directory**
  - For SAML2.0 compliant identity systems, establish a trust relationship with IAM
  - For enterprise using Microsoft AD (Active Directory), use AWS Directory Service to establish trust
  - Otherwise, set up a custom proxy server to translate user identities from enterprise to IAM roles

# IAM - Web Identity Federation

- Authenticate users using **web identities** - mobile or web apps
- Example: Open ID (Facebook, Google, etc)
- **Amazon Cognito** supports login with Facebook, Google, or other OpenID Connect compatible IdP
- Configure Role to use Web Identity as trusted entity
  - Authentication tokens exchanged using **STS AssumeRoleWithWebIdentity API**

# Identity-based and Resource-based policies



- By default only account owner has access to a S3 bucket
- Access policies enable other users to access S3 buckets and objects:
  - **Identity-based policies** : Attached to an IAM user, group, or role
  - **Resource-based policies and ACLs** : Attached to a resource - S3 buckets, Amazon SQS queues, and AWS KMS keys

# Identity-based and Resource-based policies

Policy Type	Identity-based	Resource-based
Attached with	IAM user, group, or role	A resource
Type	Managed and Inline	Inline only
Focus	What resource? What actions?	Who? What actions?
Example	Can list S3 buckets with name XYZ	Account A can read and modify. Public can read.
Cross-account access	User should switch role	Simpler. User accesses resource directly from his AWS account
Supported by	All services	Subset of services - S3, SQS, SNS, KMS etc
Policy Conditions	When (dates), Where(CIDR blocks), Enforcing MFA	When (dates), Where(CIDR blocks), Is SSL Mandatory?

# IAM Scenario Questions

In 28  
Minutes

Scenario	Solution
<b>How to rotate access keys without causing problems</b>	Create new access key Use new access key in all apps Disable original access key Test and verify Delete original access key
<b>How are multiple permissions resolved in IAM Policy</b>	If there is an explicit deny - return deny If there is no explicit deny and there is an explicit allow - allow If there is no explicit allow or deny - deny
<b>Which region are IAM users created in</b>	IAM Users are <b>global entities</b> . Can use AWS services in any geographic region
<b>What is the difference between IAM user, Federated user and Web identity federation user</b>	<b>IAM users</b> - created and maintained in your AWS account <b>Federated users</b> - managed outside of AWS - corporate directory <b>Web identity federation users</b> - Amazon Cognito, Amazon, Google, or any OpenID Connect-compatible provider Accounts

# Authentication with IAM - Remember

- IAM Users identities exist until they are **explicitly deleted**
- IAM allows you to **create a password policy**
  - What characters should your password contain?
  - When does a password expire?
- Access key's should be **constantly rotated**
- Two access keys can be **active simultaneously**. Makes rotation of keys easier.
- An IAM role can be added to already running EC2 instances. **Immediately effective**.
- An IAM role is **NOT** associated with an IAM user.
- An IAM user can assume an IAM role temporarily.

# Authentication with IAM - Remember

- An IAM role is **NOT associated** with long-term credentials
  - When a user, a resource (For example, an EC2 instance) or an application assumes a Role, it is provided with temporary credentials
- **Do NOT** use AWS IAM root user for regular everyday tasks. Lock it away after creating an admin IAM user.
- Enable **Multi Factor Authentication** for all important IAM operations
  - Extra layer of security
  - MFA Devices
    - Hardware device - Gemalto
    - Virtual device - An app on a smart phone

# IAM Best Practices - Recommended by AWS

- **Users** – Create individual users
- **Groups** – Manage permissions with groups
- **Permissions** – Grant least privilege
- **Auditing** – Turn on AWS CloudTrail
- **Password** – Configure a strong password policy
- **MFA** – Enable MFA for privileged users
- **Roles** – Use IAM roles for Amazon EC2 instances
- **Sharing** – Use IAM roles to share access
- **Rotate** – Rotate security credentials regularly
- **Conditions** – Restrict privileged access further with conditions
- **Root** – Reduce or remove use of root



AWS IAM

# Data Encryption KMS & Cloud HSM

# Data States

- **Data at rest:** Stored on a device or a backup
  - Examples : data on a hard disk, in a database, backups and archives
- **Data in motion:** Being transferred across a network
  - Also called **Data in transit**
  - **Examples :**
    - Data copied from on-premise to cloud storage
    - An application in a VPC talking to a database
  - **Two Types:**
    - In and out of AWS
    - Within AWS
- **Data in use:** Active data processed in a non-persistent state
  - Example: Data in your RAM



EC2 Instance



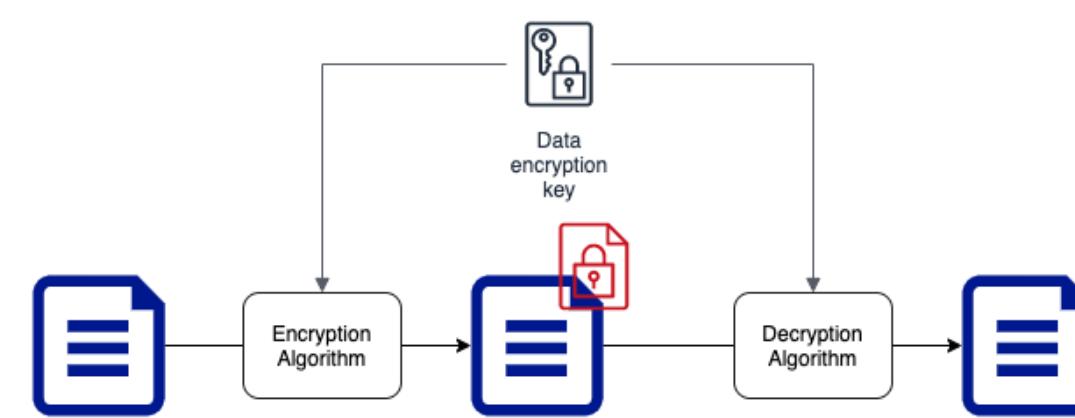
Database

# Encryption

- If you store data as is, what would happen if an **unauthorized entity gets access to it?**
  - Imagine losing an unencrypted hard disk
- **First law of security :** Defense in Depth
- Typically, enterprises encrypt all data
  - Data on your hard disks
  - Data in your databases
  - Data on your file servers
- Is it sufficient if you encrypt data at rest?
  - No. Encrypt data in transit - between application to database as well.

# Symmetric Key Encryption

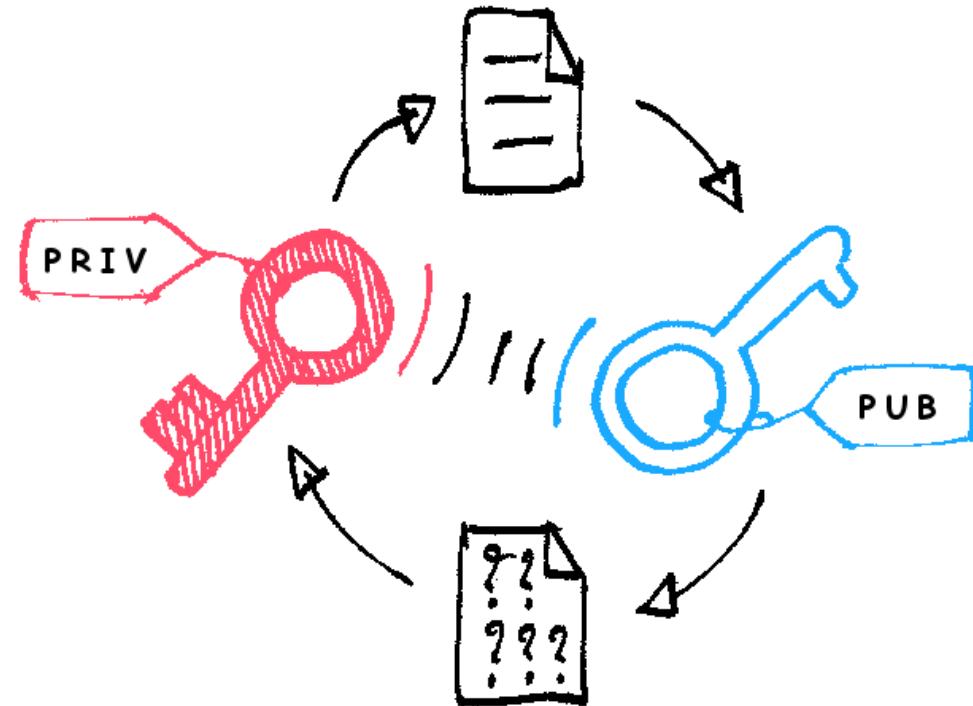
In 28  
Minutes



- Symmetric encryption algorithms use the **same key for encryption and decryption**
- Key Factor 1: Choose the **right encryption algorithm**
- Key Factor 2: How do we **secure the encryption key?**
- Key Factor 3: How do we **share the encryption key?**

# Asymmetric Key Encryption

- Two Keys : Public Key and Private Key
- Also called **Public Key Cryptography**
- Encrypt data with Public Key and decrypt with Private Key
- Share Public Key with everybody and keep the Private Key with you(YEAH, ITS PRIVATE!)
- No crazy questions:
  - Will somebody not figure out private key using the public key?
- How do you create Asymmetric Keys?



[https://commons.wikimedia.org/wiki/File:Asymmetric\\_encryption\\_\(colored\).png](https://commons.wikimedia.org/wiki/File:Asymmetric_encryption_(colored).png)

# KMS and Cloud HSM

- How do you generate, store, use and replace your keys?
- AWS provides two important services - **KMS** and **Cloud HSM**
  - Manage your keys
  - Perform encryption and decryption



AWS KMS



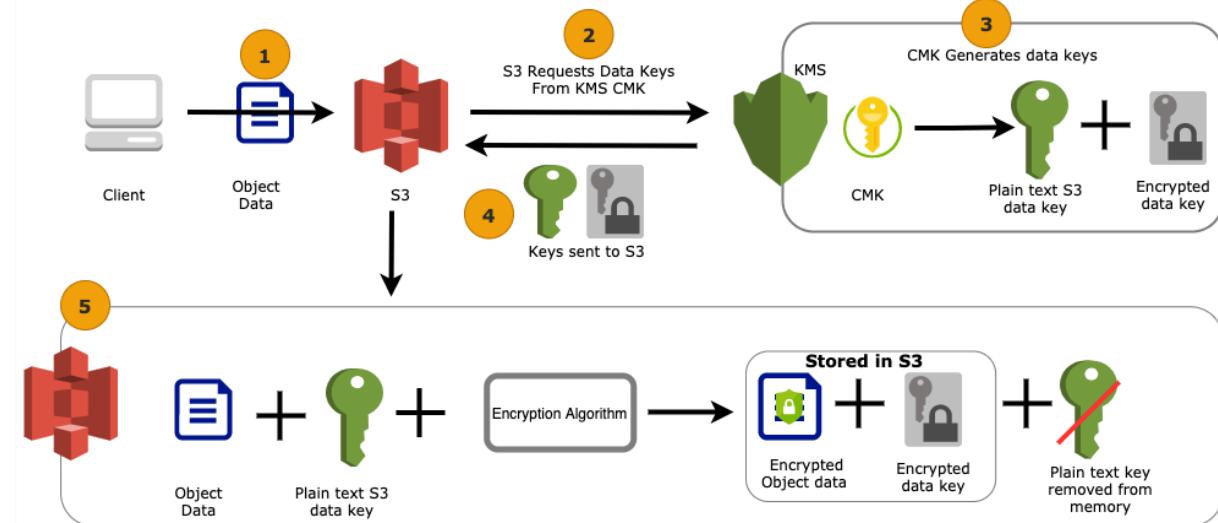
Cloud HSM

- Create and manage **cryptographic keys** (symmetric and asymmetric)
- **Control their use** in your applications and AWS Services
- Define key usage permissions (including **cross account** access)
- Track key usage in AWS CloudTrail (regulations & compliance)
- **Integrates with almost all AWS services** that need data encryption
- **Automatically rotate master keys** once a year
  - No need to re-encrypt previously encrypted data (versions of master key are maintained)
- **Schedule key deletion** to verify if the key is used
  - Mandatory minimum wait period of 7 days (max-30 days)

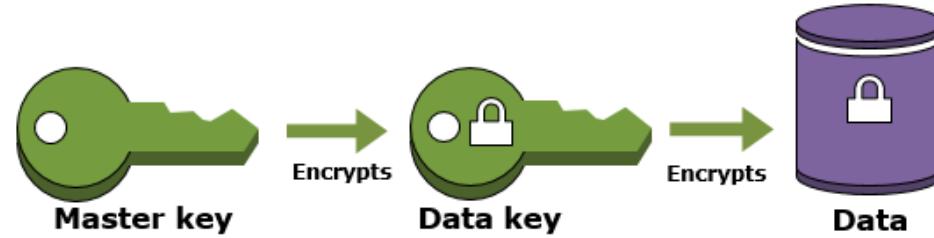


# Server Side Encryption with KMS

- Create Customer Master Key.  
Map to AWS service (S3)
- Steps
  - Data sent to S3
  - S3 receives data keys from KMS
  - S3 encrypts data
  - Stores encrypted data & data key
- Remember
  - CMK never leaves KMS
  - Encryption of data key - KMS using CMK
  - Encryption of data - AWS Service - Amazon S3 using data key



# Envelope Encryption



<https://docs.aws.amazon.com/kms/latest/developerguide/>

- The process KMS uses for encryption is called **Envelope Encryption**
  - Data is encrypted using **data key**
  - Data key is encrypted using Master key
  - Master key **never leaves KMS**
- KMS **encrypts small pieces of data** (usually data keys) <4 KB

# Customer master keys (CMKs)

- CMKs are used for encryption, decryption and signing
- 3 Types of CMKs:
  - **Customer managed:** Owned and Managed by Customer
    - Used only for your AWS account
  - **AWS managed:** Managed by AWS on your behalf
    - Used only for your AWS account
  - **AWS owned:** AWS owns and manages them
    - Used in multiple AWS accounts.
    - LIMITED usecases
- **Most Services** support both AWS managed and Customer managed Keys
  - Amazon S3, Amazon DynamoDB, Amazon EBS, Amazon SQS, Amazon SNS
- Few Services support only AWS managed keys
  - Amazon DynamoDB Accelerator (DAX), AWS CodeCommit



AWS KMS

# Decryption of data using KMS

- AWS service (Amazon S3) sends **encrypted data key** to KMS
- KMS uses Customer Master Key (CMK) to decrypt and return **plain-text data key**
- AWS service (Amazon S3) uses the plain-text data key to perform decryption
- (TIP) Remove plain-text data key from memory asap
- (TIP) AWS service needs IAM permissions to use the CMK
- Remember:
  - (Optional) You can associate a key/value map called **encryption context** with any cryptographic operation
  - (TIP) If encryption context is different, decryption will NOT succeed



# KMS APIs - Encryption

- **GenerateDataKey** - Generate data key
  - Uses CMK and Returns data key in plain text and encrypted formats
- **GenerateDataKeyWithoutPlaintext** - Generate data key
  - Uses CMK and Only returns encrypted data key
- **Encrypt** - Encrypt data (size limit - 4KB)
- **Decrypt** - Decrypt a data key (size limit - 4KB)
- **ReEncrypt** - Decrypt an encrypted data key and re-encrypt it with a different CMK immediately (size limit - 4KB)



AWS KMS

# KMS APIs - Others and Quotas

- **ListKeyPolicies**, **GetKeyPolicy**, **PutKeyPolicy** - Play with Key Policies
- **ScheduleKeyDeletion**, **CancelKeyDeletion** - Schedule and cancel deletion of keys
- (Remember) Request quotas for KMS Cryptographic operations - 5,500 to 30,000 per second (varies with Region)
  - You might get a **ThrottlingException** if you exceed the limit
  - Lower your request rate to AWS KMS
  - Retry with Exponential Backoff



# KMS with S3 (or DynamoDB or SQS or ..)

- **What happens in background?**

- Envelope Encryption with API calls
  - GenerateDataKey while storing
  - Decrypt while retrieval

- **What permissions are needed?**

- KMS Key Policy on CMK (Resource Policy) allows access from the service (S3 or DynamoDB or ..)
- Your IAM policy allows access to perform the operation on KMS
  - Call to s3:PutObject would need access to kms:GenerateDataKey if encryption is enabled on S3 bucket!

- **What could go wrong?**

- No Permissions
  - Check KMS Key Policy and User's IAM policy
- Throttling
  - Retry with Exponential Backoff or Increase KMS Quotas



# KMS with S3 - Use cases

Key Storage	Encryption Location	Requirement	Recommendation	
Customer	in S3	You want to manage the keys (including rotation) outside AWS	SSE with Customer-Provided Keys (SSE-C)	 User
KMS	in S3	Easy Management of Keys. Auditing.	SSE with Customer Master Keys (SSE-KMS)	 Amazon S3
KMS	in S3	You want Encryption but Don't want Management	SSE with Amazon S3-Managed Keys (SSE-S3)	 AWS KMS
Customer (master key stored within your app)	On Premises		CSE (Amazon S3 encryption client)	
KMS	On Premises		CSE (Amazon S3 encryption client)	

# KMS with CloudWatch



- You can use KMS to encrypt your CloudWatch logs (OPTIONAL):
- **Remember:**
  - You should have **permissions to use KMS keys**
    - kms>CreateKey, kms:GetKeyPolicy, and kms:PutKeyPolicy
  - **KMS Key Policy** should allow access from CloudWatch Logs
  - CloudWatch Logs must have permissions for the CMK whenever encrypted data is requested:
    - Without access to KMS keys, your encrypted data in CloudWatch Logs can no longer be retrieved.



- **Maximum size of an object** that can be encrypted in KMS is **4 KB**.
  - Use Envelope Encryption for larger objects
    - Generate a data key from KMS (GenerateDataKey)
    - Use data key to perform encryption on the object (within the service or client-side)
- Usage of KMS CMKs can be tracked in **CloudTrail**
- **Key policies** can be used to control access to CMKs
  - Control Cross Account Sharing of Keys in CMK key policy
- Use **AWS Encryption SDK** to interact with KMS(Provides Data Key Caching)

# AWS CloudHSM

- Managed (highly available & auto scaling) **dedicated single-tenant** Hardware Security Module(HSM) for regulatory compliance
  - (Remember) AWS KMS is a multi-tenant service
- FIPS 140-2 Level 3 compliant
- AWS **CANNOT** access your encryption master keys in CloudHSM
  - In KMS, AWS can access your master keys
  - Be ultra safe with your keys when you are using CloudHSM
  - **(Recommendation)** Use two or more HSMs in separate AZs in a production cluster



Cloud HSM

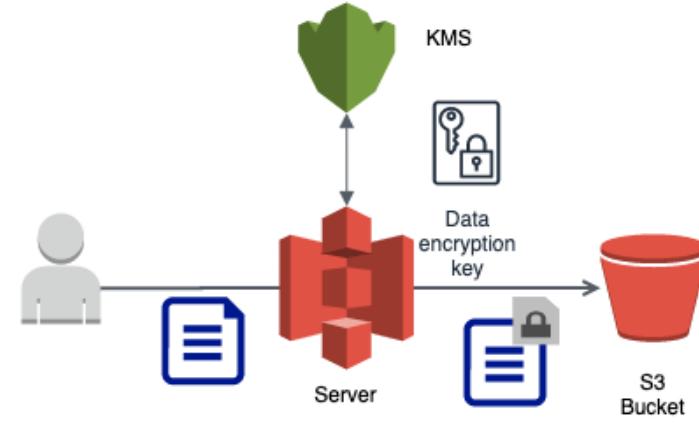
# AWS CloudHSM

- AWS KMS can use CloudHSM cluster as "custom key store" to store the keys:
  - AWS Services can continue to talk to KMS for data encryption
  - (AND) KMS does the necessary integration with CloudHSM cluster
- (Best Practice) **CloudWatch** for monitoring and **CloudTrail** to track key usage
- **Use cases**
  - (Web servers) Offload SSL processing
  - Certificate Authority
  - Digital Rights Management
  - TDE for Oracle databases



# Server Side Encryption

- Client sends data (as is) to AWS service
- AWS service interacts with KMS to perform encryption on the server side
- Recommended to **use HTTPS endpoints** to ensure encryption of data in transit
  - All AWS services (including S3) provides HTTPS endpoints
  - Encryption is optional with S3 but highly recommended in flight and at rest



# Server Side Encryption - S3

- **SSE-S3:**

- AWS S3 manages its own keys
- Keys are rotated every month
- Request Header - *x-amz-server-side-encryption(AES256)*

- **SSE-KMS:**

- Customer manages keys in KMS
- Request Headers - *x-amz-server-side-encryption(aws:kms)* and *x-amz-server-side-encryption-aws-kms-key-id(ARN for key in KMS)*

- **SSE-C:**

- Customer sends the key with every request
- S3 performs encryption and decryption without storing the key
- HTTPS is mandatory



# Client Side Encryption

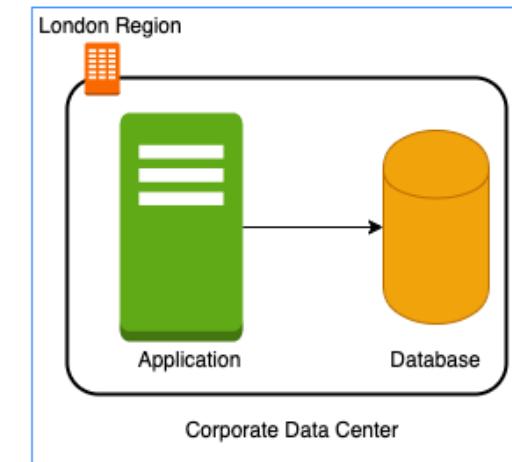
- Client manages encryption process and sends encrypted data to AWS service
  - AWS will not be aware of master key or data key
- AWS service stores data as is
- For Amazon S3, you can use a client library (Amazon S3 Encryption Client)



# Networking

# Need for Amazon VPC

- In a corporate network or an on-premises data center:
  - Can anyone on the internet **see the data exchange** between the application and the database?
    - No
  - Can anyone from internet **directly connect to your database?**
    - Typically NO.
    - You need to connect to your corporate network and then access your applications or databases.
- Corporate network provides a **secure internal network** protecting your resources, data and communication from external users
- How do you do create **your own private network** in the cloud?
  - Enter **Virtual Private Cloud (VPC)**



# Amazon VPC (Virtual Private Cloud)

- Your own **isolated network** in AWS cloud
  - Network traffic within a VPC is isolated (not visible) from all other Amazon VPCs
- You **control all the traffic** coming in and going outside a VPC
- **(Best Practice)** Create all your AWS resources (compute, storage, databases etc) **within a VPC**
  - Secure resources from unauthorized access AND
  - Enable secure communication between your cloud resources



VPC

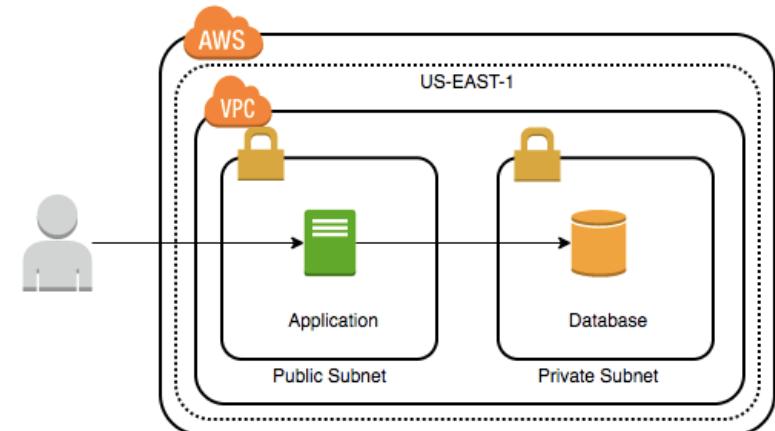
# Need for VPC Subnets



- Different resources are created on cloud - databases, compute (EC2) etc
- Each type of resource has **its own access needs**
- Public Elastic Load Balancers are accessible from internet (**public resources**)
- Databases or EC2 instances should NOT be accessible from internet
  - ONLY applications within your network (VPC) should be able to access them(**private resources**)
- How do you **separate public resources from private resources** inside a VPC?

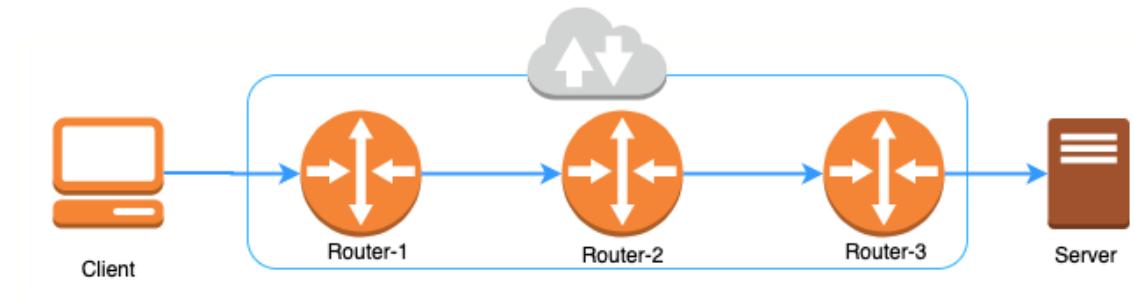
# VPC Subnets

- (Solution) Create different subnets for public and private resources
  - Resources in a public subnet **CAN** be accessed from internet
  - Resources in a private subnet **CANNOT** be accessed from internet
  - BUT resources in public subnet can talk to resources in private subnet
- Each VPC is created in a Region
- Each Subnet is created in an Availability Zone
- **Example :** VPC - us-east-1 => Subnets - AZs us-east-1a or us-east-1b or ..



# Routing on the internet

In 28  
Minutes



- You have an IP address of a website you want to visit
- There is **no direct connection** from your computer to the website
- Internet is actually a **set of routers** routing traffic
- Each router has a set of rules that help it decide the path to the destination IP address

# Routing inside AWS

In 28  
Minutes

Destination	Target
172.31.0.0/16	Local
0.0.0.0/0	igw-1234567

- In AWS, **route tables** are used for routing
- Route tables can be associated with VPCs and subnets
- Each route table consists of a **set of rules** called routes
  - Each route or routing rule has a **destination and target**
  - What Range of Addresses should be routed to which target resource?
- **Rule 1** - Route requests to VPC CIDR 172.31.0.0/16 (172.31.0.0 to 172.31.255.255) to local resources within the VPC
- **Rule 2** - Route all other IP addresses (0.0.0.0/0) to internet (internet gateway)

# Public Subnet vs Private Subnet



- Public Subnet
  - Communication allowed from subnet to internet
  - Communication allowed from internet to subnet
- Private Subnet
  - Communication NOT allowed from internet to subnet

# Public Subnet vs Private Subnet

Name	Destination	Target	Explanation
RULE 1	172.31.0.0/16	Local	Local routing
RULE 2	0.0.0.0/0	igw-1234567	Internet routing

- An **Internet Gateway** enables internet communication for subnets
- Any subnet which has a route to an internet gateway is called a **public subnet**
- Any subnet which **DOES NOT** have route to an internet gateway is called a **private subnet**



Subnet



Internet Gateway



Internet

# Network Address Translation(NAT) Instance and Gateway

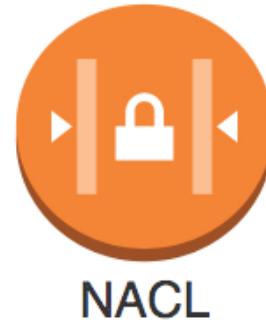
- How do you allow instances in a private subnet to download software updates and security patches while denying inbound traffic from internet?
- How do you allow instances in a private subnet to connect privately to other AWS Services outside the VPC?
- Three Options:
  - NAT Gateway: Managed Service
  - NAT Instance: Install a EC2 instance with specific NAT AMI and configure as a gateway
  - Egress-Only Internet Gateways: For IPv6 subnets

# NAT gateway vs NAT instance

Feature	NAT gateway	NAT instance
Managed by	AWS	You
Created in	Public Subnet	Public Subnet
Internet Gateway	Needed	Needed
High Availability	Yes (in an AZ) Multi AZ (higher availability)	You are responsible.

# Network Access Control List

- Security groups control traffic to a specific resource in a subnet.
- How about stopping traffic from **even entering the subnet?**
- NACL provides **stateless firewall** at subnet level.
- Each subnet **must** be associated with a NACL.
- **Default NACL** allows all inbound and outbound traffic.
- **Custom created NACL** denies all inbound and outbound traffic by default.
- Rules have a priority number.
  - Lower number => Higher priority.



# Security Group vs NACL



Feature	Security Group	NACL
<b>Level</b>	Assigned to a specific instance(s)/resource(s)	Configured for a subnet. Applies to traffic to all instances in a subnet.
<b>Rules</b>	Allow rules only	Both allow and deny rules
<b>State</b>	Stateful. Return traffic is automatically allowed.	Stateless. You should explicitly allow return traffic.
<b>Evaluation</b>	Traffic allowed if there is a matching rule	Rules are prioritized. Matching rule with highest priority wins.

# VPC Flow Logs

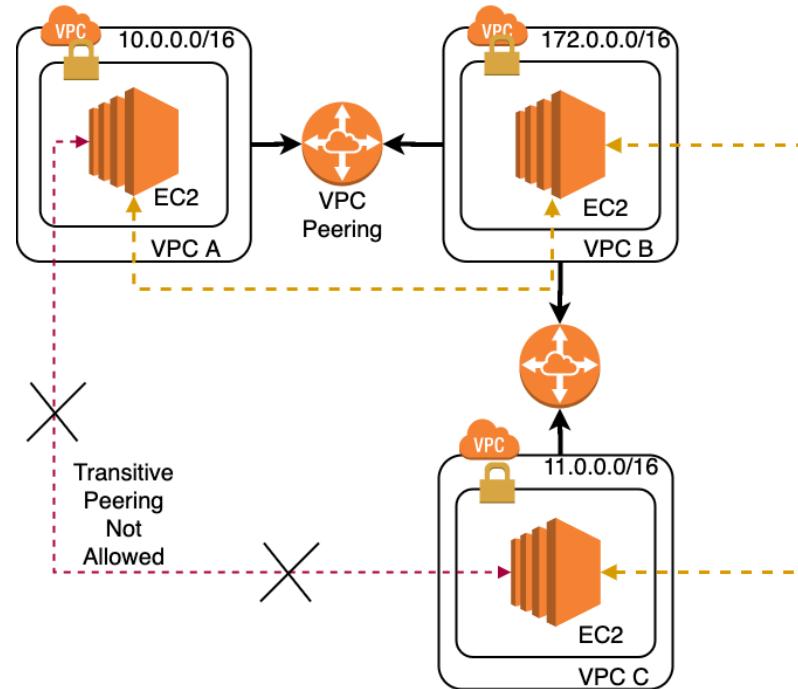
- Monitor network traffic
- Troubleshoot connectivity issues (NACL and/or security groups misconfiguration)
- Capture traffic going in and out of your VPC (network interfaces)
- Can be created for
  - a VPC
  - a subnet
- Publish logs to Amazon CloudWatch Logs or Amazon S3
- Flow log records contain ACCEPT or REJECT
  - Is traffic is permitted by security groups or network ACLs?



VPC Flow Logs

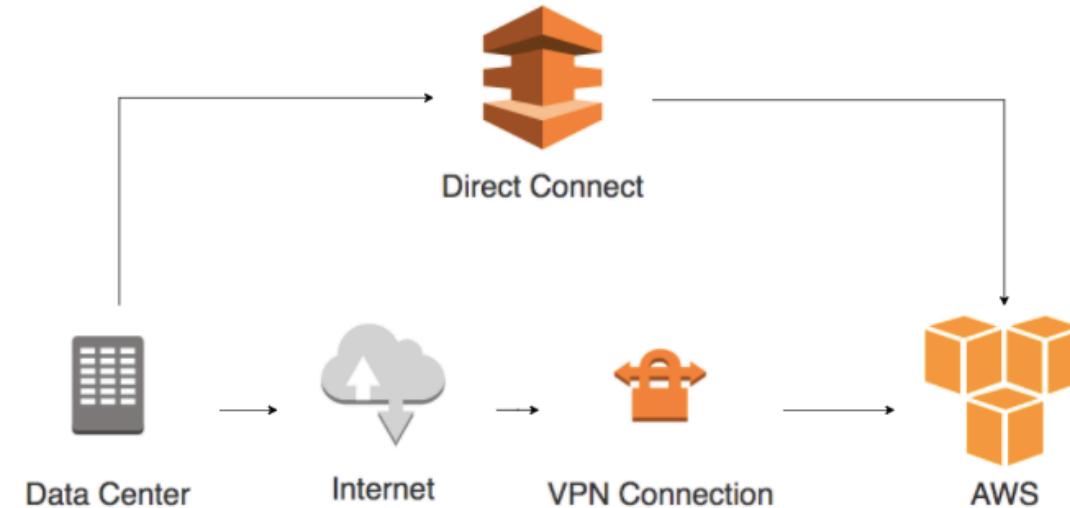
# VPC Peering

- Connect VPCs belonging to same or different AWS accounts irrespective of the region of the VPCs
- Allows private communication between the connected VPCs
- Peering uses a request/accept protocol
  - Owner of requesting VPC sends a request
  - Owner of the peer VPC has one week to accept



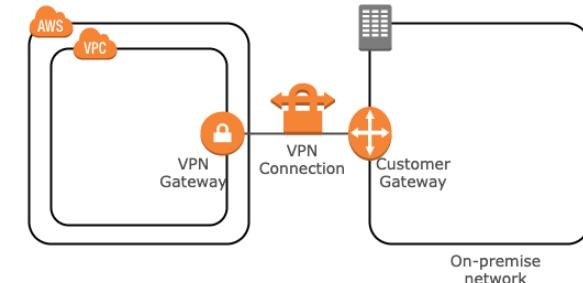
# AWS and On-Premises - Overview

- AWS Managed VPN
  - IPsec VPN tunnels from VPC to customer network
- AWS Direct Connect (DX)
  - Private dedicated network connection from on-premises to AWS



# AWS Managed VPN

- IPsec VPN tunnels from VPC to customer network
- Traffic over internet - encrypted using IPsec protocol
- VPN gateway to connect one VPC to customer network
- Customer gateway installed in customer network
  - You need a Internet-routable IP address of customer gateway



# AWS Direct Connect (DC)

- Private dedicated network connection from on-premises to AWS
- Advantages:
  - Private network
  - Reduce your (ISP) bandwidth costs
  - Consistent Network performance because of private network
- (REMEMBER) Establishing DC connection can take more than a month



# VPC Endpoint

- Securely connect your VPC to another service
- Gateway endpoint
  - Securely connect to Amazon S3 and DynamoDB
  - Endpoint serves as a target in your route table for traffic
  - Provide access to endpoint (endpoint, identity and resource policies)
- Interface endpoint
  - Securely connect to AWS services EXCEPT FOR Amazon S3 and DynamoDB
  - Powered by PrivateLink (keeps network traffic within AWS network)
  - Needs a elastic network interface (ENI) (entry point for traffic)
- (Avoid DDoS & MTM attacks) Traffic does NOT go thru internet
- (Simple) Does NOT need Internet Gateway, VPN or NAT



# VPC - Review

- **VPC:** Virtual Network to protect resources and communication from outside world.
- **Subnet:** Separate private resources from public resources
- **Internet Gateway:** Allows Public Subnets to connect/accept traffic to/from internet
- **NAT Gateway:** Allow internet traffic from private subnets
- **VPC Peering:** Connect one VPC with other VPCs
- **VPC Flow Logs:** Enable logs to debug problems
- **AWS Direct Connect:** Private pipe from AWS to on-premises
- **AWS VPN:** Encrypted (IPsec) tunnel over internet to on-premises



# Database Fundamentals

# Databases Primer

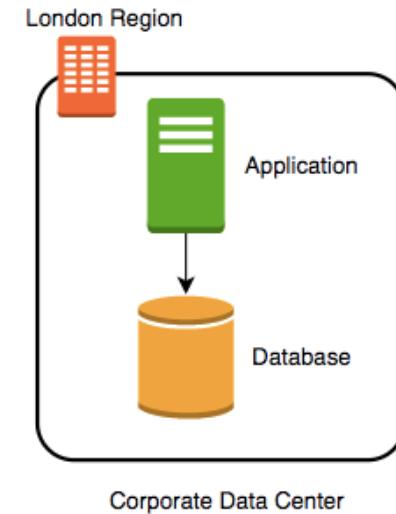
- Databases provide **organized** and **persistent** storage for your data
- To **choose between different database types**, we would need to understand:
  - Availability
  - Durability
  - RTO
  - RPO
  - Consistency
  - Transactions etc
- Let's get started on a **simple journey** to understand these



Database

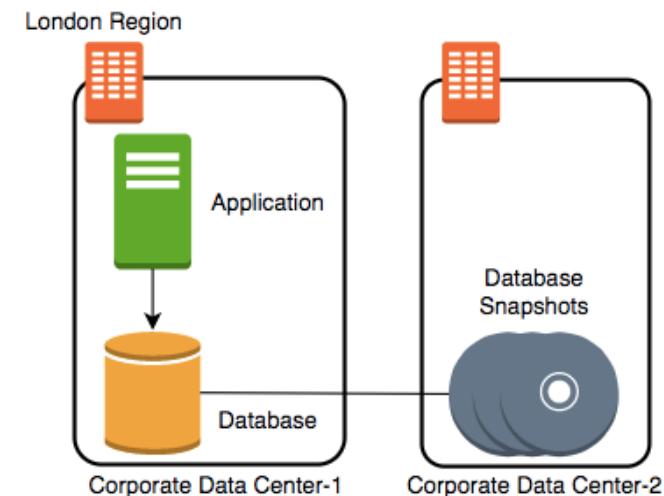
# Database - Getting Started

- Imagine a database deployed in a data center in London
- Let's consider some challenges:
  - Challenge 1: Your database will go down if the data center crashes or the server storage fails
  - Challenge 2: You will lose data if the database crashes



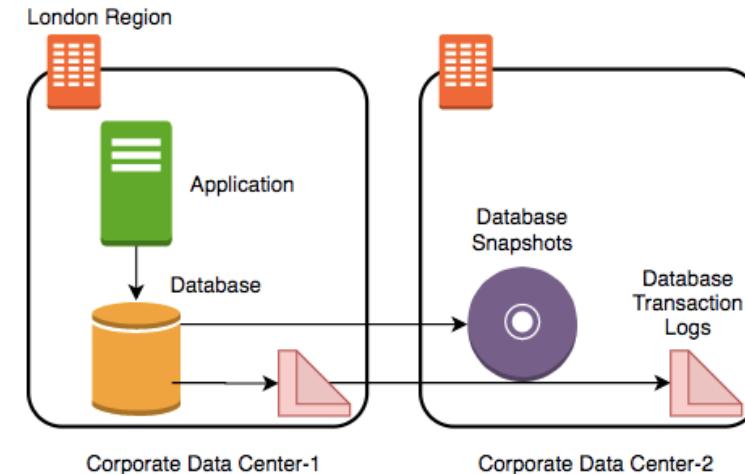
# Database - Snapshots

- Let's automate taking copy of the database (**take a snapshot**) every hour to another data center in London
- Let's consider some challenges:
  - **Challenge 1:** Your database will go down if the data center crashes
  - **Challenge 2 (PARTIALLY SOLVED):** You will lose data if the database crashes
    - You can setup database from latest snapshot. But depending on when failure occurs you can lose up to an hour of data
  - **Challenge 3(NEW):** Database will be slow when you take snapshots



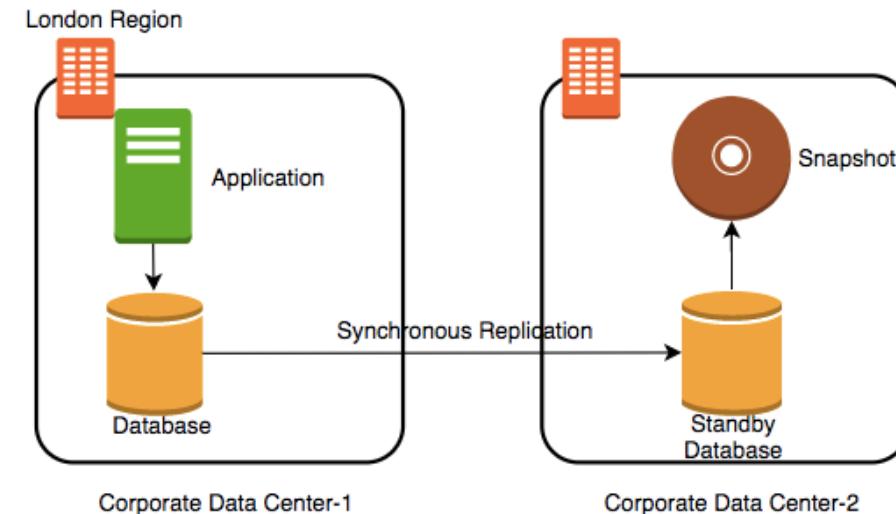
# Database - Transaction Logs

- Let's add transaction logs to database and create a process to copy it over to the second data center
- Let's consider some challenges:
  - **Challenge 1:** Your database will go down if the data center crashes
  - **Challenge 2 (SOLVED):** You will lose data if the database crashes
    - You can setup database from latest snapshot and apply transaction logs
  - **Challenge 3:** Database will be slow when you take snapshots



# Database - Add a Standby

- Let's add a **standby database** in the second data center with replication
- Let's consider some challenges:
  - **Challenge 1 (SOLVED):** Your database will go down if the data center crashes
    - You can switch to the standby database
  - **Challenge 2 (SOLVED):** You will lose data if the database crashes
  - **Challenge 3 (SOLVED):** Database will be slow when you take snapshots
    - Take snapshots from standby.
    - Applications connecting to master will get good performance always



# Availability and Durability

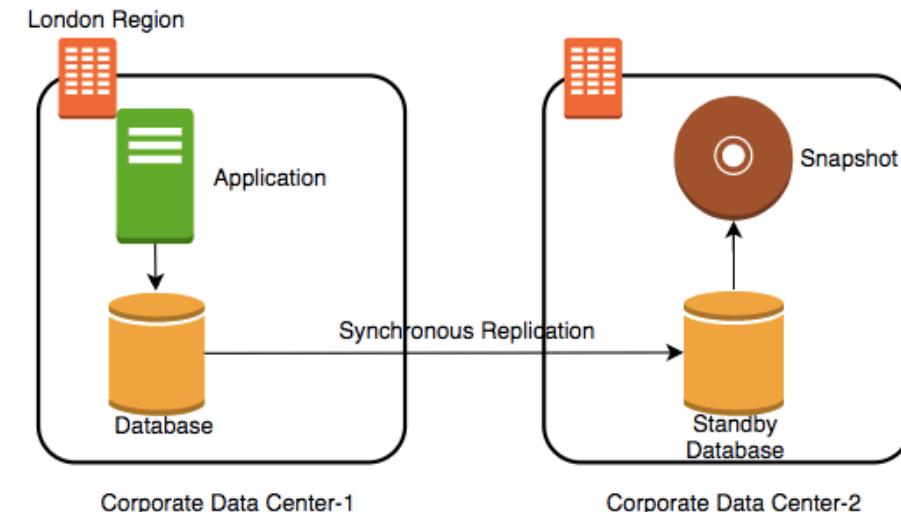
- **Availability**
  - Will I be able to access my data now and when I need it?
  - Percentage of time an application provides the operations expected of it
- **Durability**
  - Will my data be available after 10 or 100 or 1000 years?
- Examples of measuring availability and durability:
  - 4 9's - 99.99
  - 11 9's - 99.999999999
- Typically, an **availability of four 9's** is considered very good
- Typically, a **durability of eleven 9's** is considered very good

# Availability

Availability	Downtime (in a month)	Comment
99.95%	22 minutes	
99.99% (4 9's)	4 and 1/2 minutes	Typically online apps aim for 99.99% (4 9's) availability
99.999% (5 9's)	26 seconds	Achieving 5 9's availability is tough

# Durability

- What does a durability of 11 9's mean?
  - If you store one million files for ten million years, you would expect to lose one file
- Why should durability be high?
  - Because we hate losing data
  - Once we lose data, it is gone



# Increasing Availability and Durability of Databases

- **Increasing Availability:**

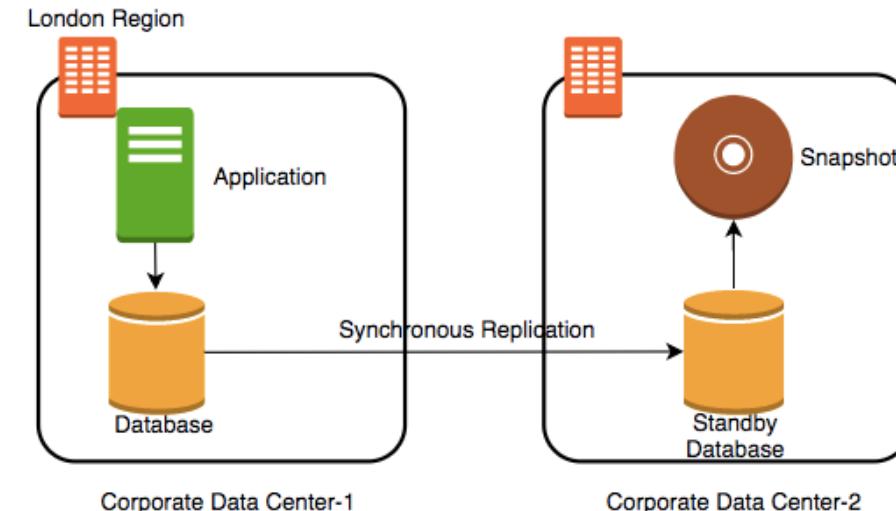
- Have multiple standbys available
  - in multiple AZs
  - in multiple Regions

- **Increasing Durability:**

- Multiple copies of data (standbys, snapshots, transaction logs and replicas)
  - in multiple AZs
  - in multiple Regions

- **Replicating data comes with its own challenges!**

- We will talk about them a little later



# Database Terminology : RTO and RPO

- Imagine a **financial transaction being lost**
- Imagine a **trade being lost**
- Imagine a **stock exchange going down for an hour**
- Typically businesses are fine with some downtime but they hate losing data
- Availability and Durability are technical measures
- How do we measure **how quickly we can recover from failure?**
  - **RPO (Recovery Point Objective)**: Maximum acceptable period of data loss
  - **RTO (Recovery Time Objective)**: Maximum acceptable downtime
- Achieving **minimum RTO and RPO is expensive**
- **Trade-off** based on the criticality of the data



Database

## Question - RTO and RPO

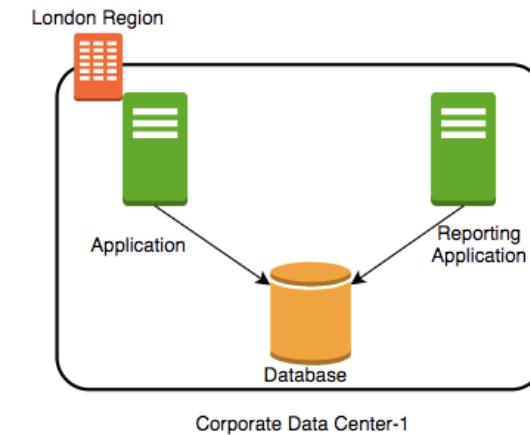
- You are running an EC2 instance storing its data on a EBS. You are taking EBS snapshots every 48 hours. If the EC2 instance crashes, you can manually bring it back up in 45 minutes from the EBS snapshot. What is your RTO and RPO?
  - RTO - 45 minutes
  - RPO - 48 hours

# Achieving RTO and RPO - Failover Examples

Scenario	Solution
<b>Very small data loss (RPO - 1 minute)</b> <b>Very small downtime (RTO - 5 minutes)</b>	<b>Hot standby</b> - Automatically synchronize data Have a standby ready to pick up load Use automatic failover from master to standby
<b>Very small data loss (RPO - 1 minute)</b> <b>BUT I can tolerate some downtimes (RTO - 15 minutes)</b>	<b>Warm standby</b> - Automatically synchronize data Have a standby with minimum infrastructure Scale it up when a failure happens
<b>Data is critical (RPO - 1 minute) but I can tolerate downtime of a few hours (RTO - few hours)</b>	Create regular data <b>snapshots and transaction logs</b> Create database from snapshots and transactions logs when a failure happens
<b>Data can be lost without a problem (for example: cached data)</b>	Failover to a completely new server

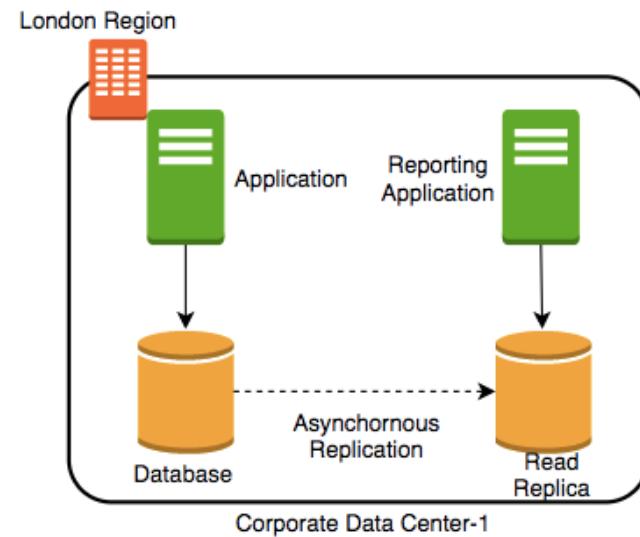
# (New Scenario) Reporting and Analytics Applications

- New reporting and analytics applications are being launched using the same database
  - These applications will ONLY read data
- Within a few days you see that the database performance is impacted
- How can we fix the problem?
  - Vertically scale the database - increase CPU and memory
  - Create a database cluster - typically database clusters are expensive to setup
  - Create read replicas - Run read only applications against read replicas



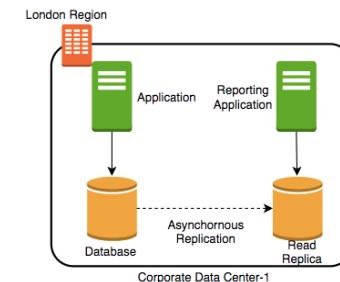
# Database - Read Replicas

- Add read replica
- Connect reporting and analytics applications to read replica
- Reduces load on the master databases
- Upgrade read replica to master database (supported by some databases)
- Create read replicas in multiple regions
- Take snapshots from read replicas



# Consistency

- How do you ensure that data in multiple database instances (standbys and replicas) is updated simultaneously?
- **Strong consistency** - Synchronous replication to all replicas
  - Will be slow if you have multiple replicas or standbys
- **Eventual consistency** - Asynchronous replication. A little lag - few seconds - before the change is available in all replicas
  - In the intermediate period, different replicas might return different values
  - Used when scalability is more important than data integrity
  - Examples : Social Media Posts - Facebook status messages, Twitter tweets, LinkedIn posts etc
- **Read-after-Write consistency** - Inserts are immediately available. Updates and deletes are eventually consistent
  - Amazon S3 provides read-after-write consistency



# Database Categories

- There are **several categories** of databases:
  - Relational (OLTP and OLAP), Document, Key Value, Graph, In Memory among others
- **Choosing type of database** for your use case is not easy. A few factors:
  - Do you want a **fixed schema**?
    - Do you want flexibility in defining and changing your schema? (schemaless)
  - What level of **transaction properties** do you need? (atomicity and consistency)
  - What kind of **latency** do you want? (seconds, milliseconds or microseconds)
  - How many **transactions** do you expect? (hundreds or thousands or millions of transactions per second)
  - How much **data** will be stored? (MBs or GBs or TBs or PBs)
  - and a lot more...



Amazon RDS



ElastiCache



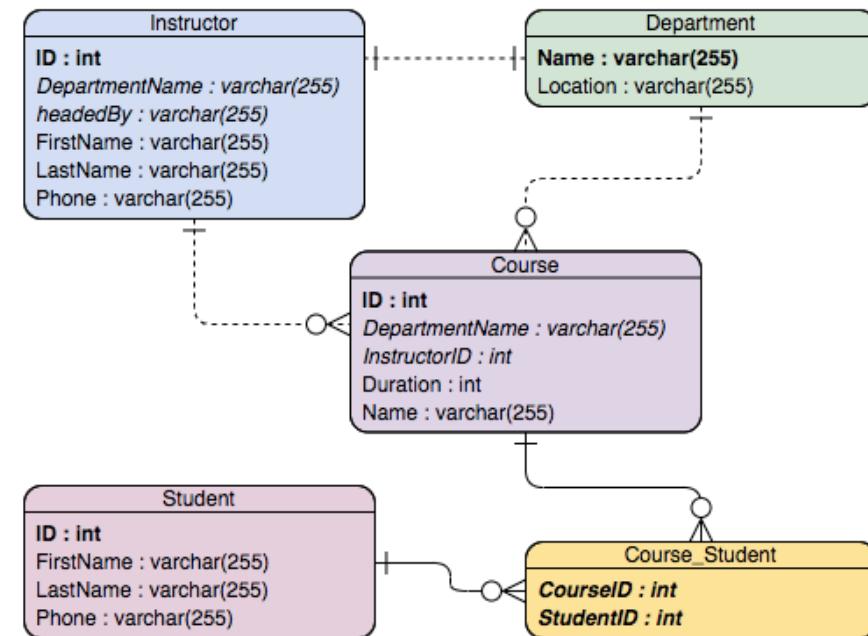
DynamoDB



Redshift

# Relational Databases

- This was the **only** option until a decade back!
- Most **popular** (or unpopular) type of databases
- **Predefined schema** with tables and relationships
- Very **strong transactional** capabilities
- Used for
  - OLTP (Online Transaction Processing) use cases and
  - OLAP (Online Analytics Processing) use cases



# Relational Database - OLTP (Online Transaction Processing)

In 28  
Minutes

- Applications where large number of users make large number of small transactions
  - small data reads, updates and deletes
- Use cases:
  - Most traditional applications, ERP, CRM, e-commerce, banking applications
- Popular databases:
  - MySQL, Oracle, SQL Server etc
- Recommended AWS Managed Service:
  - Amazon RDS
  - Supports Amazon Aurora, PostgreSQL, MySQL, MariaDB (Enhanced MySQL), Oracle Database, and SQL Server



Amazon RDS

# Relational Database - OLAP (Online Analytics Processing)

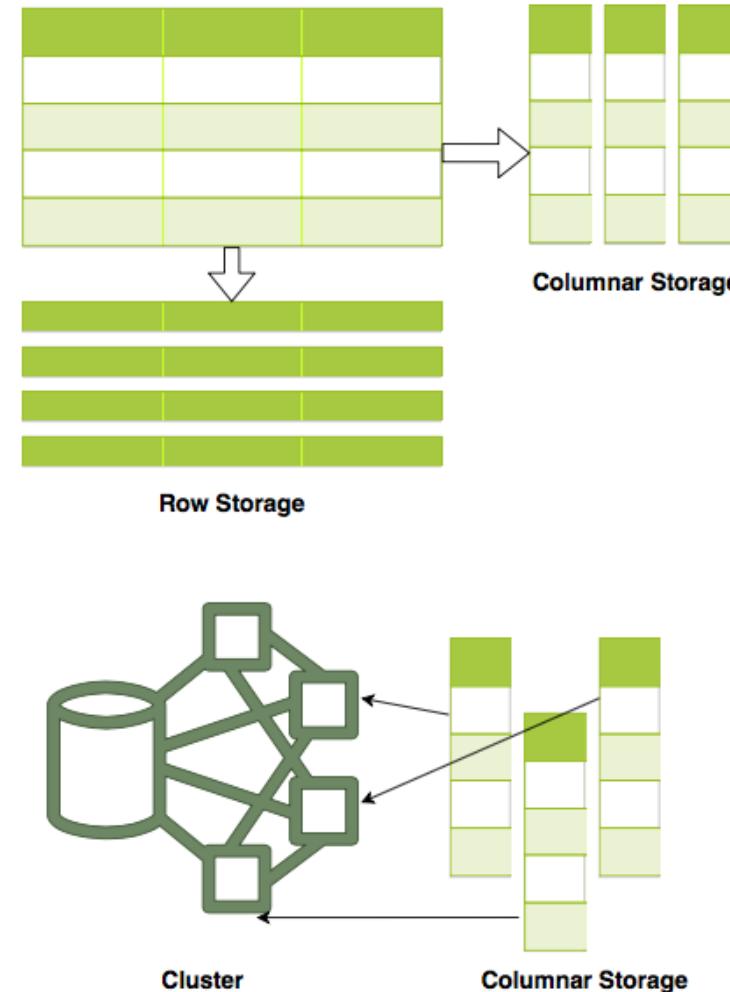
- Applications allowing users to **analyze petabytes of data**
  - Examples : Reporting applications, Data ware houses, Business intelligence applications, Analytics systems
  - Sample application : Decide insurance premiums analyzing data from last hundred years
  - Data is consolidated from multiple (transactional) databases
- Recommended AWS Managed Service
  - Amazon Redshift
  - Petabyte-scale distributed data ware house based on PostgreSQL



Redshift

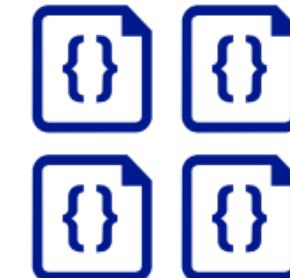
# Relational Databases - OLAP vs OLTP

- OLAP and OLTP use similar data structures
- BUT **very different approach in how data is stored**
- **OLTP databases use row storage**
  - Each table row is stored together
  - Efficient for processing small transactions
- **OLAP databases use columnar storage**
  - Each table column is stored together
  - **High compression** - store petabytes of data efficiently
  - **Distribute data** - one table in multiple cluster nodes
  - **Execute single query across multiple nodes** - Complex queries can be executed efficiently



# Document Database

- Structure data **the way your application needs it**
- Create **one table instead of dozens!**
- **Quickly evolving semi structured data (schema-less)**
- **Use cases** : Content management, catalogs, user profiles
- **Advantages** : (Horizontally) Scalable to terabytes of data with **millisecond responses upto millions of transactions per second**
- Recommended AWS Managed Service
  - Amazon DynamoDB

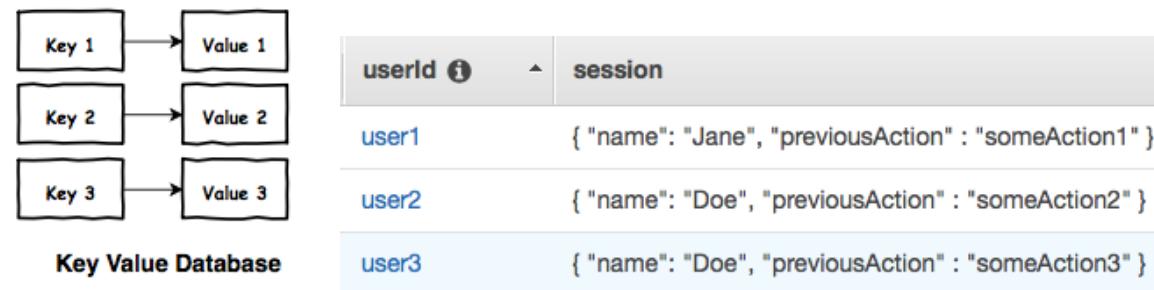


Document Database

```
{
  "id": 1,
  "name": "Jane Doe",
  "username": "abcdefg",
  "email": "someone@gmail.com",
  "address": {
    "street": "Some Street",
    "suite": "Apt. 556",
    "city": "Hyderabad",
    "zipcode": "500018",
    "geo": {
      "lat": "-3.31",
      "lng": "8.14"
    }
  },
  "phone": "9-999-999-9999",
  "website": "in28minutes.com",
  "company": "J
```

# Key-value

In 28  
Minutes



- Use a **simple key-value pair** to store data. Key is a unique identifier.
- Values can be objects, compound objects or simple data values
- **Advantages :** (Horizontally) Scalable to **terabytes of data** with **millisecond responses** upto **millions of transactions per second**
- Recommended AWS Managed Service - **Amazon DynamoDB** again
- **Use cases :** shopping carts, session stores, gaming applications and very high traffic web apps

# Graph



- **Store and navigate** data with complex relationships
- **Use cases** : Social Networking Data (Twitter, Facebook), Fraud Detection
- Recommended AWS Managed Service - **Amazon Neptune**

# In-memory Databases

- Retrieving data from memory is much faster from retrieving data from disk
- In-memory databases like Redis deliver microsecond latency by storing **persistent data in memory**
- Recommended AWS Managed Service
  - Amazon ElastiCache
  - Supports Redis and Memcached
    - Redis is recommended for persistent data
    - Memcached is recommended for simple caches
- **Use cases** : Caching, session management, gaming leader boards, geospatial applications



ElastiCache

# Databases - Summary

In 28  
Minutes

Database Type	AWS Service	Description
Relational OLTP databases	Amazon RDS	Row storage Transactional usecases needing <b>predefined schema</b> and very <b>strong transactional</b> capabilities
Relational OLAP databases	Amazon Redshift	Columnar storage Reporting, analytics & intelligence apps needing <b>predefined schema</b>
Document & Key Databases	Amazon DynamoDB	Apps needing <b>quickly evolving</b> semi structured data ( <b>schema-less</b> ) <b>Scale to terabytes of data with millisecond responses upto millions of TPS</b> Content management, catalogs, user profiles, shopping carts, session stores and gaming applications

# Databases - Summary

Database Type	AWS Service	Description
Graph Databases	Amazon Neptune	Store and navigate data with <b>complex relationships</b> Social Networking Data (Twitter, Facebook), Fraud Detection
In memory databases/caches	Amazon ElastiCache	Applications needing <b>microsecond</b> responses <b>Redis</b> - persistent data <b>Memcached</b> - simple caches

# Databases - Questions

Scenario	Solution
A start up with quickly evolving tables	DynamoDB
Transaction application needing to process million transactions per second	DynamoDB
Very high consistency of data is needed while processing thousands of transactions per second	RDS
Cache data from database for a web application	Amazon ElastiCache
Relational database for analytics processing of petabytes of data	Amazon Redshift

# Amazon RDS (Relational Database Service)

- Do you want to manage the setup, backup, scaling, replication and patching of your relational databases?
  - Or do you want to use a managed service?
- Amazon RDS is a managed relational database service for OLTP use cases
- Supports:
  - Amazon Aurora
  - PostgreSQL
  - MySQL (InnoDB storage engine full supported)
  - MariaDB (Enhanced MySQL)
  - Oracle Database
  - Microsoft SQL Server



Amazon RDS

# Amazon RDS - Features

- Multi-AZ deployment (standby in another AZ)
- Read replicas:
  - Same AZ
  - Multi AZ (Availability+)
  - Cross Region(Availability++)
- Storage auto scaling (up to a configured limit)
- Automated backups (restore to point in time)
- Manual snapshots



Amazon RDS

# Amazon RDS - You vs AWS

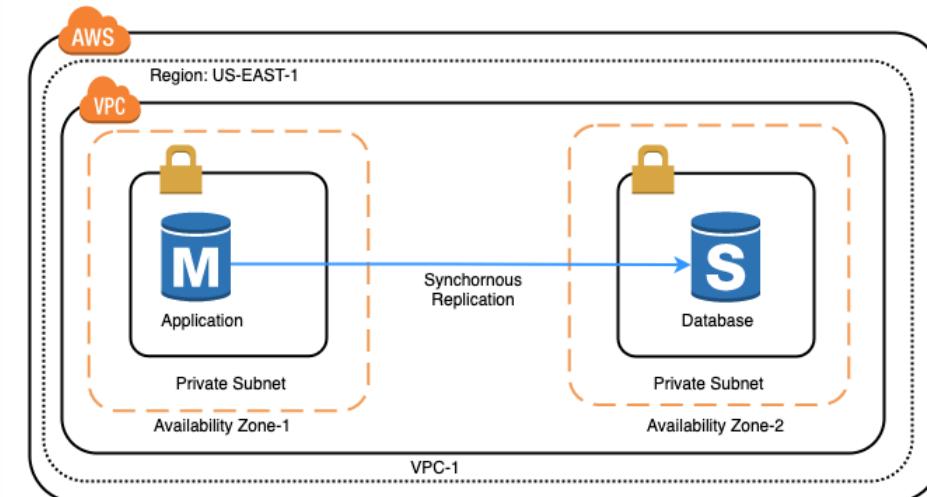
- AWS is responsible for
  - Availability (according to your configuration)
  - Durability
  - Scaling (according to your configuration)
  - Maintenance (patches)
  - Backups
- You are responsible for
  - Managing database users
  - App optimization (tables, indexes etc)
- You CANNOT
  - SSH into database EC2 instances or setup custom software (NOT ALLOWED)
  - Install OS or DB patches. RDS takes care of them (NOT ALLOWED)



Amazon RDS

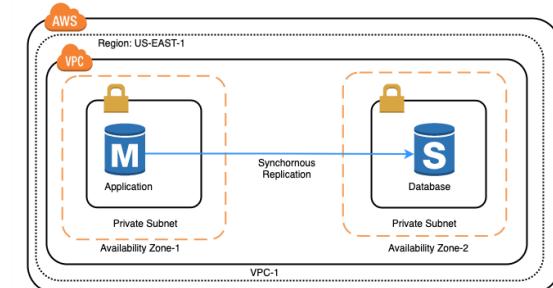
# Multi-AZ Deployments

- Standby created in a different AZ
- **Synchronous replication** (strong consistency)
- Enhances durability, availability and fault tolerance of your database
- Multi-AZ makes **maintenance easy**
  - Perform maintenance (patches) on standby
  - Promote standby to primary
  - Perform maintenance on (old) primary
- **Avoid I/O suspension** when data is backed up (snapshots are taken from standby)



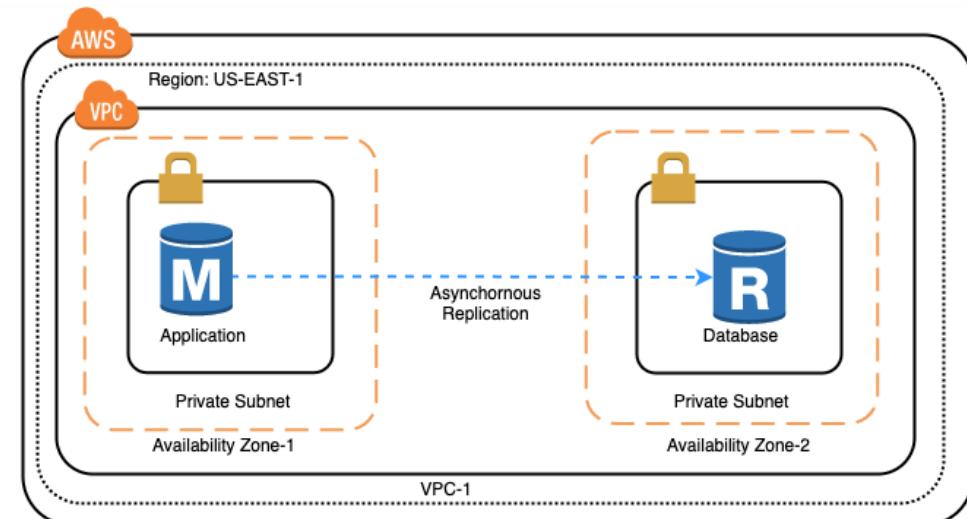
# Multi-AZ Deployments

- No downtime when database is converted to Multi AZ
  - Increased latency until standby is ready
- Not allowed to connect to standby database directly
  - For example: Standby CANNOT be used to serve read traffic
  - Standby increases availability but does not improve scalability
- Automatic failover to standby if master has problems (compute, storage or networking)
  - CNAME record flipped to standby
  - Database performance issues (long running queries or deadlocks) will NOT cause a failover
- (Good Practice) Use DNS name of database in applications configuration



# Read Replicas

- Support **read-heavy database workloads** - reporting and data warehousing
- Can be in same or different AZ or different Region
- Your apps can connect to them
- Create read replica(s) of a read replica
- Uses **asynchronous replication**
  - Provides eventual consistency (from replica)
  - For higher consistency, read from master
- Need to be **explicitly deleted** (Not deleted when database is deleted)



# Read Replicas - Few Tips

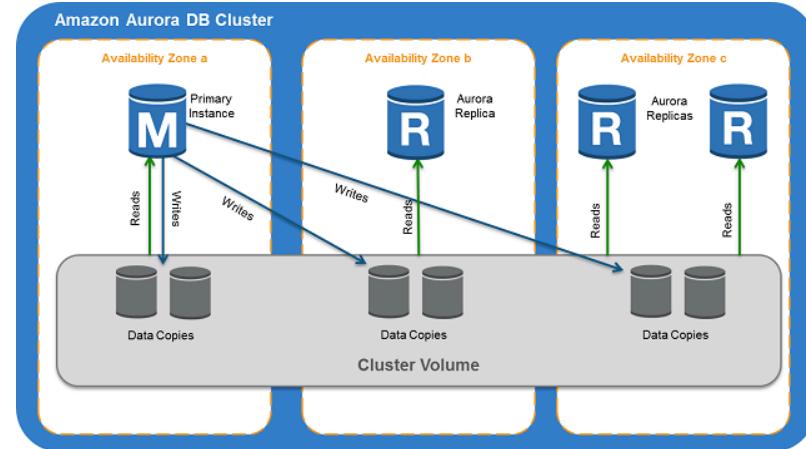
- (Mandatory) Enable automatic backups before you can create read replicas
  - Set Backup Retention period to a value other than 0
- Reduce replication lag by using better compute and storage resources
- Maximum no of read replicas:
  - MySQL, MariaDB, PostgreSQL, and Oracle - 5
  - Aurora - 15
  - SQL Server does not support read replicas

# Multi-AZ vs Multi-Region vs Read replicas

Feature	Multi-AZ deployments	Multi-Region Read Replicas	Multi-AZ Read replicas
Main purpose	High availability	Disaster recovery and local performance	Scalability
Replication	Synchronous (except for Aurora - Asynchronous)	Asynchronous	Asynchronous
Active	Only master (For Aurora - all)	All read replicas	All read replicas

# Amazon Aurora

- MySQL and PostgreSQL-compatible
- 2 copies of data each in a minimum of 3 AZ
- Up to 15 read replicas (Only 5 for MySQL)
- Provides "Global Database" option
  - Up to five read-only, secondary AWS Regions
    - Low latency for global reads
    - Safe from region-wide outages
  - Minimal lag time, typically less than 1 second
- Deployment Options
  - Single master (One writer and multiple readers)
  - Multi master deployment (multiple writers)
  - Serverless
- Uses cluster volume (multi AZ storage)



<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/>

# RDS - Scaling

- Vertical Scaling: Change DB instance type and scale storage
  - Storage and compute changes are typically applied during maintenance window
  - You can also choose to “apply-immediately”
  - RDS would take care of data migration
    - Takes a few minutes to complete
  - You can manually scale Aurora, MySQL, MariaDB, Oracle, and PostgreSQL engines to 64 TB
  - SQL Server can be scaled up to 16 TB
- Vertical Scaling: RDS also supports auto scaling storage
- Horizontal Scaling
  - Configure Read Replicas
  - For Aurora (Multi-master, Writer with multiple readers etc)



Amazon RDS

# RDS - Operations

- RDS console shows metrics upto a certain time period
- CloudWatch show historical data
- Configure CloudWatch alarms to alert when you near max capacity
- Enable Enhanced Monitoring to monitor slow queries
- Automatic backup during backup window (to Amazon S3)
  - Enables restore to **point in time**
  - Backups retained for 7 days by default (max - 35 days)
  - Elevated latency when snapshots are taken (except for Multi-AZ setup)
- **Backup window used to apply patches**
  - If you do not configure a 30 minute backup window, RDS chooses one randomly
- Achieve RPO of up to 5 minutes



Amazon RDS



Cloudwatch



AWS Config

# RDS - Security and Encryption

- Create in a VPC private subnet
- Use security groups to control access
- Option to use IAM Authentication with Aurora, MySQL and PostgreSQL
  - Use IAM roles and no need for passwords
- Enable encryption with keys from KMS
- When encryption is enabled
  - Data in the database, automated backups, read replicas and snapshots are all encrypted
- Data In-flight Encryption
  - Using SSL certificates



AWS KMS



Subnet



Amazon RDS



Security Group

# RDS - Costs - Key Elements

- **DB instance hours** - How many hours is the DB instance running?
- **Storage (per GB per month)** - How much storage have you provisioned for your DB instance?
- **Provisioned IOPS per month** - If you are using Amazon RDS Provisioned IOPS (SSD) Storage
- **Backups and snapshot storage (across multi AZ)** - More backups, More snapshots => More cost
- **Data transfer costs**

# Amazon RDS - When to use?

- Use Amazon RDS for transactional applications needing
  - Pre-defined schema
  - Strong transactional capabilities
  - Complex queries
- Amazon RDS is **NOT recommended** when
  - You need highly scalable massive read/write operations - for example millions of writes/second
    - Go for DynamoDB
  - When you want to upload files using simple GET/PUT REST API
    - Go for Amazon S3
  - When you need heavy customizations for your database or need access to underlying EC2 instances
    - Go for a custom database installation



Amazon RDS

# RDS - Scenarios

In 28  
Minutes

Scenario	Solution
<b>You want full control of OS or need elevated permissions</b>	Consider going for a custom installation (EC2 + EBS)
<b>You want to migrate data from an on-premise database to cloud database of the same type</b>	Consider using AWS Database Migration Service
<b>You want to migrate data from one database engine to another (Example : Microsoft SQL Server to Amazon Aurora)</b>	Consider using AWS Schema Conversion Tool
<b>What are retained when you delete a RDS database instance?</b>	All automatic backups are deleted All manual snapshots are retained (until explicit deletion) (Optional) Take a final snapshot

# RDS - Scenarios

In 28  
Minutes

Scenario	Solution
<b>How do you reduce global latency and improve disaster recovery?</b>	Use multi region read replicas
<b>How do you select the subnets a RDS instance is launched into?</b>	Create DB Subnet groups
<b>How can you add encryption to an unencrypted database instance?</b>	Create a DB snapshot Encrypt the database snapshot using keys from KMS Create a database from the encrypted snapshot
<b>Are you billed if you stop your DB instance?</b>	You are billed for storage, IOPS, backups and snapshots. You are NOT billed for DB instance hours
<b>I will need RDS for at least one year. How can I reduce costs?</b>	Use Amazon RDS reserved instances.
<b>Efficiently manage database connections</b>	Use Amazon RDS Proxy Sits between client applications (including lambdas) and RDS

# Amazon DynamoDB

# Amazon DynamoDB

- Fast, scalable, **distributed** for any scale
- Flexible **NoSQL** Key-value & document database (schemaless)
- **Single-digit millisecond responses for million of TPS**
- Do not worry about scaling, availability or durability
  - Automatically partitions data as it grows
  - Maintains 3 replicas within the same region
- No need to provision a database
  - Create a table and configure read and write capacity (RCU and WCU)
  - Automatically scales to meet your RCU and WCU
- Provides an **expensive serverless mode**
- **Use cases:** User profiles, shopping carts, high volume read write applications



DynamoDB

# DynamoDB Tables

- Hierarchy : Table > item(s) > attribute (key value pair)
- Mandatory primary key
- Other than the primary key, tables are **schemaless**
  - No need to define the other attributes or types
  - Each item in a table can have distinct attributes
- Max 400 KB per item in table
  - Use S3 for large objects and DynamoDB for smaller objects
- DynamoDB Tables are region specific.
  - If your users are in multiple regions, mark the table as **Global Table**
  - Replicas are created in selected regions

```
{  
  "id": 1,  
  "name": "Jane Doe",  
  "username": "abcdefgh",  
  "email": "someone@gmail.com",  
  "address": {  
    "street": "Some Street",  
    "suite": "Apt. 556",  
    "city": "Hyderabad",  
    "zipcode": "500018",  
    "geo": {  
      "lat": "-3.31",  
      "lng": "8.14"  
    }  
  },  
  "phone": "9-999-999-9999",  
  "website": "in28minutes.com",  
  "company": {  
    "name": "in28minutes"  
  }  
}
```

# Data Types

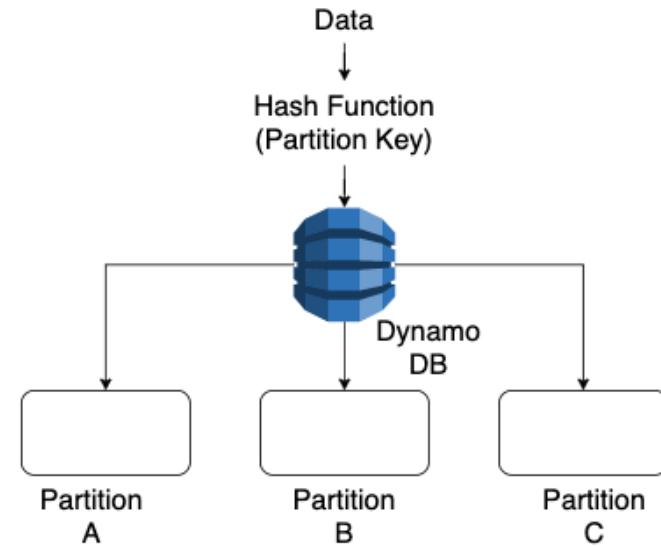
- **Scalar** (one value) - String, Number, Binary (base 64 encoded), Boolean (true or false) and null (unknown or undefined state)
  - Use String to represent date or timestamp (2035-12-21T17:42:34Z)
- **Document** (List and Map)
  - Supports complex JSON structures
  - List Example - Todos: ["Learn to Dance", "Get Certified in AWS, Azure and Google Cloud", 3.14159]
  - Map Example - {Day: "Monday", UnreadEmails: 42}
  - List and Map can have different data types
- **Set** (multiple values) - String set, Number set, and Binary set
  - All elements of the same scalar type
  - Each value within a set must be unique
  - Order is not important



DynamoDB

# DynamoDB - Primary Key

- Two Types
  - Simple: Partition Key (or Hash)
  - Composite: Partition Key + Sort Key (or Hash + Range)
- Primary key should be **unique** (Cannot be changed later)
- Partition key **decides the partition** (input to hash function)
  - Same partition key items stored together (sorted by sort key, if it exists)
  - Choose a partition key that helps you to distribute items evenly across partitions:
    - Prefer High Cardinality
    - Append a Random Value (if needed)



# DynamoDB - Secondary Indexes - Local secondary index

- Example:
  - Attributes
    - CustomerId, OrderId, OrderCreationDt, ProductDetails, OrderStatus, etc
  - Primary Key
    - CustomerId (partition key) + OrderId (sort key)
  - LSI
    - CustomerId (partition key) + OrderCreationDt OR
    - CustomerId (partition key) + OrderStatus
- Same partition key as primary key but different sort key
- Defined at table creation (Modifications NOT allowed)
- Up to 5 per table (HARD LIMIT)

# DynamoDB - Secondary Indexes - Global secondary index

- Example
  - Attributes
    - storeCode, city, state, country, address, etc..
  - Primary Key - storeCode
  - GSI - state or city or country
- Partition and sort key can be different from primary key
- Can be added, modified & removed later
- Stored separately (Separate RCU and WCU) from the main table
- Up to 20 per table (Reach out to AWS to increase it)
- (Recommended) Project fewer attributes to save cost
- (Recommended) Avoid throttling on main table - Assign RCU and WCU at least equal to main table

# DynamoDB Consistency Levels

- Eventually Consistent Reads
  - You might not get the latest data
  - If tried after few seconds, you will get the latest data
- Strongly Consistent Reads
  - Return the most up-to-date data reflecting the updates from all the previous successful write operations
  - Disadvantages:
    - Returns 500 error in case of network delay
    - May have higher latency
    - Not supported on Global Secondary Indexes
    - Uses more throughput capacity units



DynamoDB

# DynamoDB Consistency Levels

- By default, eventually consistent (lag of about a second)
- Request for strongly consistent reads
  - Set `ConsistentRead` to true
  - Slow and more expensive
- Supports transactions (`TransactWriteItems`, `TransactGetItems`)
  - All-or-nothing changes to multiple items (`PutItem`, `UpdateItem` and `DeleteItem` operations) both within and across tables
  - More expensive



DynamoDB

# DynamoDB Read/Write Capacity Modes

- Provisioned
  - Provision read (RCU) and write (WCU) capacity needed per second
  - Dynamically adjustable
  - Unused capacity can be used in bursts
  - You are billed for the provisioned capacity irrespective of whether you make use of it or not
- On Demand
  - Truly serverless and expensive
  - For unknown workloads or traffic with huge spikes
  - Use On Demand only when:
    - Workloads are really spiky causing low utilization of Provisioned Capacity OR
    - Usage is very low (for example, in test environments) making manual adjustments expensive



DynamoDB

# DynamoDB Read/Write Capacity Calculations

Minimum RCU/WCU	Operations
1 RCU	Up to 2 Eventually Consistent Reads(1 item up to 4KB) per second
1 RCU	1 Strongly Consistent Read(1 item up to 4KB) per second
2 RCU	1 Transactional Read(1 item up to 4KB) per second
1 WCU	1 Standard Write per second (1 item up to 1KB)
2 WCU	1 Transactional Write per second(1 item up to 1KB)



DynamoDB

# Read Capacity Unit - One Operation Per Second

Size	Eventual Consistent Reads	Strongly Consistent Reads	Transactional Reads
1 KB	1 RCU	1 RCU	2 RCU
2 KB	1 RCU	1 RCU	2 RCU
3 KB	1 RCU	1 RCU	2 RCU
4 KB	1 RCU	1 RCU	2 RCU
5 KB	1 RCU	2 RCU	4 RCU
6 KB	1 RCU	2 RCU	4 RCU
7 KB	1 RCU	2 RCU	4 RCU
8 KB	1 RCU	2 RCU	4 RCU
15 KB	2 RCU	4 RCU	8 RCU



DynamoDB

# Write Capacity Unit - One Operation Per Second

Size	Standard Writes	Transactional Writes
1 KB	1 WCU	2 WCU
1.5 KB	2 WCU	4 WCU
2 KB	2 WCU	4 WCU
2.5 KB	3 WCU	6 WCU
3 KB	3 WCU	6 WCU



DynamoDB

# Read & Write Request Units - Calculations

- How many RCU are needed to support 25 strongly consistent reads per second of 15KB?
  - 1 RCU is need to support 1 strongly consistent read of 4 KB
  - $15/4$  approximates 3.7 rounded up to 4. So we need 4 RCU to read 15 KB
  - For 25 strongly consistent read then  $25*4 = 100$  RCU
- How many RCU are needed to support 25 eventually consistent reads per second of 15KB?
  - 0.5 RCU is need to support 1 eventually consistent read of 4 KB
  - $15/4$  approximates 3.7 rounded up to 4. So we need 2 RCU to read 15 KB
  - For 25 eventually consistent read then  $25*2 = 50$  RCU
- How many WCU are needed to support 100 writes per second of 512 Bytes?
  - 1 WCU is need to support 1 write of 1 KB. Here it is 512 so 0.5 WCU rounded up so 1 WCU.
  - $100*1 = 100$  WCU needed

# DynamoDB Query vs Scan

- **Query**
  - Search using a partition key attribute and a distinct value to search
  - Optional - sort key and filters
- **Scan**
  - Reads every item in a table
  - Expensive compared to query
  - Returns all attributes by default (Recommended to use ProjectionExpression to return selected attributes)
  - Parallel Scan option available (Divides table/index into segments). Recommended for large tables (>20 GB) in situations where RCU is NOT being fully used.



DynamoDB

# Projection and Filter expressions

```
aws dynamodb scan --table-name MyTodos \
    --projection-expression "username"

aws dynamodb scan --table-name MyTodos \
    --projection-expression "username, #desc" \
    --expression-attribute-names '{"#desc":"desc"}'

aws dynamodb scan --table-name MyTodos \
    --filter-expression "begins_with(username,:username)" \
    --expression-attribute-values '{":username":{"S":"Ranga"}}'
```

- **Projection Expression** - Retrieve selected attributes in an API Operation
- Use in combination with **expression-attribute-names** if projected attributes contain keywords. Begins with # (hash).
- **Filter Expression** - Filter items based on condition
  - (REMEMBER) You pay for scanned records(ScannedCount). NOT just filtered records(Count).
- **Expression attribute values** - Compare attribute with value.
  - Begins with :(colon). Example : {":username": {"S": "Ranga"}}

# Pagination

```
aws dynamodb scan --table-name MyTodos \
    --max-items 10

aws dynamodb scan --table-name MyTodos \
    --max-items 100
    --starting-token eyJFeGNsdXNpdmVTdGFydEtleSI6IG51bGwsICJib3RvX3RydW5jYXRlx:
```

- **--max-items**: total number of items to return in the command's output. If there are more than max items, a token is provided in the output.
- **--starting-token**: A token to specify where to start paginating. This is the NextToken from a previously truncated response.
- (CAUTION) **--page-size** : Size of each page to get in the AWS service call. Does not affect the number of items returned. Prevent AWS service calls from timing out.

# DynamoDB API's

API	Description
<b>Query</b>	Query a Table, LSI or GSI using partition key and (optional) sort key. Supports FilterExpression, Pagination and ProjectionExpression.
<b>Scan</b>	Retrieves all items in the specified table or index. Supports FilterExpression, Pagination and ProjectionExpression.
<b>GetItem</b>	Retrieve single item. Primary key mandatory. Retrieve entire item, or subset of attributes.
<b>BatchGetItem</b>	Retrieve up to 100 items from multiple tables. Avoid network round trips.
<b>PutItem</b>	Write one Item. Primary key mandatory.
<b>UpdateItem</b>	Modify one or more attributes in an item. Primary key mandatory. Add/update new/existing attributes.
<b>DeleteItem</b>	Delete single item using primary key
<b>BatchWriteItem</b>	Put/Delete upto 25 items to multiple tables. Reduces network round trips.

# DynamoDB API Command Examples

```
aws dynamodb get-item --table-name MyTodos \
    --key '{"id": {"S": "2"}'"

aws dynamodb update-item --table-name MyTodos \
    --key '{"id": {"S": "2"}' \
    --update-expression "SET username = :u, newattr=:u" \
    --expression-attribute-values '{":u":{"S":"RangaABC"}}'

aws dynamodb update-item --table-name MyTodos \
    --key '{"id": {"S": "2"}' \
    --update-expression "REMOVE newattr" \
    --expression-attribute-values '{":u":{"S":"RangaABC"}}'

aws dynamodb update-item --table-name MyTodos \
    --key '{"id": {"S": "2"}' \
    --update-expression "SET #U = :u" \
    --expression-attribute-values '{":u":{"S":"RangaABCD"}}' \
    --expression-attribute-names '{"#U":"username"}'

aws dynamodb delete-item --table-name MyTodos \
    --key '{"id": {"S": "2"}' \
    --condition-expression "begins_with(username,:username)" \
    --expression-attribute-values '{":username":{"S":"RangaABC"}}'
```

# Conditional expression

```
aws dynamodb delete-item --table-name MyTodos \
    --key '{"id": {"S": "2"}}' \
    --condition-expression "#desc=:desc" \
    --expression-attribute-names '{"#desc":"desc"}' \
    --expression-attribute-values '{":desc":{"S":"Learn to Dance"}}'
```

- Update/delete an item only if the condition expression is true
- Enables you to check for consistency of update/delete
  - Error message if check fails - An error occurred (ConditionalCheckFailedException) when calling the DeleteItem operation: The conditional request failed

# DynamoDB API Errors

- 200 OK - Successful
- 4XX - Problem with request (authentication failure, exceeding a table's provisioned throughput etc)
  - ConditionalCheckFailedException : Conditional update evaluated to false.
  - ProvisionedThroughputExceededException: Exceeded provisioned RCU/WCU for Table/GSI (Use CloudWatch to analyze and Retry. Also called Throttling.).
  - ThrottlingException: (Mostly) You are doing too many table operations (CreateTable/DeleteTable/UpdateTable).
- 5XX - Problem that must be resolved by AWS
  - In most cases, just a retry should solve the problem!
- For BatchGetItem / BatchWriteItem, UnprocessedKeys / UnprocessedItems contain individual failed requests:
  - Status of the operation will be successful even if at least one individual request is successful.

# TTL (Time To Live)

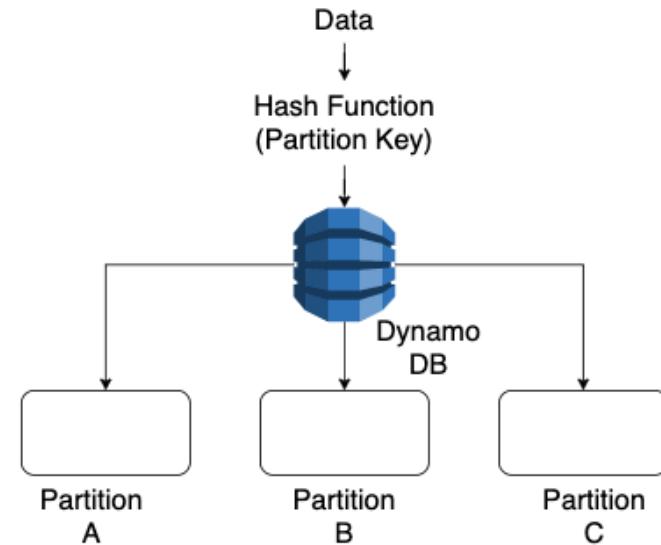
- Imagine storing millions of records in a table.
- Let's say you want to expire records after a certain time period.
  - Ex: Remove user data after two years of inactivity
- Mark specific attribute in a table as TTL attribute (should use timestamp format)
  - Item would be deleted if TTL attribute value older than the current time but not five years older or more (avoid accidental deletion)
  - There could be time delay between expiration time and actual deletion time
  - Does not consume WCU



DynamoDB

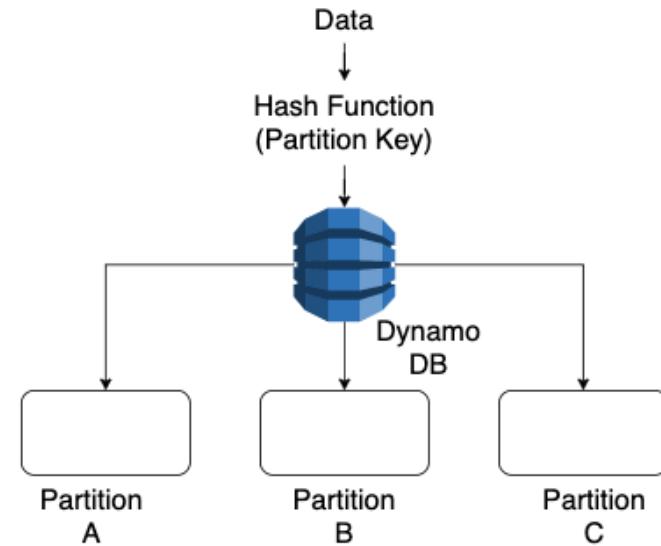
# Designing Partition Keys

- Each partition - Max RCU 3000, Max WCU 1000
- DynamoDB divides provisioned I/O capacity divided evenly among partitions
- For good performance, distribute items evenly across partitions.
  - Prefer attributes that have mostly unique values
- Bad Uniformity:
  - Client Type: Only 4 types of clients
  - Creation date, rounded to the nearest time period (for example, day, hour, or minute)
- Good Uniformity:
  - Product ID which is unique for each Item
  - Write Sharding Using Suffixes



# Time Series Data

- Problem:
  - Tracking a high volume of events
  - Write Access Pattern: All events have today's date
  - Read Access Pattern: Recent events are accessed more frequently. Events older than couple of days very rarely accessed.
  - You are very cost sensitive!
- Solution:
  - Create a new table for each period.
  - As table gets older, reduce its WCU to minimum.  
Reduce RCU to appropriate value.



# DynamoDB - Optimistic Locking

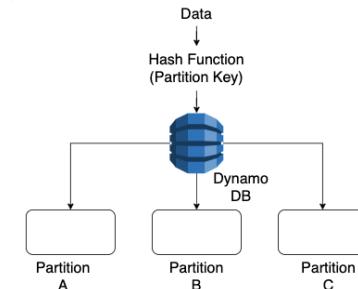
- What is Optimistic Locking?
  - Protect your writes by ensuring that they are not overwritten
  - Check the value of your item before update
- Implementation Option:
  - AWS SDK for Java provides @DynamoDBVersionAttribute annotation
- Implementation Logic:
  - Designate one property in your item to store the version number
  - Check and update version in every update
  - For example: On item update, AWS SDK for Java would throw ConditionalCheckFailedException if value on server is different from value on client side (if optimistic locking with @DynamoDBVersionAttribute is enabled)



DynamoDB

# DynamoDB Best Practices

- Different design approach (compared to Relational Databases):
  - Minimum number of tables
  - Understand access patterns to create primary key and secondary indices
  - Cost and Time involved
- Avoid Scans as much as possible. At least avoid sudden spikes:
  - Reducing page size
  - Isolating scan operations by duplicating content in multiple tables
- Ensure that you are not exceeding the limits on a specific partition keys (Avoid hot keys and hot partitions)
- AWS SDK uses Error Retries and Exponential Backoff (1st retry - 50 ms, 2nd retry 100 ms, ..)
  - For custom implementations you need to implement retry



# IAM Policy - Enforce MFA

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {"Effect": "Allow",  
         "Action": [  
             "service-prefix-1:*",  
         ],  
         "Resource": "*",  
         "Condition": {  
             "Bool": {"aws:MultiFactorAuthPresent": true}  
         }  
     }  
}
```

- Only allow requests authenticated with MFA
  - "Bool": {"aws:MultiFactorAuthPresent": true}

# IAM Policy - Limits User Access on DynamoDB

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:DeleteItem", "dynamodb:GetItem", "dynamodb:PutItem",  
                "dynamodb:Query", "dynamodb:UpdateItem"  
            ],  
            "Resource": ["arn:aws:dynamodb:*:*:table/MyTable"],  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]  
                }  
            }  
        }  
    ]  
}
```

- Use dynamodb:LeadingKeys condition key to limit user actions:
  - Allow access only to items partition key value matches Cognito user id

# IAM Policy - Limit User Access on S3 Bucket

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket",  
            "Resource": ["arn:aws:s3:::BUCKET-NAME"],  
            "Condition": {"StringLike": {"s3:prefix": ["${cognito-identity.amazonaws.com:sub}"]}}  
        },  
        {  
            "Sid": "ReadWriteDeleteYourObjects",  
            "Effect": "Allow",  
            "Action": ["s3:GetObject", "s3:PutObject", "s3>DeleteObject"],  
            "Resource": [  
                "arn:aws:s3:::BUCKET-NAME/${cognito-identity.amazonaws.com:sub}",  
                "arn:aws:s3:::BUCKET-NAME/${cognito-identity.amazonaws.com:sub}/*"  
            ]  
        }  
    ]  
}
```

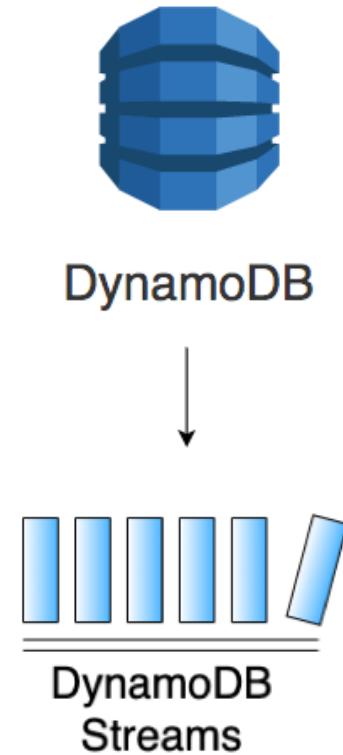
- You can use AWS user name as well - \${aws:username}

# DynamoDB - Remember

- Reach out to AWS if you want RCU or WCU > 10,000 Units for a table
- Global Table needs DynamoDB streams
- *"A filter is applied after Query/Scan finishes, but before results are returned. A Query/Scan consumes the same amount of read capacity with/without filter."*
- Other DynamoDB APIs:
  - CreateTable, DescribeTable, ListTables, UpdateTable, DeleteTable - Table operations
  - TransactWriteItems, TransactGetItems - Batch operations with transactions
- Options to access DynamoDB:
  - AWS Management Console
  - AWS CLI
  - SDKs
  - NoSQL Workbench for DynamoDB

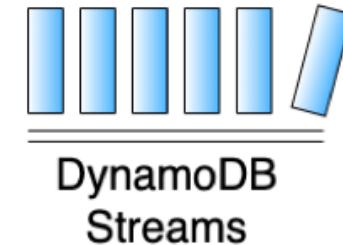
# DynamoDB Streams

- DynamoDB streams captures time ordered sequence of item modifications
  - Stored up to 24 hours
- If DynamoDB stream is enabled:
  - On create, update or delete of an item, DynamoDB writes a stream record in near real time
- StreamViewType decides what is captured on the stream record (KEYS\_ONLY, NEW\_IMAGE, OLD\_IMAGE, NEW\_AND\_OLD\_IMAGES)
- A different endpoint is created for DynamoDB Streams
  - One for DynamoDB ([dynamodb.amazonaws.com](https://dynamodb.amazonaws.com))
  - One for DynamoDB Streams ([streams.dynamodb.amazonaws.com](https://streams.dynamodb.amazonaws.com))



# DynamoDB Streams - Record & Shard

- Stream record represents a single data modification in the table.
  - Has a sequence number reflecting the order
- Stream records organized in to a group called Shards.
  - Shards are ephemeral. They are created and deleted automatically.
  - Disabling a stream, closes any open shards
- AWS SDK provides separate clients for DynamoDB and DynamoDB streams.
- If you want to process a Stream from a Lambda:
  - Create an event source mapping to tell Lambda to send records from your stream to a Lambda function



# DynamoDB - Operations

- Performance Monitoring - CloudWatch
- Alerts on RCU, WCU and Throttle Requests - CloudWatch Alarms
- Migrate data from RDS or MongoDB to DynamoDB - AWS Database Migration Service
- (Feature) Enable point-in-time recovery (max 35 days)
- Use Time to Live (TTL) to automatically expire items



DynamoDB

# DynamoDB - IAM and Encryption

- Server-side encryption in integration with keys from KMS
  - Always enabled
  - Automatically encrypts tables, DynamoDB streams, and backups
- Client-side encryption with DynamoDB Encryption Client
  - You can manage your keys with KMS or CloudHSM
- Use IAM roles to provide EC2 instances or AWS services access to DynamoDB tables
  - Predefined policies available for DynamoDB
    - AmazonDynamoDBReadOnlyAccess
    - AmazonDynamoDBFullAccess etc
  - Fine-grained control at the individual item level
- Does NOT support resource based policies



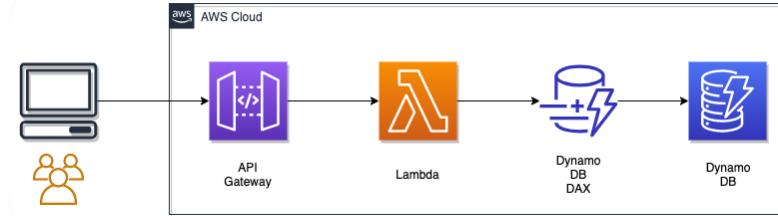
DynamoDB

# DynamoDB vs RDS

Feature	DynamoDB	RDS
Scenario	Millisecond latency with millions of TPS	Stronger consistency (schema) and transactional capabilities
Schema	Schemaless (needs only a primary key - Great for use cases where your schema is evolving)	Well-defined schema with relationships
Data Access	Using REST API provided by AWS using AWS SDKs or AWS Management Console or AWS CLI	SQL queries
Complex Data Queries Involving Multiple Tables	Difficult to run	Run complex relational queries with multiple entities
Scaling	No upper limits	64 TB
Consistency	Typically lower consistency	Typically higher consistency
Performance	Total Throughput (TPS) / Day	Total Throughput / Day

# DynamoDB Accelerator (DAX)

- In-memory caching for DynamoDB providing microsecond response times
  - Uses write-through (cache and database updated simultaneously)
  - Recommended for read intensive workloads especially for hot items (small no of items read more frequently)
- Very few changes needed to connect to DAX
  - Can reduce your costs by saving your read capacity units
- NOT recommended:
  - If you need strongly consistent reads or
  - Your application is write-intensive with very few reads



# DynamoDB - Scenario Questions

Scenario	Solution
<p>Your Provisioned RCU is not being completely utilized. How can you make scans on large tables (&gt;20 GB) faster?</p>	Use Parallel Scan option. Divides table/index into segments
<p>A table has an LSI, GSI configured. Main table is being throttled. However, WCU is available on the Main table. What could be wrong?</p>	If GSI is throttled on writes, then the main table will also be throttled as a result. Ensure that GSI WCU is at least equal to that of the table.
<p>You are infrequently getting ThrottlingExceptions</p>	Retry with exponential backoff
<p>You are not making use of all attributes returned from a query. You would want to make it more efficient</p>	Use a projection or Create an GSI index with few projected attributes.
<p>You would want to efficiently query on attribute which is not part of primary key</p>	Create a GSI

# DynamoDB - Scenario Questions - 2

Scenario	Solution
<b>Order Date is being used as partition key causing throttling in high load periods</b>	Append a random string to the partition key
You would want to delete all the million records from the table and reload again. You want to minimize RCU and WCU used.	Drop the table and recreate it!
<b>You want to store large images (0 to 100 MB)</b>	Store Images in S3 Store reference to S3 object (URL, metadata and/or keys) in DynamoDB

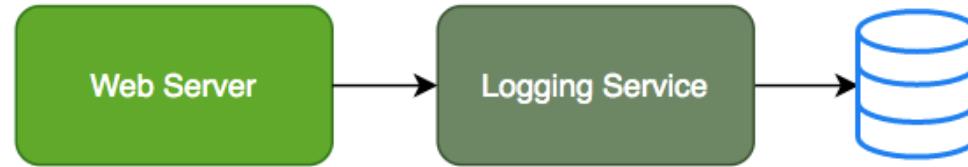
# Decoupling Applications with SQS, SNS and MQ

# Need for Asynchronous Communication

- Why do we need asynchronous communication?

# Synchronous Communication

In 28  
Minutes



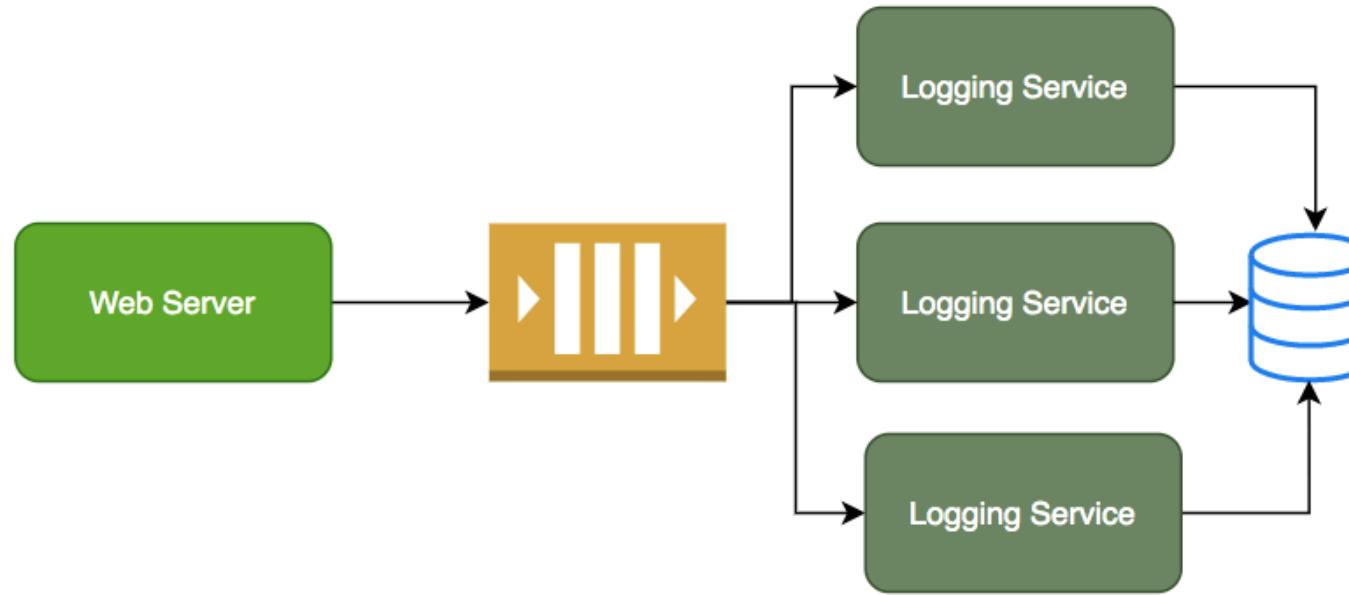
- Applications on your web server make synchronous calls to the logging service
- What if your logging service goes down?
  - Will your applications go down too?
- What if all of sudden, there is high load and there are lots of logs coming in?
  - Log Service is not able to handle the load and goes down very often

# Asynchronous Communication - Decoupled



- Create a queue or a topic
- Your applications put the logs on the queue
- They would be picked up when the logging service is ready
- Good example of decoupling!

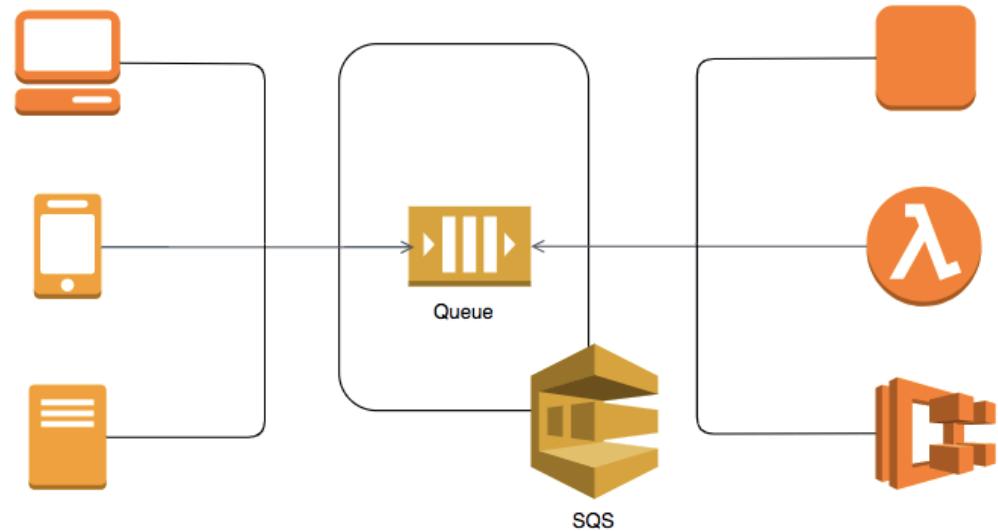
# Asynchronous Communication - Scale up



- You can have multiple logging service instances reading from the queue!

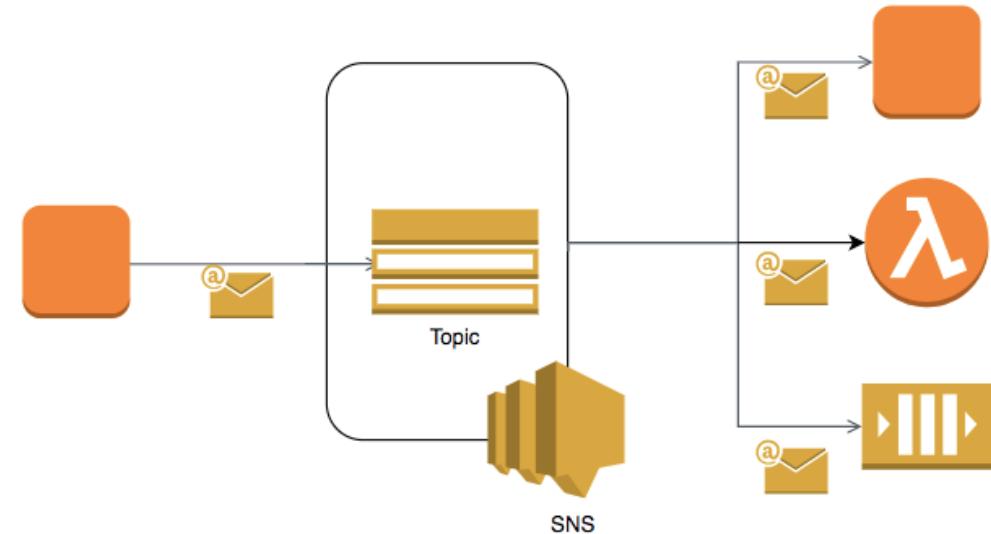
# Asynchronous Communication - Pull Model - SQS

- Producers put messages on the queue
- Consumers poll on the queue
  - Only one of the consumers will successfully process a given message
- Scalability
  - Scale consumer instances under high load
- Availability
  - Producer up even if a consumer is down
- Reliability
  - Work is not lost due to insufficient resources
- Decoupling
  - Make changes to consumers without effect on producers worrying about them



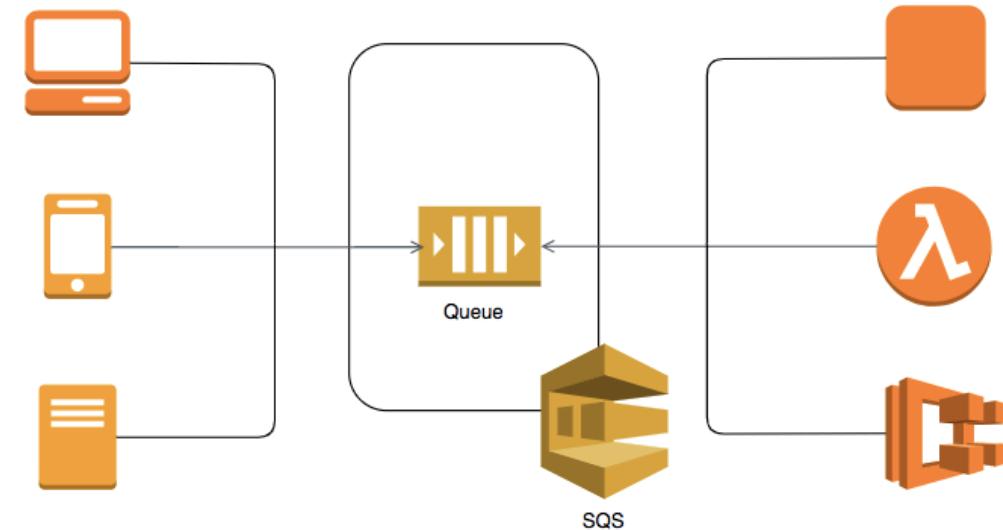
# Asynchronous Communication - Push Model - SNS

- Subscribers subscribe to a topic
- Producers send notifications to a topic
  - Notification sent out to all subscribers
- Decoupling
  - Producers don't care about who is listening
- Availability
  - Producer up even if a subscriber is down



# Simple Queuing Service

- Reliable, scalable, fully-managed message queuing service
- High availability
- Unlimited scaling
  - Auto scale to process billions of messages per day
- Low cost (Pay for use)



# Standard and FIFO Queues

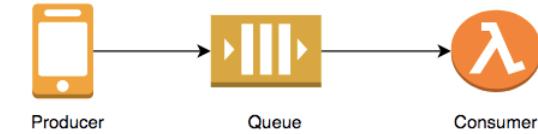
- Standard Queue
  - Unlimited throughput
  - BUT NO guarantee of ordering (Best-Effort Ordering)
  - and NO guarantee of exactly-once processing
    - Guarantees at-least-once delivery (some messages can be processed twice)
- FIFO (first-in-first-out) Queue
  - First-In-First-out Delivery
  - Exactly-Once Processing
  - BUT throughput is lower
    - Up to 300 messages per second (300 send, receive, or delete operations per second)
    - If you batch 10 messages per operation (maximum), up to 3,000 messages per second
- Choose
  - Standard SQS queue if throughput is important
  - FIFO Queue if order of events is important



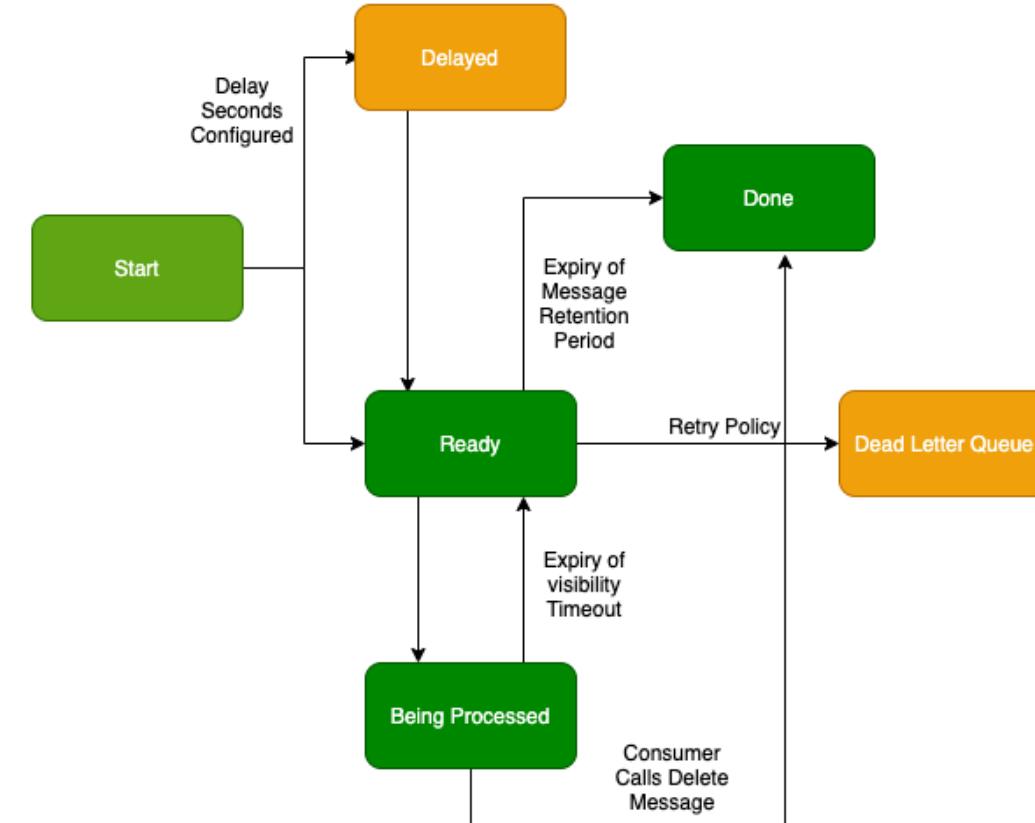
Amazon SQS

# Sending and receiving a SQS Message - Best case scenario

- Producer places message on queue
  - Receives globally unique message ID ABCDEFGHIJ (used to track the message)
- Consumer polls for messages
  - Receives the message ABCDEFGHIJ along with a receipt handle XYZ
- Message remains in the queue while the consumer processes the message
  - Other consumers will not receive ABCDEFGHIJ even if they poll for messages
- Consumer processes the message successfully
  - Calls delete message (using receipt handle XYZ)
  - Message is removed from the queue



# Simple Queuing Service Lifecycle of a message



# SQS - Auto Scaling



- Use target tracking scaling policy
- Use a SQS metric like ApproximateNumberOfMessages

# SQS Queue - Important configuration

Configuration	Description
<b>Visibility timeout</b>	<p>Other consumers will not receive a message being processed for the configured time period (default - 30 seconds, min - 0, max - 12 hours)</p> <p>Consumer processing a message can call <code>ChangeMessageVisibility</code> to increase visibility timeout of a message (before visibility timeout)</p>
<b>DelaySeconds</b>	<p>Time period before a new message is visible on the queue</p> <p>Delay Queue = Create Queue + Delay Seconds</p> <p>default - 0, max - 15 minutes</p> <p>Can be set at Queue creation or updated using <code>SetQueueAttributes</code></p> <p>Use message timers to configure a message specific <code>DelaySeconds</code> value</p>
<b>Message retention period</b>	<p>Maximum period a message can be on the queue</p> <p>Default - 4 days, Min - 60 seconds, Max - 14 days</p>
<b>MaxReceiveCount</b>	Maximum number of failures in processing a message

# Simple Queuing Service Security

In 28  
Minutes



- You can provide access to other AWS resources to access SQS using IAM roles (EC2 -> SQS)
- By default only the queue owner is allowed to use the queue
  - Configure SQS Queue Access Policy to provide access to other AWS accounts

# IAM Role - Trust Policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "abcdTrustPolicy",  
            "Effect": "Allow",  
            "Action": "sts:AssumeRole",  
            "Principal": {"Service": "ec2.amazonaws.com"}  
        }  
    ]  
}
```

- **Role's trust policy** - who or which service is allowed to assume the role.
- To be able to assume a Role using AssumeRole:
  - IAM User should have permission to AssumeRole
  - IAM Role (being assumed) should have trust policy allowing the IAM User
- This is also applicable to services using AssumeRole
  - For example, even EC2 service needs to be allowed by trust policy to assume role (NOT AUTOMATICALLY created when using CLI)

# SQS - Message deduplication

- Consider these scenarios:
  - Messages with identical message bodies. But you want SQS to treat them as **unique**.
  - Messages with identical content but different message attributes. But you want SQS to treat them as **unique**.
  - Messages sent with different content. But you want SQS to treat them as **duplicates**.
- How does FIFO Queue **identify a message as duplicate**?
  - **Content based:**
    - SQS generates a message deduplication ID using body of the message (BUT NOT the attributes of the message)
    - Recommended when you have an application specific unique id in the message
  - **Use message deduplication ID:**
    - Explicitly provide the message deduplication ID for the message
    - Example: Send MessageDeduplicationId along with SendMessage



Amazon SQS

# Amazon SQS - Important APIs

In 28  
Minutes

APIs	Description
<b>CreateQueue, DeleteQueue</b>	Create or Delete a Queue
<b>GetQueueAttributes, SetQueueAttributes</b>	Get or Set Attributes on a Queue (DelaySeconds, MessageRetentionPeriod, VisibilityTimeout etc )
<b>SendMessage</b>	Send a message to the Queue
<b>SendMessageBatch</b>	SendMessageBatch delivers up to ten messages at a time
<b>ReceiveMessage</b>	Retrieves one or more messages (up to 10), from the specified queue. Response include ReceiptHandle.
<b>ChangeMessageVisibility(ChangeMessageVisibilityBatch)</b>	Change visibility timeout of a specified message
<b>DeleteMessage (DeleteMessageBatch)</b>	Delete message using the ReceiptHandle
<b>PurgeQueue</b>	Delete all messages from a queue

# Amazon SQS - ReceiveMessage

- (REMEMBER) Receive upto 10 messages from the specified queue
  - MaxNumberOfMessages (1 to 10) - Maximum no of messages to receive.
  - WaitTimeSeconds - Enables Long Polling. Upto 20 Seconds
- Returns
  - Message body (and MD5 digest)
  - MessageId
  - Receipt handle.
  - Message attributes (and MD5 digest)
- (RECOMMENDED) Use Long Polling
  - (SAVE \$\$\$) Reduce the number of API call you need to make
  - (EFFICIENT) Reduce number of empty responses
  - (FASTER UPDATES) Receive messages as soon as they arrive in your queue

# SQS - Scenarios

In 28  
Minutes

Scenario	Result
Consumer takes more than visibility timeout to process the message	Message is visible on queue after visibility timeout and another consumer might receive the message
Consumer calls ChangeMessageVisibility before visibility timeout	Visibility timeout is extended to requested time
DelaySeconds is configured on the queue	Message is delayed for DelaySeconds before it is available
Receiver wants to decide how to handle the message without looking at message body	Configure Message Attributes

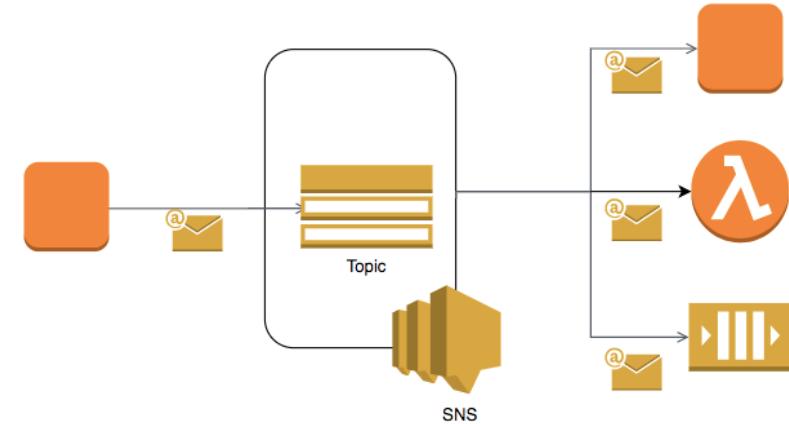
# SQS - Scenarios - 2

In 28  
Minutes

Scenario	Result
You are having problem processing the messages	Configure Dead Letter Queue
I want to send a large message (1GB) to SQS	Use Amazon SQS Extended Client Library. Upto 2 GB. Messages are stored in S3
How to reduce number of API calls to SQS?	Use Long Polling - When looking for messages, you can specify a WaitTimeSeconds upto 20 seconds
Your receive messages and start processing them after a week. You see that some messages are not processed at all!	Exceeded message retention period. Default message retention period is 4 days. Max 14 days.
Give high priority to premium customers	Create separate queues for free and premium customers

# Amazon Simple Notification Service(SNS)

- Publish-Subscribe (pub-sub) paradigm
- Broadcast asynchronous event notifications
- Simple process
  - Create an SNS Topic
  - Subscribers can register for a Topic
  - When an SNS Topic receives an event notification (from publisher), it is broadcast to all Subscribers
- Use Cases : Monitoring Apps, workflow systems, mobile apps



# Amazon Simple Notification Service(SNS)

- Provides mobile and enterprise messaging web services
  - Push notifications to Apple, Android, FireOS, Windows devices
  - Send SMS to mobile users
  - Send Emails
- REMEMBER : SNS does not need SQS or a Queue
- You can allow access to other AWS accounts using AWS SNS generated policy



Amazon SNS

# Amazon MQ

In 28  
Minutes

- Managed message broker service for Apache ActiveMQ
- (Functionally) Amazon MQ = Amazon SQS (Queues) + Amazon SNS (Topics)
  - BUT with restricted scalability
- Supports traditional APIs (JMS) and protocols (AMQP, MQTT, OpenWire, and STOMP)
  - Easy to migrate on-premise applications using traditional message brokers
  - Start with Amazon MQ as first step and slowly re-design apps to use Amazon SQS and/or SNS
- Scenario: An enterprise uses AMQP (standard message broker protocol). They want to migrate to AWS without making code changes
  - Recommend Amazon MQ

# Handling Data Streams

# Streaming Data

- Imagine implementing analytics for a website:
  - You have a continuous stream of data (page views, link clicks etc)
- Characteristics of streaming data:
  - Continuously generated
  - Small pieces of data
  - Sequenced - mostly associated with time
- How do you process continuous streaming data originating from application logs, social media applications?

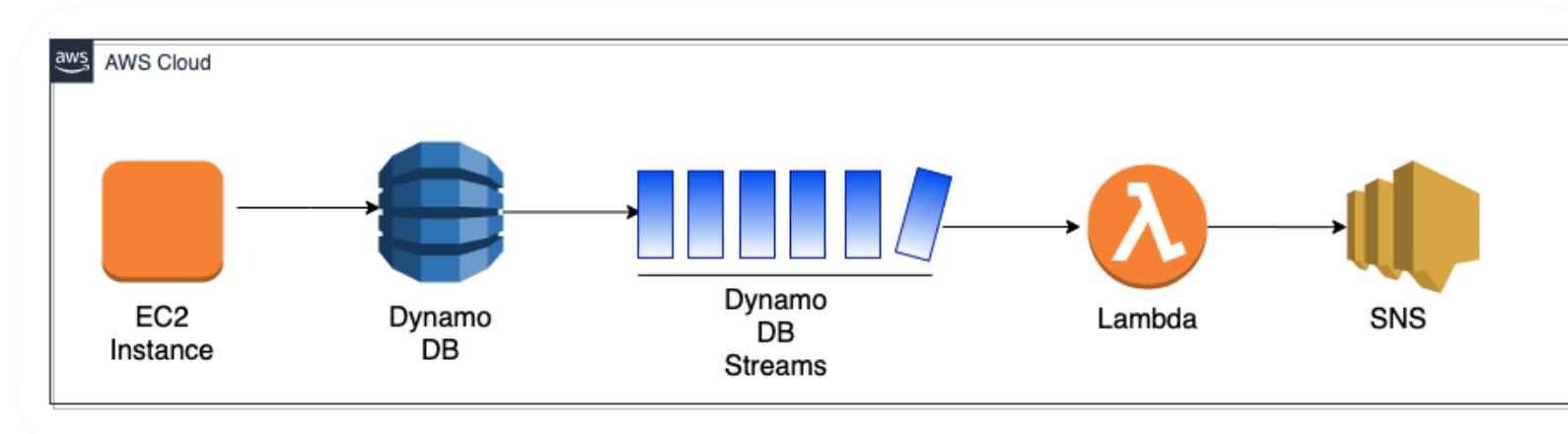


# S3 Notifications

- Send notifications to SNS, SQS, trigger lambda functions on
  - creation, deletion or update of an S3 object
- Setup at bucket level
  - You can use prefix and suffix to configure
- Cost efficient for simple use cases
  - S3 notification -> Lambda
  - Almost negligible cost (storage for file + invocation)



# DynamoDB Streams



- Each event from DynamoDB (in time sequenced order) is buffered in a stream near real-time
- Can be enabled or disabled
- Use case - Send email when user registers
  - Tie a Lambda function to DynamoDB Streams
- Stream allow iteration through records (**last 24 hours**)

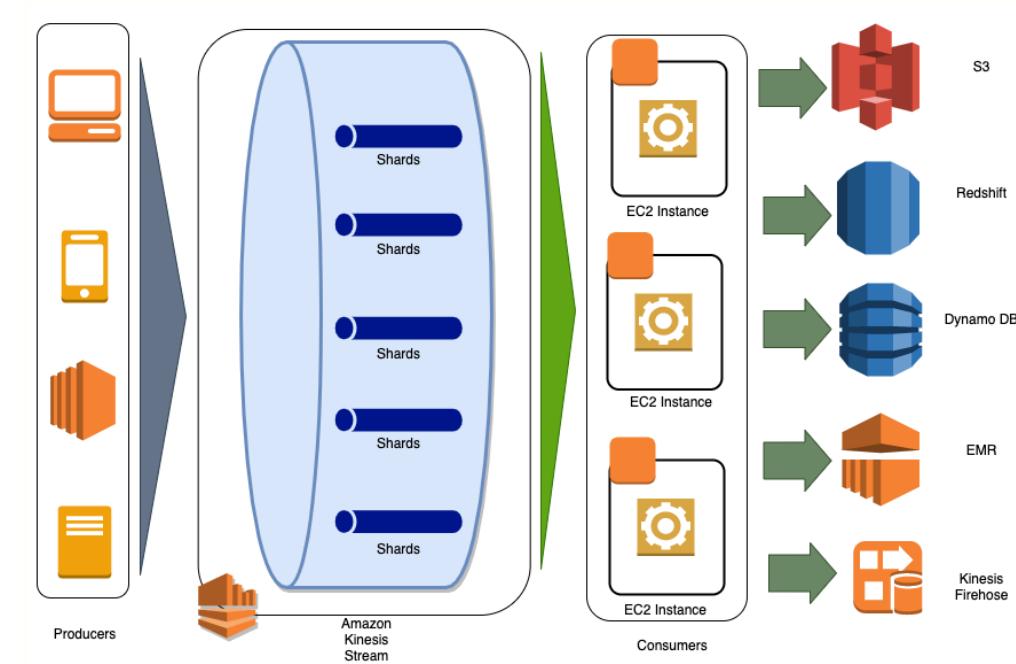
# Amazon Kinesis

- Handle streaming data
  - NOT recommended for ETL Batch Jobs
- Amazon Kinesis Data Streams
  - Process Data Streams
- Amazon Kinesis Firehose
  - Data ingestion for streaming data : S3, Elasticsearch etc
- Amazon Kinesis Analytics
  - Run queries against streaming data
- Amazon Kinesis Video Streams
  - Monitor video streams



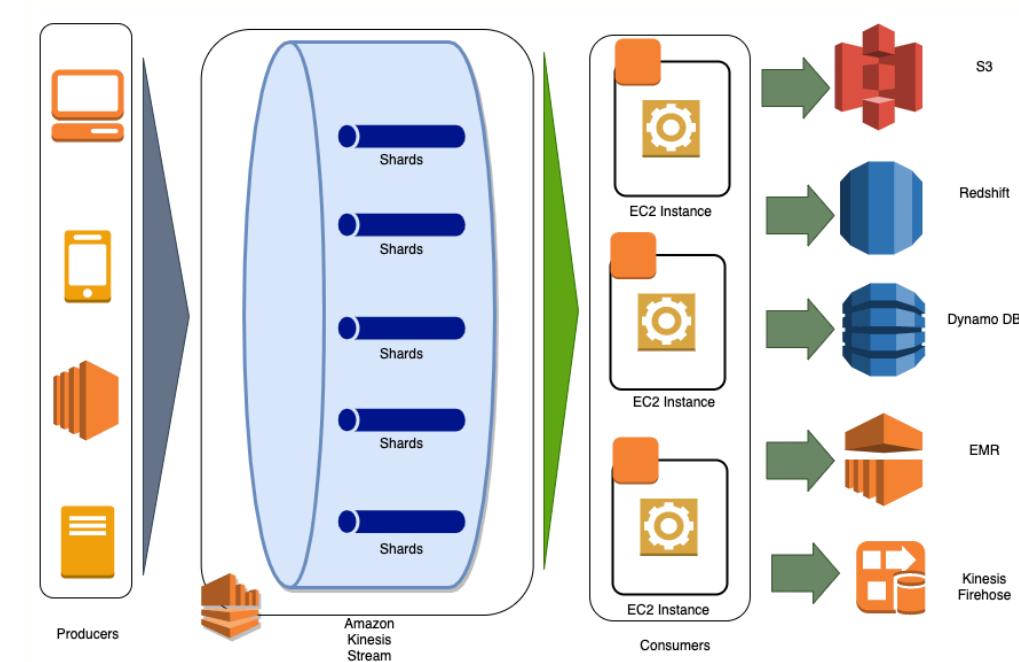
# Amazon Kinesis Data Streams

- Limitless Real time stream processing
  - Sub second processing latency
- Alternative for Kafka
- Supports multiple clients
  - Each client can track their stream position
- Retain and replay data (max 7 days & default 1 day)



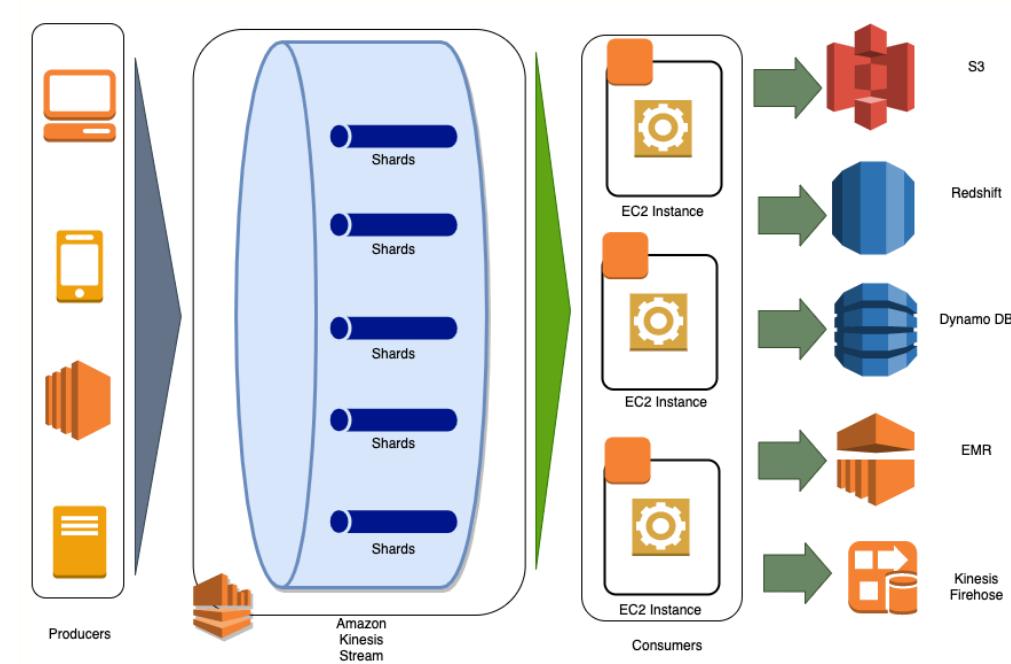
# Amazon Kinesis Data Streams - Integrations

- Use application integrations to generate streams
  - Toolkits : AWS SDK, AWS Mobile SDK, Kinesis Agent
  - Service Integrations : AWS IOT, CloudWatch Events and Logs
- Process streams using Kinesis Stream Applications
  - Run on EC2 instances
  - Written using Kinesis Data Streams APIs



# Kinesis Streams - Hierarchy

- **Hierarchy** : Data stream > Shards > Data Records
- Kinesis Data Streams uses a **Partition Key** to distribute the stream among the shards
- Ordering of records in a shard is guaranteed
  - Data Records in each Shard are ordered with a unique sequence number
- Each Shard supports 1,000 records per second
  - Increasing the number of shards increases the data capacity of the stream



# Kinesis Streams - Resharding

- **Resharding:** Adapt number of shards according to rate of data flow
- Only two low level operations are supported:
  - **Shard split:** Divide a shard into two
  - **Shard merge:** Merge two shards into one
- A high level operation update-shard-count is provided which internally can increase/decrease shards by splitting/merging.
- **(BEST PRACTICE):** Maintain the same number of consumer instances as the number of shards.
  - Maximum no of consumer instances = number of shards

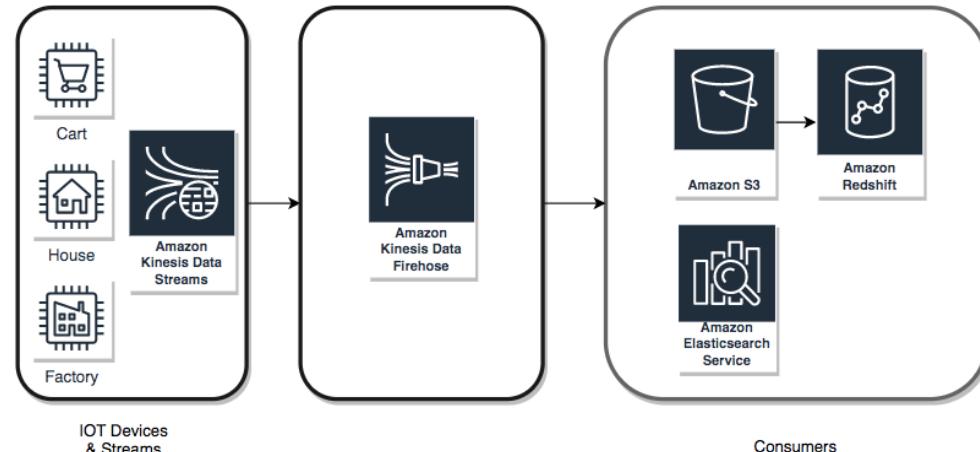
# Kinesis Streams - API

In 28  
Minutes

API	Description
<code>put-record</code>	Send one record at a time. Returns shard ID and sequence number
<code>put-records</code>	Write multiple records. Up to 500 records. (RECOMMENDED) Higher throughput per data producer
<code>create-stream, delete-stream, describe-stream, list-streams</code>	Stream Operations
<code>list-stream-consumers</code>	Lists the consumers registered to receive data from a stream
<code>register-stream-consumer</code>	Register a new consumer for the stream
<code>update-shard-count, list-shards, merge-shards, split-shard</code>	Merge and Split shards. When you use Update Shard Count, Kinesis decides how to Merge and Split.

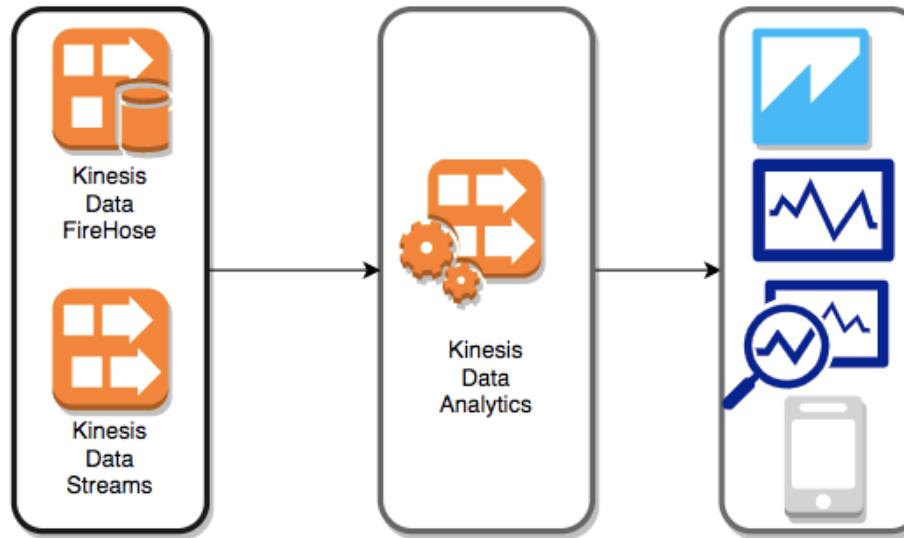
# Amazon Kinesis Data Firehose

- Data ingestion for streaming data
  - Receive
  - Process ( transform - Lambda, compress, encrypt )
  - Store stream data to S3, Elasticsearch, Redshift and Splunk
- Use existing analytics tools based on S3, Redshift and Elasticsearch
- Pay for volume of data ingested (Serverless)



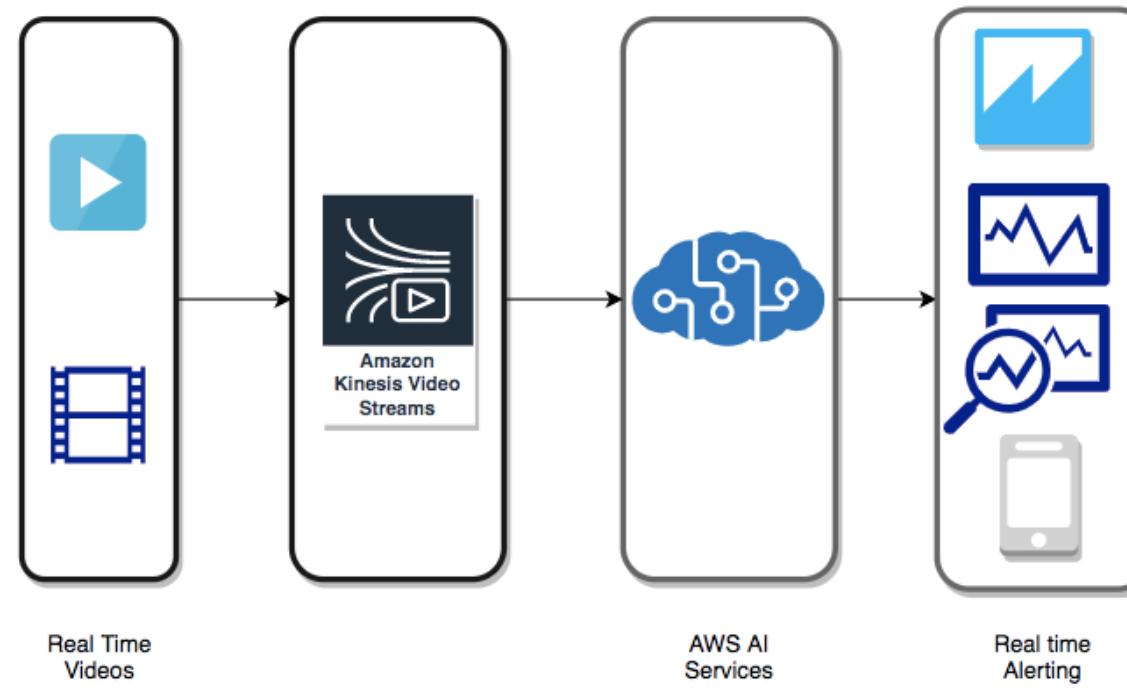
# Amazon Kinesis Analytics

In 28  
Minutes



- You want to continuously find active number of users on a website in the last 5 minutes based on streaming website data
- With Amazon Kinesis Analytics, you can write SQL queries and build Java applications to continuously analyze your streaming data

# Amazon Kinesis Video Streams



- Monitor video streams from web-cams
- Examples: traffic lights, shopping malls, homes etc
- Integrate with machine learning frameworks to get intelligence

# Kinesis Streams - Scenario Questions

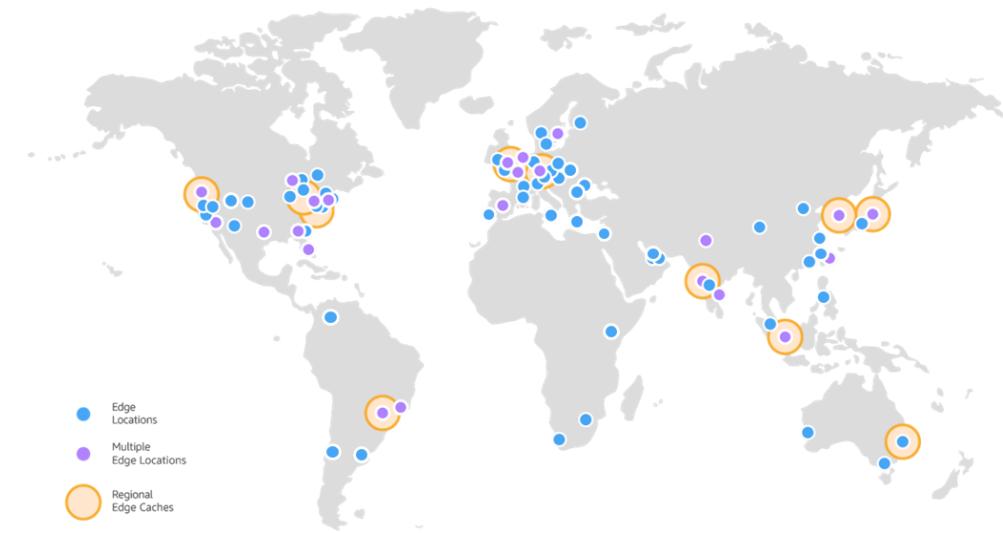
In 28  
Minutes

Scenario	Solution
<b>What are the recommended use cases for Kinesis Streams?</b>	<ol style="list-style-type: none"><li>1. A continuous stream of analytics from a web application</li><li>2. Multiple consumers consuming from a stream of data</li><li>3. Consume records in the same order a few hours later</li></ol>
<b>Develop Producer Applications for Kinesis Streams</b>	Use Amazon Kinesis Producer Library (KPL) or Amazon Kinesis Agent (pre-built Java application)
<b>Build Consumer Applications for Kinesis Streams</b>	Use Amazon Kinesis Client Library (KCL). Supports Java, Python, Ruby, Node.js and .NET.
<b>ProvisionedThroughputExceededException happens infrequently</b>	Retry with exponential backoff
<b>Increase throughput of an Amazon Kinesis Data Stream</b>	Resharding - Increase the number of shards.

# Routing and Content Delivery

# Content Delivery Network

- You want to deliver content to your global audience
- Content Delivery Networks distribute content to multiple edge locations around the world
- AWS provides 200+ edge locations around the world
- Provides high availability and performance



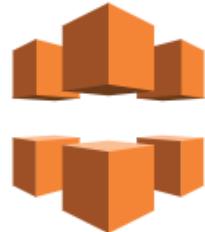
# Amazon CloudFront

- How do you enable serving content directly from AWS edge locations?
  - Amazon CloudFront (one of the options)
- Serve users from nearest edge location (based on user location)
- Source content can be from S3, EC2, ELB and External Websites
- If content is not available at the edge location, it is retrieved from the origin server and cached
- No minimum usage commitment
- Provides features to protect your private content



# Amazon CloudFront

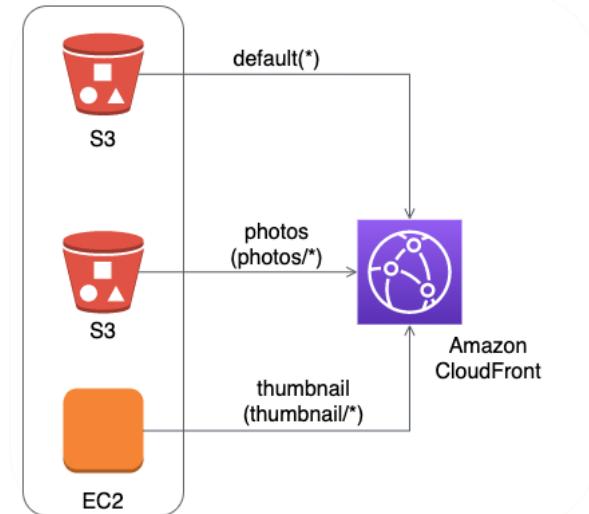
- Use Cases
  - Static web apps. Audio, video and software downloads. Dynamic web apps
  - Support media streaming with HTTP and RTMP
- Integrates with
  - AWS Shield to protect from DDoS attacks
  - AWS Web Application Firewall (WAF) to protect from SQL injection, cross-site scripting, etc
- Cost Benefits
  - Zero cost for data transfer between S3 and CloudFront
  - Reduce compute workload for your EC2 instances



CloudFront

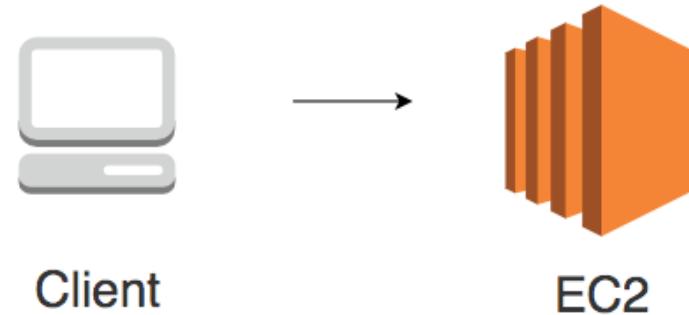
# Amazon CloudFront Distribution

- Create a CloudFront Distribution to distribute your content to edge locations
  - DNS domain name - example abc.cloudfront.com
  - Origins - Where do you get content from? S3, EC2, ELB, External Website
  - Cache-Control
    - By default objects expire after 24 hours
    - Customize min, max, default TTL in CloudFront distribution
    - (For file level customization) Use Cache-Control max-age and Expires headers in origin server
- You can configure CloudFront to only use HTTPS (or) use HTTPS for certain objects
  - Default is to support both HTTP and HTTPS
  - You can configure CloudFront to redirect HTTP to HTTPS



# Simple Architecture - Static Content

In 28  
Minutes



- (NOT RECOMMENDED) Serving static content from EC2 puts unnecessary load on EC2 instances

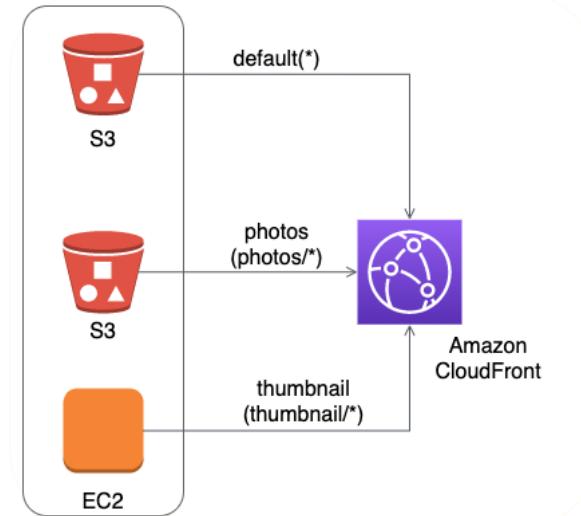
# Recommended Architecture - Static Content



- Store static content in S3
- Distribute it to edge locations around the world using CloudFront
- Advantages:
  - Pay for use
  - Low latency
  - Simple Caching (with TTL)
  - Reduce load on your compute instances (for example, EC2)

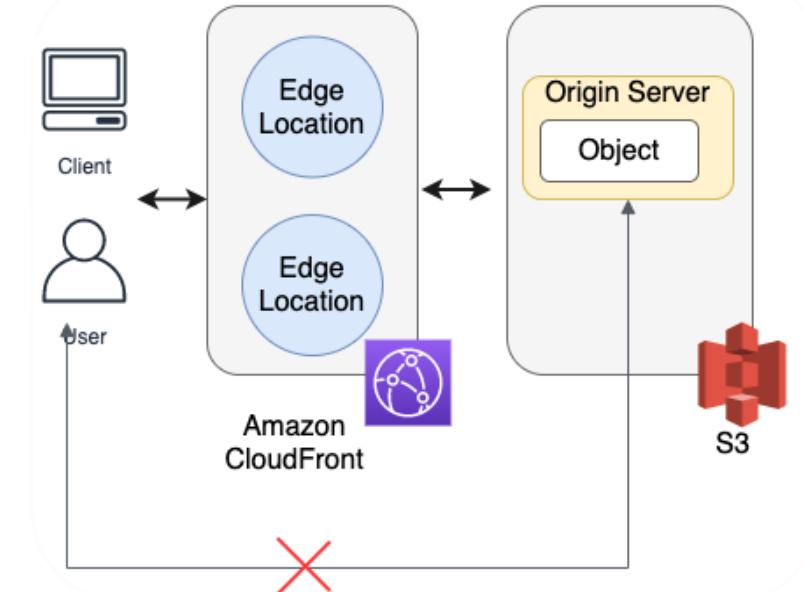
# Amazon CloudFront - Cache Behaviors

- Configure different CloudFront behavior for different URL path patterns from same origin
  - Path pattern(can use wild cards - \*.php, \*.jsp),
  - Do you want to forward query strings?
  - Should we use https?
  - TTL



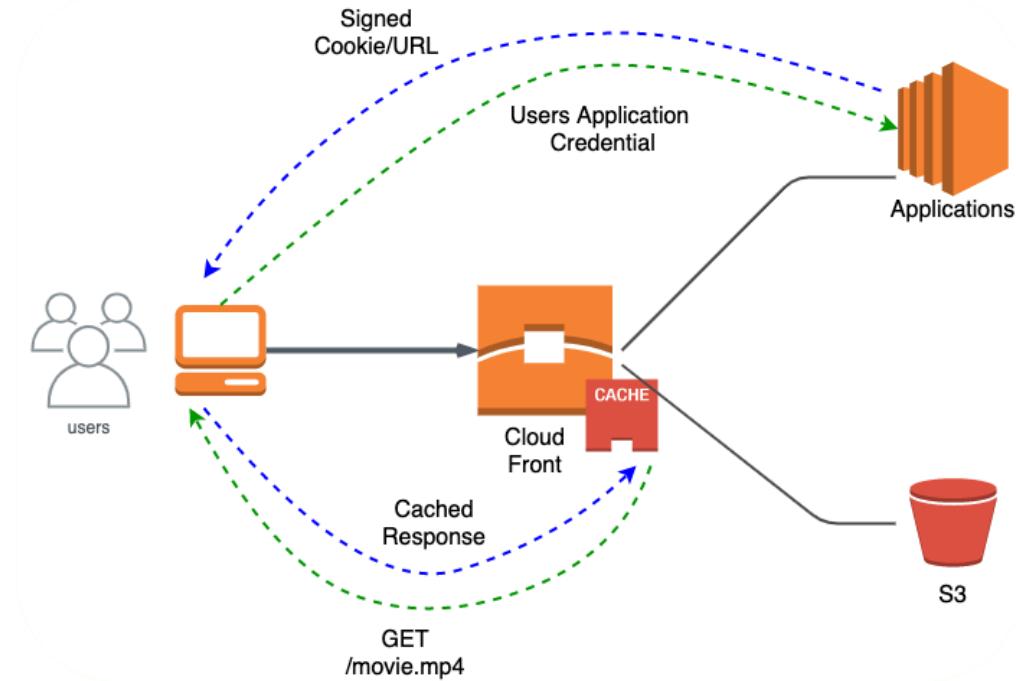
# Amazon CloudFront - Private content

- Signed URLs
- Signed cookies using key pairs
- Origin Access Identities(OAI)
  - Ensures that only CloudFront can access S3
  - Allow access to S3 only to a special CloudFront user

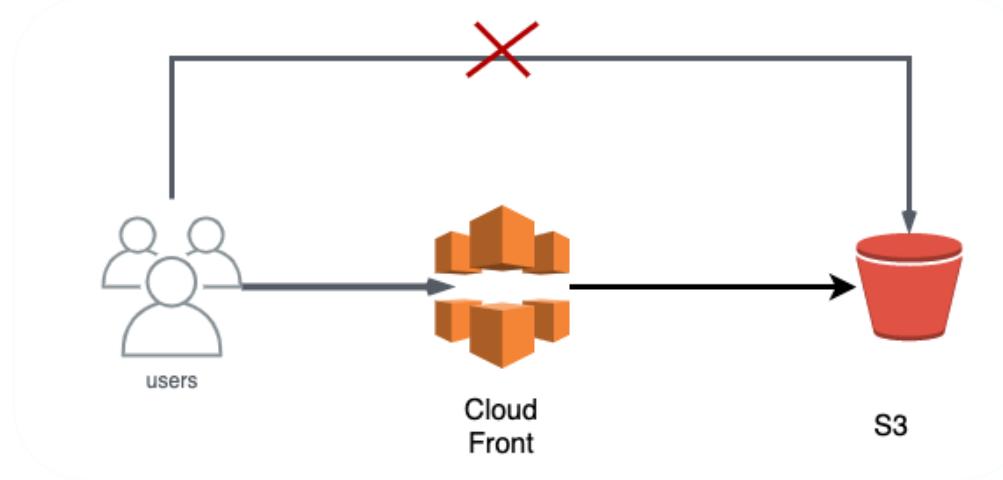


# Amazon CloudFront - Signed URLs and Cookies

- Signed URLs
  - RTMP distribution
  - Application downloads (individual files) and
  - Situations where cookies are not supported
- Signed Cookies
  - Multiple files (You have a subscriber website)
  - Does not need any change in application URLs



# Amazon CloudFront - Origin Access Identities(OAI)



- Only CloudFront can access S3
- Create a Special CloudFront user - Origin Access Identities(OAI)
- Associate OAI with CloudFront distribution
- Create a S3 Bucket Policy allowing access to OAI

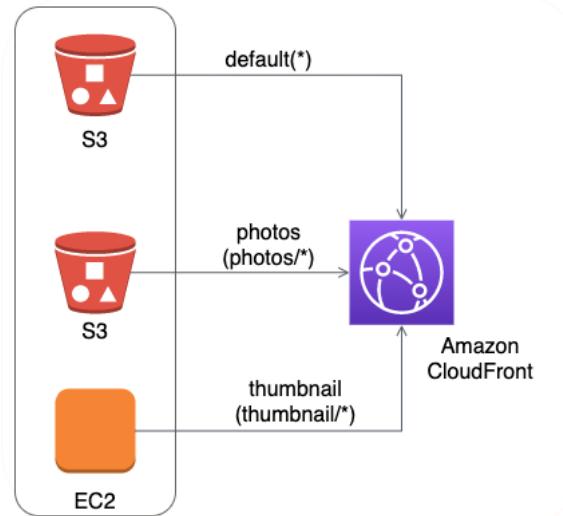
# Bucket Policy - S3 ONLY through Cloud Front



```
{  
    "Version": "2012-10-17",  
    "Id": "PolicyForCloudFrontPrivateContent",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS":  
                    "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity YOUR_IDENTITY_NAME"},  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::mybucket/*"  
        }  
    ]  
}
```

# Amazon CloudFront - Remember

- Old content automatically expires from CloudFront
- Invalidation API - remove object from cache
  - REMEMBER : Designed for use in emergencies
- Best Practice - Use versioning in object path name
  - Example : /images/profile.png?version=1
  - Prevents the need to invalidated content
- Do not use CloudFront for
  - all requests from single location
  - all requests from corporate VPN
- Scenario: Restrict content to users in certain countries
  - Enable CloudFront Geo restriction
  - Configure White list(countries to be allowed) and Blacklist(countries to be blocked)



# Route 53

- What would be the steps in setting up a website with a domain name (for example, in28minutes.com)?
  - Step I : Buy the domain name in28minutes.com (Domain Registrar)
  - Step II : Setup your website content (Website Hosting)
  - Step III : Route requests to in28minutes.com to the my website host server (DNS)
- Route 53 = Domain Registrar + DNS
  - Buy your domain name
  - Setup your DNS routing for in28minutes.com



Route53

# Route 53 - DNS (Domain Name Server)

*How should traffic be routed for in28minutes.com?*



Route53

- Configure Records:
  - Route api.in28minutes.com to the IP address of api server
  - Route static.in28minutes.com to the IP address of http server
  - Route email (ranga@in28minutes.com) to the mail server(mail.in28minutes.com)
  - Each record is associated with a TTL (Time To Live) - How long is your mapping cached at the routers and the client?

# Route 53 Hosted Zone

- Container for records containing DNS records routing traffic for a specific domain
- I want to use Route 53 to manage the records (Name Server) for [in28minutes.com](http://in28minutes.com)
  - Create a hosted zone for [in28minutes.com](http://in28minutes.com) in Route 53
- Hosted zones can be
  - private - routing within VPCs
  - public - routing on internet
- Manage the DNS records in a Hosted Zone



Route53

# Standard DNS Records

- A - Name to IPV4 address(es)
- AAAA - Name to IPV6 address(es )
- NS - Name Server containing DNS records
  - I bought in28minutes.com from GoDaddy (Domain Registrar)
  - BUT I can use Route 53 as DNS
    - Create NS records on GoDaddy
    - Redirect to Route 53 Name Servers
- MX - Mail Exchange
- CNAME - Name1 to Name2

	Name	Type	Value
	api.in28minutes.com.	A	192.0.2.235
	static.in28minutes.com.	AAAA	2001:0db8:85a3:0:0:8a2e:0370:7334
	dummy.in28minutes.com.	CNAME	www.example.com
	in28minutes.com.	MX	10 mailserver.in28minutes.com
			ns-1423.awsdns-49.org. ns-146.awsdns-18.com. ns-981.awsdns-58.net. ns-1997.awsdns-57.co.uk.
	in28minutes.com.	NS	
	in28minutes.com.	SOA	ns-1423.awsdns-49.org. awsdns-hostmaster.amazo

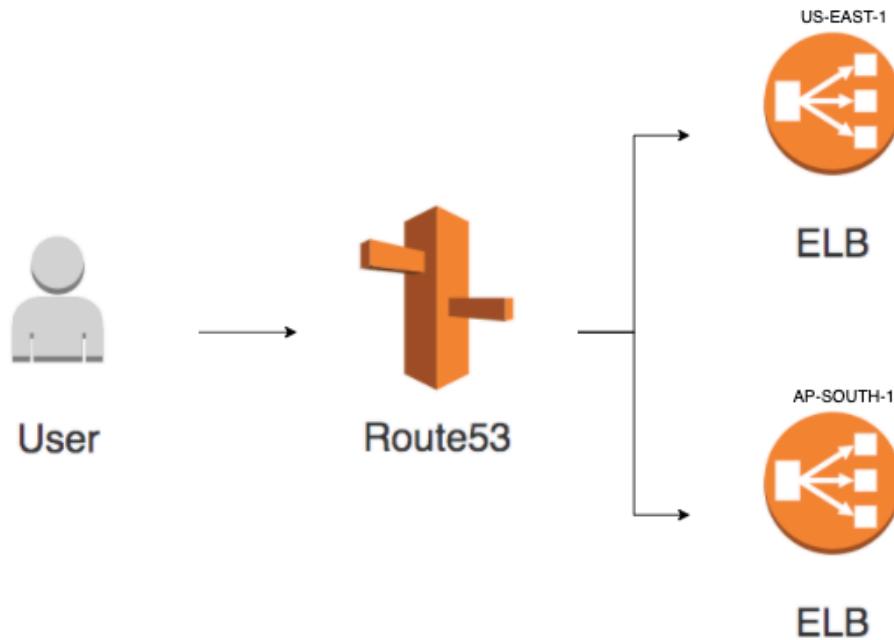
# Route 53 Specific Extension - Alias records

- Route traffic to selected AWS resources
  - Elastic Beanstalk environment
  - ELB load balancer
  - Amazon S3 bucket
  - CloudFront distribution
- Alias records can be created for
  - root(in28minutes.com) and
  - non root domains(api.in28minutes.com)
- COMPARED to CNAME records which can only be created for
  - non root domains (api.in28minutes.com)



Route53

# Route 53 - Routing



- Route 53 can route across Regions
  - Create ALBs in multiple regions and route to them!
  - Offers multiple routing policies

# Route 53 Routing Policies

Policy	Description
<b>Simple</b>	Maps a domain name to (one or more) IP Addresses
<b>Weighted</b>	Maps a single DNS name to multiple weighted resources 10% to A, 30% to B, 60% to C (useful for canary deployments)
<b>Latency</b>	Choose the option with minimum latency Latency between hosts on the internet can change over time
<b>Failover</b>	Active-passive failover. Primary Health check fails (optional cloud Watch alarm) => DR site is used
<b>Geoproximity</b>	Choose the nearest resource (geographic distance) to your user. Configure a bias.
<b>Multivalue answer</b>	Return multiple healthy records (upto 8) at random You can configure an (optional) health check against every record
<b>Geolocation</b>	Choose based on the location of the user

# Route 53 Routing Policies - Geolocation

- Choose based on the location of the user
  - continent, country or a (state in USA)
  - Send traffic from Asia to A
  - Send traffic from Europe to B etc.
- Record set for smallest geographic region has priority
- Use case
  - Restrict distribution of content to specific areas where you have distribution rights
- (RECOMMENDED) Configure a default policy (used if none of the location records match)
  - Otherwise, Route 53 returns a "no answer" if none of the location records match



Route53

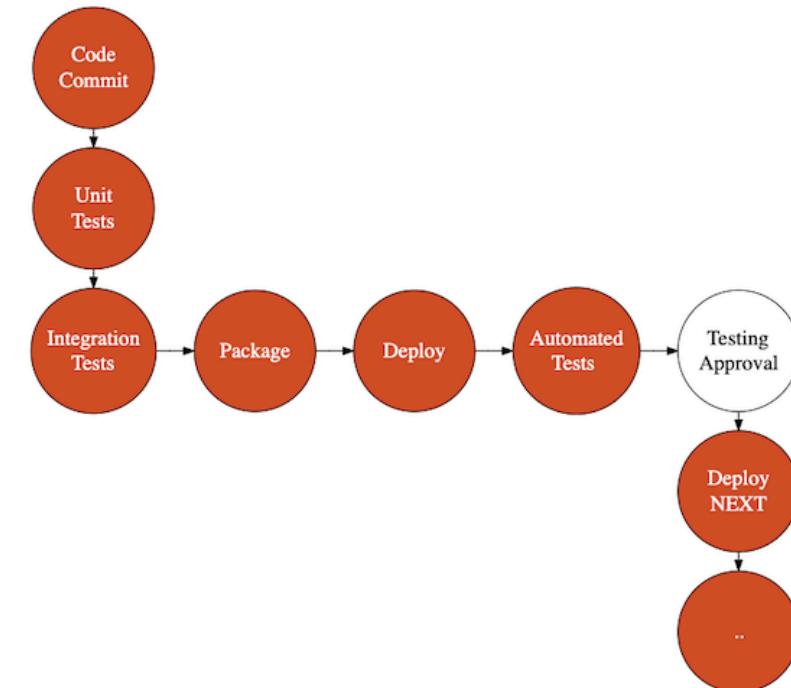
# DevOps



- Getting Better at "**Three Elements of Great Software Teams**"
  - **Communication** - Get teams together
  - **Feedback** - Earlier you find a problem, easier it is to fix
  - **Automation** - Automate testing, infrastructure provisioning, deployment, and monitoring

# DevOps - CI, CD

- **Continuous Integration**
  - Continuously run your tests and packaging
- **Continuous Deployment**
  - Continuously deploy to test environments
- **Continuous Delivery**
  - Continuously deploy to production



# DevOps - CI, CD Tools

In 28  
Minutes



Codecommit



Codepipeline



Codebuild



Codedeploy

- **AWS CodeCommit** - Private source control (Git)
- **AWS CodePipeline** - Orchestrate CI/CD pipelines
- **AWS CodeBuild** - Build and Test Code (application packages and containers)
- **AWS CodeDeploy** - Automate Deployment (EC2, ECS, Lambda etc)

# DevOps - IAC (Infrastructure as Code)



- Treat infrastructure the same way as application code
- Track your infrastructure changes over time (version control)
- Bring repeatability into your infrastructure
- Two Key Parts
  - **Infrastructure Provisioning**
    - Provisioning compute, database, storage and networking
    - Open source cloud neutral - Terraform
  - **Configuration Management**
    - Install right software and tools on the provisioned resources
    - Open Source Tools - Chef, Puppet, Ansible



- **Infrastructure Provisioning**
  - Open Source
    - Terraform
  - AWS CloudFormation
    - Provision AWS Resources
  - AWS SAM (Serverless Application Model)
    - Provision Serverless Resources
- **Configuration Management**
  - Open Source Tools
    - Chef, Puppet, Ansible
  - AWS Service
    - OpsWorks

# AWS CodeCommit

- Version control service hosted by AWS
  - Teams can work collaboratively on the code-base
- Features:
  - Based on Git => ZERO learning curve
  - Integrates with other AWS or Third-Party services
  - CodeCommit repositories are automatically encrypted at rest and in transit
    - KMS key is managed by AWS
  - Authentication
    - IAM User (RECOMMENDED)
      - Users can log in using SSH or HTTPS Git access credentials
    - IAM Role
      - Set up federated access, cross-account access and other AWS service access
  - Monitoring
    - Events such as pull request or repository deletion can be captured in CloudWatch events
      - They can then trigger targets such as Lambda or SNS notifications



Codecommit

# AWS CodeBuild

- Deployable artifacts are generated from code
  - Example: jar or war for Java applications
- **CodeBuild** - Fully managed build service in AWS
  - Provides pre-configured build environments (Docker images) for popular programming languages
  - Compile source code, run unit tests, and create artifacts
  - Integrates with Amazon CloudWatch for logs
- How does it work?
  - Step 1: Create Build Spec file with instructions on:
    - Commands to build your project
    - Define your output files (output files are uploaded to S3)
  - Step 2: Create a Code Build project defining:
    - Where is your source code?
      - CodeCommit/Amazon S3/GitHub/Bitbucket
    - Build environment (Operating System, Programming Language or Runtime, Tools)



Codebuild

# AWS CodeBuild - Buildspec

- Buildspec is a collection of build commands and related settings
- Default name - `buildspec.yml` (at root of source directory)
- Major Elements:
  - Env - Information about environment variables
  - Phases - Build is divided into pre-defined phases. Each phase can have its own set of commands.
    - Install
    - Pre-Build
    - Build
    - Post-Build
  - Artifacts - Build output files (uploaded to S3 after build)



Codebuild

# AWS CodeBuild - Build Spec - Docker Example

In 28  
Minutes

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: openjdk8
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - TAG="$(echo ${CODEBUILD_RESOLVED_SOURCE_VERSION} | head -c 8)"
      - IMAGE_URI=${REPOSITORY_URI}:${TAG}
  build:
    commands:
      - mvn clean package -Ddockerfile.skip
      - docker build --tag $IMAGE_URI .
  post_build:
    commands:
      - docker push $IMAGE_URI
      - echo push completed
      - printf '[{"name":"%s","imageUri":"%s"}]' $CONTAINER_NAME $IMAGE_URI > imagedefinition.json
artifacts:
  files:
    - imagedefinitions.json
```

# AWS CodeBuild - VPC support

- Typically CodeBuild can't access resources in the VPC
- What if you need to run integration tests by connecting to the test database inside a VPC?
  - Additional configurations have to be provided for VPC integrations
    - VPC ID
    - Subnet IDs
    - Security group IDs
  - You can also use VPC Endpoint to give access to VPC
  - Typical use-cases :
    - Connecting to web services hosted in EC2
    - Query data from the database



Codebuild

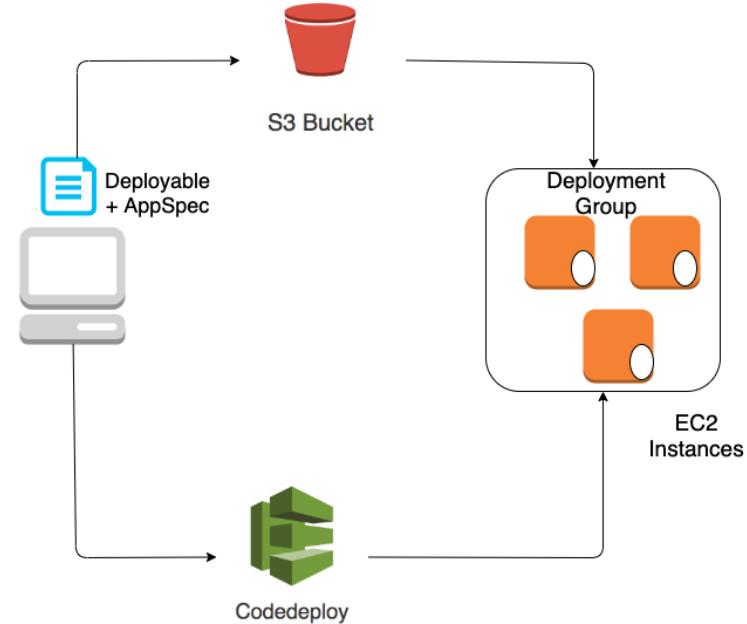
# AWS CodeDeploy

- Automate application deployments to:
  - Amazon EC2 instances
  - On-premises instances
  - Serverless Lambda functions
  - Amazon ECS services
- Important Features:
  - Autoscaling
  - Integrates with CodePipeline
  - Automated rollbacks
  - Reuse existing setup:
    - Integrates with:
      - CI/CD tools (Jenkins, GitHub etc)
      - IAAC tools (Ansible, Puppet, Chef etc)



# AWS CodeDeploy - Deployment Types

- Deployment - Deploying new version to target
  - Two types: In-place and Blue-Green
- **In-place deployment**
  - Step 1 : Application is stopped
  - Step 2 : Latest version is installed
  - Step 3 : Application started and validated
  - (CONSTRAINT) Only EC2/On-Premises can use this
- **Blue/green deployment**
  - A new environment is created and traffic shifted to a new environment
  - Supports:
    - EC2 (Does NOT work with on-premises)
    - Lambda - Traffic shifted to a new serverless environment
    - ECS - Traffic shifted from the current task set to replacement



# AWS CodeDeploy - Important Configuration

- Questions to ask?
  - What to deploy?
  - Where to deploy?
  - What type of deployment?
    - in-place, blue/green
  - At what rate to shift traffic?
    - canary, linear, all-at-once
- Allows you to:
  - Release new features avoiding manual deployment errors
  - Avoid/minimize downtime due to releases
    - Multiple deployment options - in-place, blue/green deployments
    - Concurrent deployments



Codedeploy

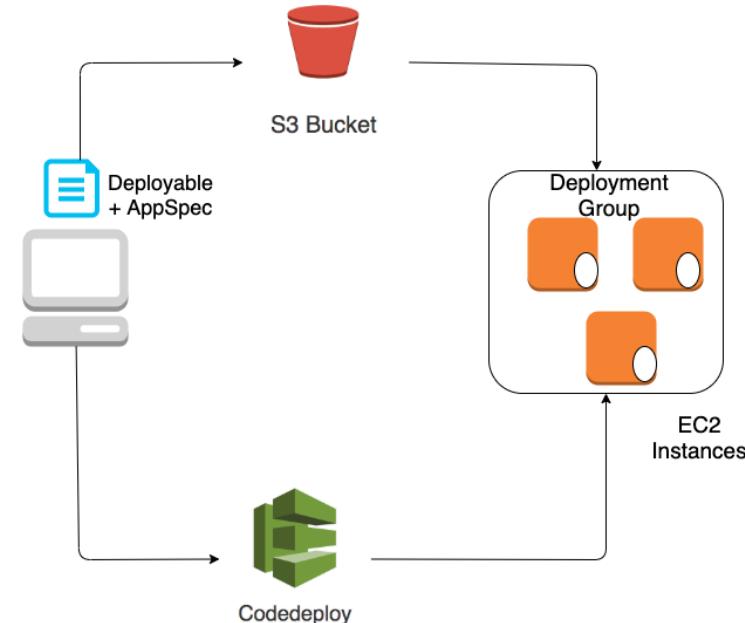
# AWS CodeDeploy - Components

- **Application** - Unique name for application to be deployed
- **Compute platform** - Where to deploy?
  - EC2/On-premises / AWS Lambda / Amazon ECS
- **Deployment configuration** - Deployment rules and success/failure conditions:
  - **Deployment group** - Which instances should be used?
  - **Target revision** - Which version should be deployed?
  - **Deployment type** - What type of deployment?
    - In-place deployment, Blue/Green deployment
  - **Traffic Shifting** - How is traffic is routed to new deployment?
    - Canary , Linear , All-at-once



# AWS CodeDeploy - EC2/On-Premises

- Deploy executable files, configuration files, images, and more to Amazon EC2 cloud instances, on-premises servers, or both:
  - CodeDeploy agent - Installed on an instance to be used in CodeDeploy deployments
  - Code can be picked up from S3 or Github
  - Identify resources using tags or use instances in Auto Scaling Groups or both
- Supports in-place or blue/green deployment:
  - Blue/Green deployment is NOT supported for on-premise instances
  - (Blue Green Deployment) ELB is used to route traffic from original environment to replacement environment



# AWS CodeDeploy - Sample AppSpec For EC2/On-Premise

```
version: 0.0
os: linux
# os: windows
files:
  - source: #where to copy from (in source bundle)
    destination: #where to copy to
  - source:
    destination:
permissions: # permissions to apply to files in "files" section
  - object:
    pattern:
hooks:
  ApplicationStop: #scripts/stop_server.sh
    - location: #script location
      timeout:
      runas:
  BeforeInstall: #scripts/install_dependencies.sh
  AfterInstall: #scripts/change_permissions.sh
  ApplicationStart: #scripts/start_server.sh
  ValidateService: #scripts/run_tests.sh
```

# AWS CodeDeploy - AWS Lambda

```
version: 0.0
Resources:
  - myLambdaFunction:
      Type: AWS::Lambda::Function
      Properties:
        Name: "myLambdaFunction"
        Alias: "myLambdaFunctionAlias"
        CurrentVersion: "1"
        TargetVersion: "2"
    Hooks:
      - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
      - AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"
```

- Deploy updated version of a Lambda function
- Supports only Blue/Green deployments
  - Traffic Shifting - canary, linear, or all-at-once

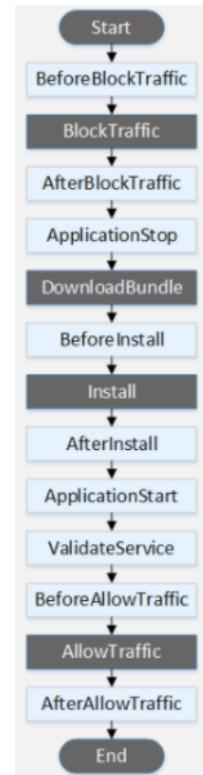
# AWS CodeDeploy - Amazon ECS

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
    Properties:
      TaskDefinition: "" # ARN of your task definition
      LoadBalancerInfo:
        ContainerName: "" # ECS application's container name
        ContainerPort: ""
        #(OPTIONAL)NetworkConfiguration
        #
        #           > AwsVpcConfiguration
        #
        #           > Subnets,SecurityGroups,AssignPublicIp
  Hooks:
    - BeforeInstall: "" # Lambda function name or ARN for each hook
    - AfterInstall: ""
    - AfterAllowTestTraffic: ""
    - BeforeAllowTraffic: ""
    - AfterAllowTraffic: ""
```

- Updated version of application installed as a new replacement task set
- Supports only Blue/Green deployments
  - Traffic Shifting - canary, linear, or all-at-once

# CodeDeploy - Order of hooks

- Picture shows order of hooks for in-place deployment
- Start, DownloadBundle, Install, AllowTraffic and End events in the deployment cannot be scripted
  - You can write scripts for all other events for EC2/on-premise deployment!
- For Blue/green deployments:
  - BlockTraffic events executed before End (Between AfterAllowTraffic and End)
- Lambda Deployment supports following hooks:
  - BeforeAllowTraffic, AfterAllowTraffic
- ECS Deployment supports following hooks:
  - BeforeInstall, AfterInstall, AfterAllowTestTraffic, BeforeAllowTraffic, and AfterAllowTraffic



# CodeDeploy - Rollbacks

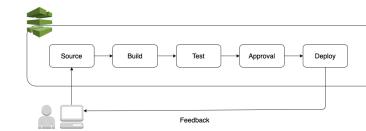
In 28  
Minutes



- How do Rollbacks work?
  - "Redeploy a previously deployed revision of an application as a new deployment" (technically **new deployments** using old versions)
  - Two types of Rollbacks: Automatic and Manual
- **Automatic rollbacks:** Automatically triggered on failed deployments or when CloudWatch alarm thresholds are met
- **Manual rollbacks:** Create a new deployment with previously deployed revision

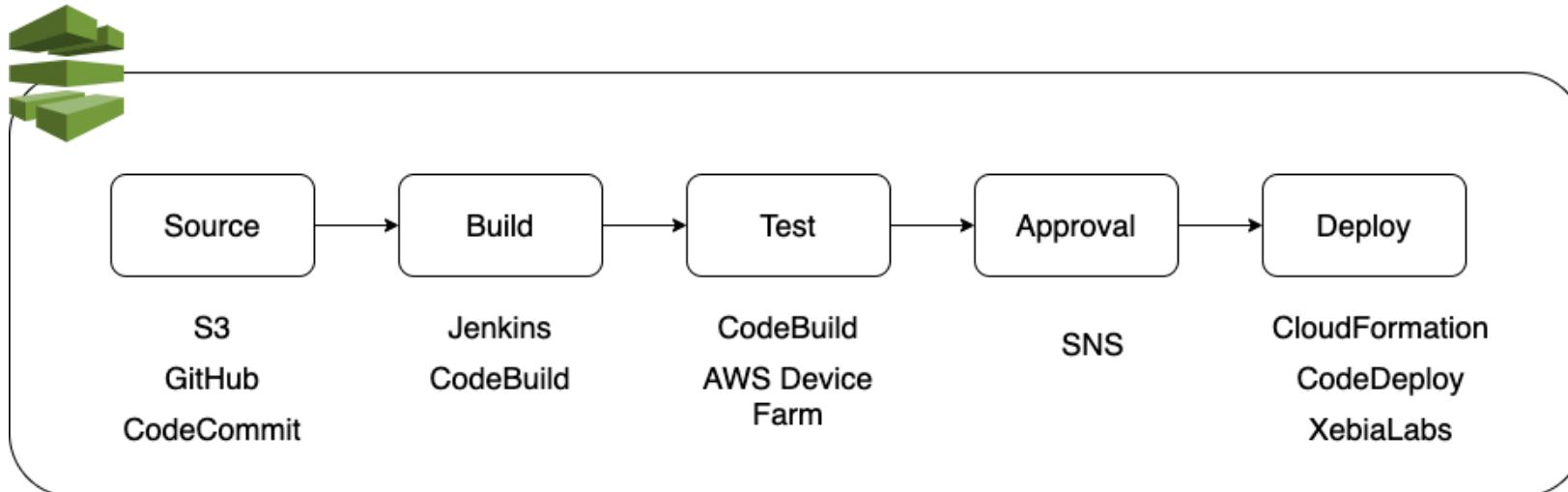
# AWS CodePipeline

- Testing, Building, Releasing and Deploying software involve multiples steps
- How do we automate these steps?
- Enter **AWS CodePipeline**
  - Each pipeline consists of multiple stages depending on the release workflow. Build, Test, Deploy are some examples
  - Model, visualize and automate steps involved
  - Integrates with various AWS & Third-party tools to automate end to end release process



# AWS CodePipeline - Integrations

In 28  
Minutes



- Integrates with various tools to automate the complete flow (shown in screen above).
- Integrates with AWS CloudTrail, CloudWatch, and CloudWatch Events.

# CodeStar - Develop and Deploy to AWS in minutes

- Set up your **entire development and continuous delivery tool-chain**:
  - Automatically set up coding, building, testing, and deploying your application code
- Features:
  - Project templates for Java, JavaScript, PHP, Ruby, C#, & Python:
    - Get started with Amazon EC2, AWS Lambda, and AWS Elastic Beanstalk
  - End to End Project Management:
    - Dashboard with integrated issue tracking(Atlassian JIRA Software), Project wiki etc.
  - Simplified IDE Configuration:
    - Cloud9, Microsoft Visual Studio and Eclipse
  - Source Version Control:
    - AWS CodeCommit
  - Automated CI/CD Pipeline:
    - AWS CodePipeline, AWS CodeDeploy and AWS CloudFormation
- (REMEMBER) Need to setup version control, build and deployment with a project dashboard quickly => **Use CodeStar**

# DevOps - Scenario Questions

Scenario	Solution
Which stage of CodePipeline is used to build a deployable artifact?	CodeBuild
Which stage of CodePipeline is used to run your unit tests?	CodeBuild
Can you run CodeBuild in your local machines?	Yes. Install AWS CodeBuild agent.
Can you move your code from a Github Repo to CodeCommit?	<p>Yes.</p> <ol style="list-style-type: none"><li>1. Clone GitHub Repo. Create CodeCommit Repo.</li><li>2. Create an IAM user for CodeCommit with policy AWSCodeCommitPowerUser.</li><li>3. Configure Git credentials for CodeCommit (HTTPS, recommended for most users) or an SSH key pair to use when accessing CodeCommit (SSH)</li><li>4. Push to CodeCommit Repo.</li></ol>

# DevOps - Scenario Questions - 2

In 28  
Minutes

## Scenario

**What events can be used to trigger CodePipelines?**

## Solution

Other than usual Git repos (Bitbucket, GitHub, GitHub Enterprise), you can trigger builds based on changes to Amazon S3 buckets and Amazon ECR repositories

**Can you add manual approvals in CodePipeline?**

Yes. Add an approval action to the corresponding stage in a CodePipeline pipeline.

**Where should configuration files for CodeDeploy (appspec.yml or appspec.json) and CodeBuild(buildspec.yml) be located (By Default)?**

Default is at the root of your CodeCommit directory.  
1. Remember that CodeDeploy supports YAML and JSON. CodeBuild supports only YAML.  
2. Remember the extension of YAML files - It is `yml`.

# AWS CloudFormation

# AWS Cloud Formation - Introduction

- Lets consider an example:
  - I would want to create a new VPC and a subnet
  - I want to provision a ELB, ASG with 5 EC2 instances & RDS database
  - I would want to setup the right security groups
- AND I would want to create 4 environments
  - Dev, QA, Stage and Production!
- CloudFormation can help you do all these with a simple (actually NOT so simple) script!
- Advantages (Infrastructure as Code - IAC & CloudFormation) :
  - Automate deployment of AWS resources in a controlled, predictable way
  - Avoid mistakes with manual configuration
  - Think of it as version control for your environments



CloudFormation

# AWS Cloud Formation

- All configuration is defined in a simple text file - JSON or YAML
  - I want a VPC, a subnet, a database and ...
- CloudFormation understands dependencies
  - Creates VPCs first, then subnets and then the database
- (Default) Automatic rollbacks on errors (Easier to retry)
  - If creation of database fails, it would automatically delete the subnet and VPC
- Version control your configuration file and make changes to it over time
- Free to use - Pay only for the resources provisioned
  - Get an automated estimate for your configuration



CloudFormation

# AWS Cloud Formation - Example 1 - JSON

```
{  
    "Resources" : {  
        "MyBucket" : {  
            "Type" : "AWS::S3::Bucket"  
            "Properties" : {  
                "AccessControl" : "PublicRead"  
            }  
        }  
    }  
}
```

# AWS CloudFormation - Example 2 - YAML

```
Resources:  
  MyBucket:  
    Type: AWS::S3::Bucket  
    Properties:  
      AccessControl: PublicRead
```

# AWS Cloud Formation - Terminology

- Template
  - A Cloud Formation JSON or YAML defining multiple resources
- Stack
  - A group of resources that are created from a CloudFormation template
- Change Sets
  - To make changes to stack, update the template
  - Change set shows what would change if you execute
  - Allows you to verify the changes and then execute

# AWS Cloud Formation - Important template elements

In 28  
Minutes

```
{  
    "AWSTemplateFormatVersion" : "version date",  
    "Description" : "JSON string",  
    "Metadata" : {},  
    "Parameters" : {},  
    "Mappings" : {},  
    "Resources" : {},  
    "Outputs" : {}  
}
```

- Resources - What do you want to create?
  - One and only mandatory element
- Parameters - Values to pass to your template at runtime
  - Which EC2 instance to create? - ("t2.micro", "m1.small", "m1.large")
- Mappings - Key value pairs
  - Example: Configure different values for different regions
- Outputs - Return values from execution
  - See them on console and use in automation

# AWS Cloud Formation - Resources

```
Resources:  
  HelloBucket:  
    Type: AWS::S3::Bucket  
    Properties:  
      AccessControl: PublicRead  
      WebsiteConfiguration:  
        IndexDocument: index.html  
        ErrorDocument: error.html
```

- The only mandatory section in the template
- Contains the list of resource objects to be created
- Each resource has different attributes (mandatory & optional)
  - Imageld attribute for an EC2 Instance resource
  - Specified under **Properties**
- **Type** attribute specifies the type of the resource.
- Format for type attribute is (**AWS::ProductIdentifier::ResourceType**)

# AWS Cloud Formation - Parameters

```
Parameters:  
  InstanceTypeParameter:  
    Type: String  
    Default: t2.micro  
    AllowedValues:  
      - t2.micro  
      - m1.large  
    Description: Enter t2.micro or m1.large.  
Resources:  
  Instance:  
    Type: 'AWS::EC2::Instance'  
    Properties:  
      InstanceType: !Ref InstanceTypeParameter
```

- Parameters make the template dynamic
  - You can define constraints on parameters - AllowedPattern, AllowedValues, MaxLength, MaxValue, MinLength, MinValue etc
- Type is mandatory (String, Number etc)
  - Can be AWS-specific parameter type (Ex: AWS::EC2::Image::Id)

# AWS Cloud Formation - Pseudo parameters

Outputs:

MyStacksRegion:

Value: !Ref "AWS::Region"

- Parameters predefined by AWS CloudFormation
- Examples
  - AWS::AccountId
  - AWS::Region
  - AWS::StackId
  - AWS::StackName

# Common Resource Attributes - CreationPolicy

```
AutoScalingGroup:  
  Type: AWS::AutoScaling::AutoScalingGroup  
  Properties:  
    CreationPolicy:  
      ResourceSignal:  
        Count: '3'  
        Timeout: PT15M
```

*Attributes added to a resource to control its behavior*

- **CreationPolicy:** When is the creation of a resource complete?
  - AutoScalingCreationPolicy: How many instances in ASG should be ready?
  - ResourceSignal: No of signals and max wait time
  - Used with Amazon EC2 and Auto Scaling resources
  - (Alternative) For coordination with external configuration actions use WaitCondition

# Common Resource Attributes - Others

`UpdatePolicy:`

`AutoScalingReplacingUpdate:`

`WillReplace: true`

- `DeletionPolicy`: Preserve or backup resource when stack is deleted.
  - Possible Values:
    - Retain(Do not delete)
    - Snapshot(take a snapshot)
    - Delete (default)
- `DependsOn`: I would be created only after another resource is created
- `UpdatePolicy`: How should updates be handled (For example, to an ASG?)
  - `AutoScalingReplacingUpdate` (`WillReplace`)
  - `AutoScalingRollingUpdate` ( `MaxBatchSize`, `PauseTime` etc)

# AWS Cloud Formation - Conditions

- Condition can be attached with a resource or output section
- Based on the condition: Resource or output is created
- Example: Based on the input parameter value of Environment Type
  - Test environments use t2.micro
  - Prod environments use t2.large

```
AWSTemplateFormatVersion: "2010-09-09"
Mappings:
Parameters:
EnvType:
  Description: Environment type.
  Default: test
  Type: String
  AllowedValues:
    - prod
    - test
  ConstraintDescription: must specify prod or test.
Conditions:
CreateProdResources: !Equals [ !Ref EnvType, prod ]
```

```
Resources:
NewVolume:
  Type: "AWS::EC2::Volume"
  Condition: CreateProdResources
  Properties:
    Size: 100
    AvailabilityZone:
      !GetAtt EC2Instance.AvailabilityZone
```

```
Outputs:
VolumeId:
  Condition: CreateProdResources
  Value:
```

# AWS CloudFormation - Mappings

```
Mappings:  
RegionMap:  
  us-east-1:  
    "HVM64": "ami-0ff8a91507f77f867"  
    "HVMG2": "ami-0a584ac55a7631c0c"  
  us-west-1:  
    "HVM64": "ami-0bdb828fd58c52235"  
    "HVMG2": "ami-0a584ac55a7xyzabc"  
Resources:  
  myEC2Instance:  
    Type: "AWS::EC2::Instance"  
    Properties:  
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", HVM64]  
      InstanceType: m1.small
```

- Matches a key to the set of values(can contain one or multiple values)
- Example:
  - RegionMap defines the Mapping of regions and their AMIs
  - Usage of RegionMap: !FindInMap [RegionMap, !Ref "AWS::Region", HVM64]

# AWS Cloud Formation - Outputs

## Outputs:

**BackupLoadBalancerDNSName:**

**Description:** The DNSName of the backup load balancer

**Value:** !GetAtt BackupLoadBalancer.DNSName

**Condition:** CreateProdResources

**InstanceID:**

**Description:** The Instance ID

**Value:** !Ref EC2Instance

- Export values from templates for later use
  - Maximum of 60 outputs in a template
  - Can be used to create cross stack reference by exporting the value
  - CloudFormation does not hide or encrypt output section
    - If you export password it will be visible

# AWS CloudFormation - Transform

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31  
Resources:  
  CreateThumbnail:  
    Type: AWS::Serverless::Function
```

- Transform section specifies one or more macros that AWS CloudFormation uses to process your template
  - AWS::Include transform - Insert boilerplate template snippets into your templates
  - AWS::Serverless transform - Converts SAM templates to Cloud Formation
  - AWS::CodeDeployBlueGreen - Enable blue/green deployments through CodeDeploy

# AWS Cloud Formation - Intrinsic Functions - Ref

- **Intrinsic Functions** - Built-in functions provided by cloud formation
- Ref Function refers to other resources which are:
  - Defined in the template (OR)
  - Existing in AWS environment (OR)
  - Input Parameters defined in the templates
- Example: **EC2Instance** refers to the **SecurityGroup InstanceSecurityGroup**
  - Uses the logical name given to that resource. For example 'InstanceSecurityGroup'

```
Resources:  
Ec2Instance:  
  Type: 'AWS::EC2::Instance'  
  Properties:  
    SecurityGroups:  
      - !Ref InstanceSecurityGroup  
      - MyExistingSecurityGroup  
    KeyName: mykey  
    ImageId: ami-7a11e213  
  InstanceSecurityGroup:  
    Type: 'AWS::EC2::SecurityGroup'  
    Properties:  
      GroupDescription: Enable SSH access via port 22  
      SecurityGroupIngress:  
        - IpProtocol: tcp  
        FromPort: '22'  
        ToPort: '22'  
        CidrIp: 0.0.0.0/0
```

# AWS Cloud Formation - Intrinsic Functions - GetAtt

- Given logical name, Ref function returns the attribute that identifies the resource
- If we need any other specific attribute then **GetAtt** can be used
- Fn::GetAtt takes two parameters:
  - Logical name of the resource
  - Name of the attribute to be retrieved
- Example: Create a CloudFront distribution with S3 bucket as origin:
  - We need to get the S3 buckets domain name.
  - DomainName: !GetAtt**  
**'myBucket.DomainName'** (Short Form)

```
Resources:
myBucket:
  Type: 'AWS::S3::Bucket'
myDistribution:
  Type: 'AWS::CloudFront::Distribution'
  Properties:
    DistributionConfig:
      Origins:
        - DomainName: !GetAtt
          - myBucket
          - DomainName
        Id: myS3Origin
        S3OriginConfig: {}
      Enabled: 'true'
      DefaultCacheBehavior:
        TargetOriginId: myS3Origin
        ForwardedValues:
         QueryString: 'false'
      ViewerProtocolPolicy: allow-all
```

# AWS Cloud Formation - Functions - FindInMap

```
Mappings:  
RegionMap:  
  us-east-1:  
    "HVM64": "ami-0ff8a91507f77f867"  
    "HVMG2": "ami-0a584ac55a7631c0c"  
  us-west-1:  
    "HVM64": "ami-0bdb828fd58c52235"  
    "HVMG2": "ami-0a584ac55a7xyzabc"  
Resources:  
myEC2Instance:  
  Type: "AWS::EC2::Instance"  
  Properties:  
    ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", HVM64]  
    InstanceType: m1.small
```

- How to make template more generic and to use across different regions?
  - Create a Mapping and refer to value in it using `FindInMap`
- Parameters - name of the map, key and label
  - Example : `!FindInMap [RegionMap, !Ref "AWS::Region", HVM64]`

# AWS Cloud Formation - Functions - Join

- Join multiple values
- Two parameters:
  - delimiter
  - array of values to be joined
- Example:
  - "Fn::Join": [ "", ["http://", {"Fn::GetAtt": ["ElasticLoadBalancer", "DNSName"] } ] ]

```
Resources:
ElasticLoadBalancer:
Type: 'AWS::ElasticLoadBalancing::LoadBalancer'
Properties:
  AvailabilityZones: !GetAZs ''
  Instances:
  - !Ref Ec2Instance1
  - !Ref Ec2Instance2
  Listeners:
  - LoadBalancerPort: '80'
    InstancePort: !Ref WebServerPort
    Protocol: HTTP
  HealthCheck:
  Target: !Join
    - ''
    - - 'HTTP:'
    - !Ref WebServerPort
    - /
  HealthyThreshold: '3'
  UnhealthyThreshold: '5'
  Interval: '30'
  Timeout: '5'
```

# Other Intrinsic functions

- **Fn::Cidr** - Returns an array of CIDR address blocks
  - !Cidr [ "192.168.0.0/24", 6, 5 ]
- **Fn::GetAZs** - Returns an array that lists Availability Zones for a specified region in alphabetical order
  - Fn::GetAZs: us-east-1 or Fn::GetAZs: !Ref 'AWS::Region'
- **Fn::ImportValue** - Use output from another stack
- **Fn::Select** - Returns value at index
  - !Select [ "1", [ "apples", "grapes", "oranges", "mangoes" ] ] => "grapes"
- **Fn::Transform** - Specifies a macro to perform custom processing

# CloudFormation Execution Status Examples

Status	Description
CREATE_COMPLETE	Successful creation of one or more stacks
CREATE_IN_PROGRESS	Ongoing creation of one or more stacks
CREATE_FAILED	Unsuccessful creation of one or more stacks
DELETE_COMPLETE	Successful deletion of one or more stacks
DELETE_FAILED	Unsuccessful deletion of one or more stacks
ROLLBACK_COMPLETE	Successful removal after a failed stack creation
ROLLBACK_FAILED	Unsuccessful removal after a failed stack creation
UPDATE_COMPLETE	Successful update of one or more stacks
UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	Ongoing removal of old resources after a stack update
UPDATE_FAILED	Unsuccessful update of one or more stacks

# AWS Cloud Formation - Cross Stack Reference

```
//One Script
Outputs:
  AccountSG:
    Value: !Ref SomeSG
    Export:
      Name: SomeSGExported

//Another Script
EC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    SecurityGroups:
      - !ImportValue SomeSGExported
```

- Create modular Cloud Formation scripts
  - Break up a big Cloud Formation script into smaller scripts
  - For example: Network, Database and Web infrastructure separate scripts
- Output of one stack can be imported in another stack
  - Use **Export** output field and use **Fn::ImportValue** intrinsic function

# AWS Cloud Formation - Nested Stacks

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myStackWithParams:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: "https://s3.amazonaws.com/***/EC2ChooseAMI.template"
    Parameters:
      InstanceType: "t1.micro"
      KeyName: "mykey"
```

- Imagine your organization deploys a database for each application
  - How about defining standard CF script for creating databases in your organization?
  - How about reusing this in other CloudFormation scripts?
- Use resource type **AWS::CloudFormation::Stack**. Reference database stack.
- (BEST PRACTICE) Define standard templates for different type of resources
- Nested stacks can be hierarchical
  - Nested Stack can contain other nested stacks

# AWS Cloud Formation - Nested Stacks vs Cross Stack

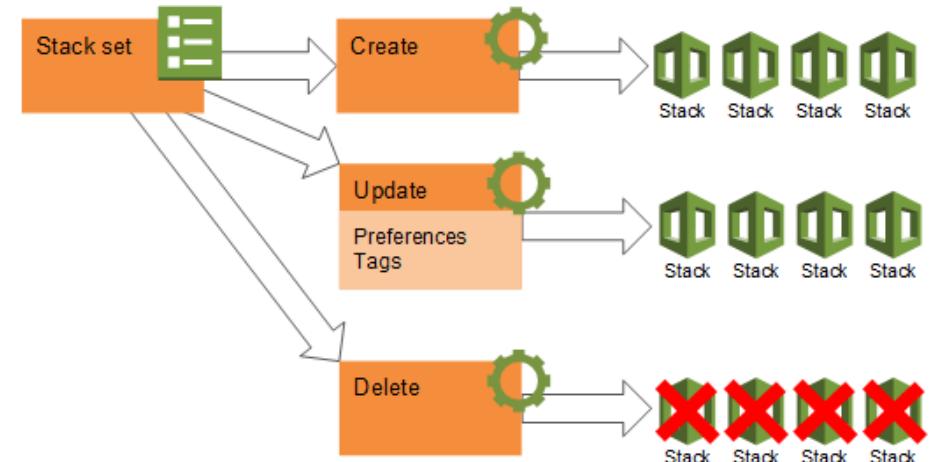
- Cross Stack:
  - Output from one stack is referenced in another stack
    - References another resource for reuse (SecurityGroup, VPC Subnet etc)
  - Resource is NOT recreated
- Nested stack:
  - Resource is recreated
  - Allows Reuse of templates



CloudFormation

# AWS Cloud Formation - Stack set

- You want to create same resources in different accounts across different regions:
  - Define a StackSet
  - Administrator account is required to create StackSets.
  - Trust relationship has to be established between administrator and target accounts



<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html>

*concepts.html*

# AWS Cloud Formation - Remember

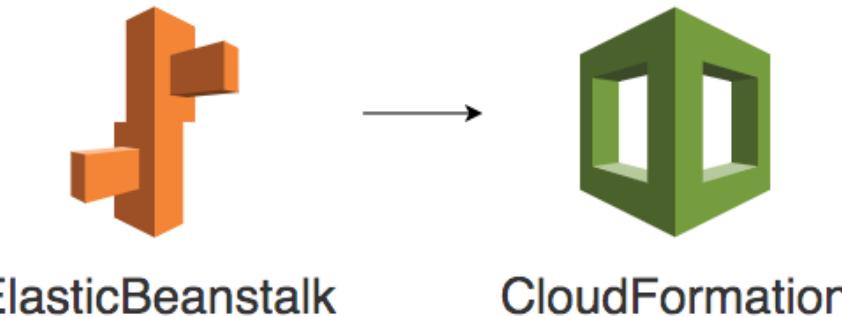
- Deleting a stack deletes all the associated resources
  - EXCEPT for resources with DeletionPolicy attribute set to "Retain"
  - You can enable termination protection for the entire stack
- You can execute CloudFormation templates from AWS CLI:
  - aws cloudformation create-stack/list-stacks/describe-stacks
- Python helper scripts simplify deployment on EC2 instances:
  - `cfn-init`: Retrieve resource metadata, install packages etc
  - `cfn-signal`: Enable you to synchronize with other resources in stack - Signal with a CreationPolicy or WaitCondition
  - `cfn-get-metadata`: Retrieve metadata for a resource or path to a specific key
  - `cfn-hup`: Check for updates to metadata and execute custom hooks



CloudFormation

# Cloud Formation vs AWS Elastic Beanstalk

In 28  
Minutes

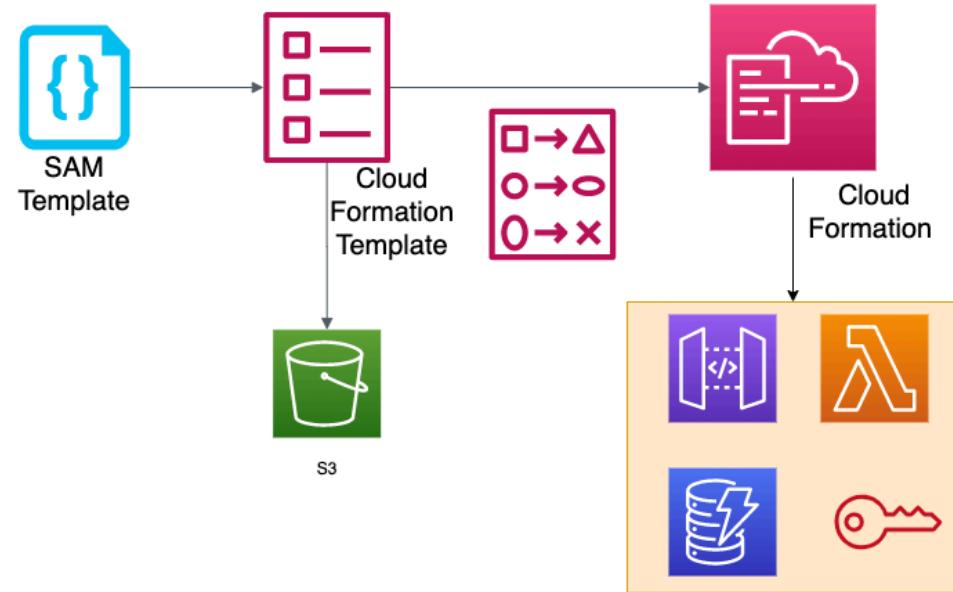


- (Do you know?) You can create an Elastic Beanstalk environment using CloudFormation!
- Think of Elastic Beanstalk as a pre-packaged CloudFormation template with a User Interface
  - You choose what you want
  - (Background) A Cloud Formation template is created and executed
  - The environment is ready!

# Serverless Application Model SAM

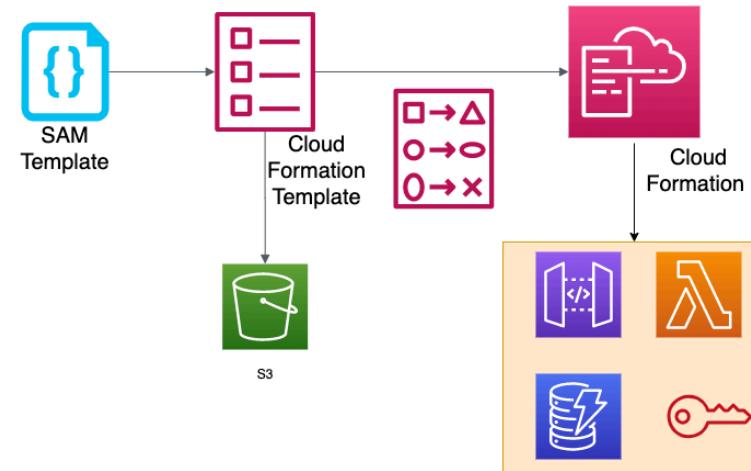
# Serverless Application Model

- Serverless projects can become a maintenance headache
  - 1000s of Lambda functions to manage, versioning, deployment etc
  - How to test serverless projects with Lambda, API Gateway and DynamoDB in your local?
  - How to ensure that your serverless projects are adhering to best practices?
    - Tracing (X-Ray), CI/CD(CodeBuild, CodeDeploy, CodePipeline) etc
- Welcome SAM - Serverless Application Model
  - Infrastructure as Code (IAC) for Serverless Applications



# Serverless Application Model - Approach & Advantages

- Open source framework for building serverless applications
- Define YAML file with resources (Functions, APIs, Databases..)
- (BEHIND THE SCENES) SAM config => CloudFormation scripts
- Benefits of SAM:
  - Single deployment configuration
  - Extends CloudFormation & hides complexity
  - Built-in best practices
  - Local debugging and testing
  - Benefits of IAC(Infrastructure as Code)
    - No Manual Errors, Version Control, Avoid configuration drift



# AWS SAM Template - Template Anatomy

```
//Main indicator that it is a serverless template – Mandatory
Transform: AWS::Serverless-2016-10-31

Globals: //Global attributes used across resources – Mandatory
         //set of globals
Description:
         //String
Metadata:
         //template metadata
Parameters:
         //set of parameters
Mappings:
         //set of mappings
Conditions:
         //set of conditions
         //conditionally create resources for different environments
Resources: //Mandatory
         //AWS CloudFormation resources and AWS SAM resources
Outputs:
         //set of outputs
```

# AWS SAM - Supported Resources

- Container Application
  - AWS::Serverless::Application
- Lambda Functions and Layers
  - AWS::Serverless::Function
  - AWS::Serverless::LayerVersion
- API Gateways
  - AWS::Serverless::Api
  - AWS::Serverless::HttpApi
- DynamoDB Tables
  - AWS::Serverless::SimpleTable
- Step Functions
  - AWS::Serverless::StateMachine
- For other resources, use AWS CloudFormation definition

# Example SAM Template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  CreateThumbnail:
    Type: AWS::Serverless::Function
    Properties:
      Handler: handler
      Runtime: runtime
      Timeout: 60
      Policies: AWSLambdaExecute
    Events:
      CreateThumbnailEvent:
        Type: S3
        Properties:
          Bucket: !Ref SrcBucket
          Events: s3:ObjectCreated:*
  SrcBucket:
    Type: AWS::S3::Bucket
```

# Key SAM CLI Commands

- `sam init` - Initializes a serverless application with an AWS SAM template
- `sam validate` - Validate SAM Template
- `sam build` - Builds a serverless application, and prepares it for subsequent steps in your workflow
- `sam local invoke` - Invokes a local Lambda function
- `sam package` - Bundle your application code and dependencies into a "deployment package"
- `sam deploy` - Deploy SAM Application to AWS (also does package)
- `sam logs` - Fetches logs that are generated by your Lambda function.
- `sam publish` - Publish an AWS SAM application to the AWS Serverless Application Repository.

# AWS SAM Policy Templates

- How do you give access to AWS resources to your Lambda functions?
- SAM allows you to choose from a list of pre-configured policy templates
- Examples:
  - SQSPollerPolicy
  - CloudWatchPutMetricPolicy
  - DynamoDBCrudPolicy
  - DynamoDBReadPolicy
  - DynamoDBWritePolicy

# SAM Policy Template - Syntax

```
//Syntax
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
      - PolicyTemplateName1:          # Policy template with placeholder value
        Key1: Value1
      - PolicyTemplateName2: {}       # Policy template with no placeholder value

//Example
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
  Policies:
    - SQSPollerPolicy:
      QueueName: !GetAtt MyQueue.QueueName
```

# SAM - Scenario Questions

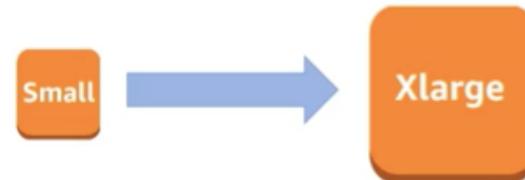
Scenario	Solution
Where is the deployment package created by SAM stored?	S3 bucket
How much does AWS SAM cost?	No cost. You only pay for the resources provisioned.
How do you deploy an application built using SAM from CLI?	<ol style="list-style-type: none"><li>1. sam package &amp;&amp; sam deploy (OR)</li><li>2. aws cloudformation package &amp;&amp; aws cloudformation deploy</li></ol>
Which sections of the SAM template are required?	Transform and Resources

# EC2 - Advanced

# Scalability

- A system is handling 1000 transactions per second. Load is expected to increase 10 times in the next month
  - Can we handle a growth in users, traffic, or data size without any drop in performance?
  - Does ability to serve more growth increase proportionally with resources?
- Ability to **adapt** to changes in demand (users, data)
- What are the options that can be considered?
  - Deploy to a bigger instance with bigger CPU and more memory
  - Increase the number of application instances and setup a load balancer
  - And a lot more.

# Vertical Scaling



- Deploying application/database to **bigger instance**:
  - A larger hard drive
  - A faster CPU
  - More RAM, CPU, I/O, or networking capabilities
- There are limits to vertical scaling

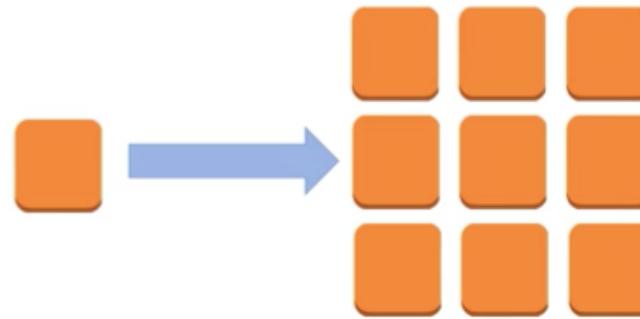
# Vertical Scaling for EC2

Instance	vCPU*	CPU Credits / hour	Mem (GiB)	Storage	Network Performance
t2.nano	1	3	0.5	EBS-Only	Low
t2.micro	1	6	1	EBS-Only	Low to Moderate
t2.small	1	12	2	EBS-Only	Low to Moderate
t2.medium	2	24	4	EBS-Only	Low to Moderate
t2.large	2	36	8	EBS-Only	Low to Moderate
t2.xlarge	4	54	16	EBS-Only	Moderate
t2.2xlarge	8	81	32	EBS-Only	Moderate

- Increasing EC2 instance size:
  - *t2.micro* to *t2.small* or
  - *t2.small* to *t2.2xlarge* or
  - ...

# Horizontal Scaling

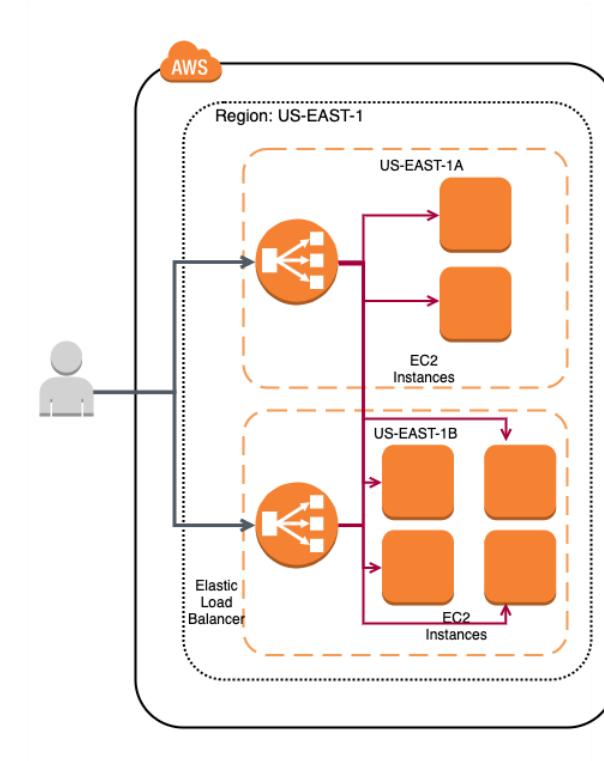
In 28  
Minutes



- Deploying multiple instances of application/database
- (Typically but not always) Horizontal Scaling is preferred to Vertical Scaling:
  - Vertical scaling has limits
  - Vertical scaling can be expensive
  - Horizontal scaling increases availability
- (BUT) Horizontal Scaling needs additional infrastructure:
  - Load Balancers etc.

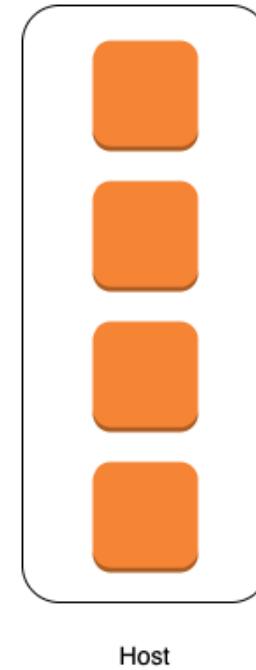
# Horizontal Scaling for EC2

- Distribute EC2 instances
  - in a single AZ
  - in multiple AZs in single region
  - in multiple AZs in multiple regions
- **Auto scale:** Auto Scaling Group
- **Distribute load :** Elastic Load Balancer



# EC2 Tenancy - Shared vs Dedicated

- **Shared Tenancy (Default)**
  - Single host machine can have instances from multiple customers
- **EC2 Dedicated Instances**
  - Virtualized instances on hardware dedicated to one customer
  - You do NOT have visibility into the hardware of underlying host
- **EC2 Dedicated Hosts**
  - Physical servers dedicated to one customer
  - You have visibility into the hardware of underlying host (sockets and physical cores)
  - (Use cases) Regulatory needs or server-bound software licenses like Windows Server, SQL Server

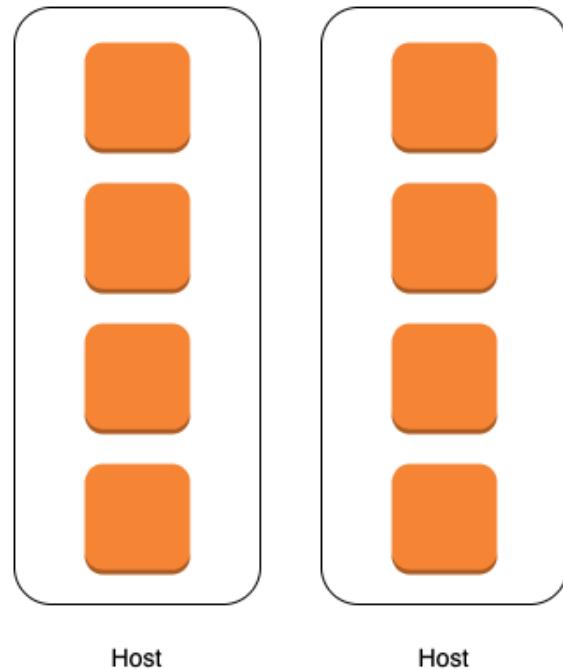


# EC2 Pricing Models Overview

Pricing Model	Description	Details
On Demand	Request when you want it	Flexible and Most Expensive
Spot	Quote the maximum price	Cheapest (upto 90% off) BUT NO Guarantees
Reserved	Reserve ahead of time	Upto 75% off. 1 or 3 years reservation.
Savings Plans	Commit spending \$X per hour on (EC2 or AWS Fargate or Lambda)	Upto 66% off. No restrictions. 1 or 3 years reservation.

# EC2 On-Demand

- On demand resource provisioning - Use and Throw!
- Highest cost and highest flexibility
- This is what we have been using until now in this course
- **Ideal for:**
  - A web application which receives spiky traffic
  - A batch program which has unpredictable runtime and cannot be interrupted
  - A batch program being moved from on-premises to cloud for the first time



# EC2 Spot instances

- (Old Model) Bid a price. Highest bidder wins
- (New Model) Quote your maximum price. Prices decided by long term trends.
- Up to 90% off (compared to On-Demand)
- Can be terminated with a **2 minute** notice
- Ideal for **Non time-critical workloads** that can **tolerate interruptions** (fault-tolerant)
  - A batch program that does not have a strict deadline AND can be stopped at short notice and re-started

# EC2 Reserved Instances

- Reserve EC2 instances ahead of time!
- Get upto 75% OFF!
- **Payment models:**
  - No Upfront - \$0 upfront. Pay monthly installment.
  - Partial Upfront - \$XYZ upfront. Pay monthly installment
  - All Upfront - Full amount upfront. \$0 monthly installment.
  - **Cost wise :** Earlier you pay, more the discount. All Upfront < Partial Upfront < No Upfront
  - A difference upto 5%

# EC2 Savings Plans

- EC2 Compute Savings Plans
  - **Commitment** : I would spend X dollars per hour on AWS compute resources (Amazon EC2 instances, AWS Fargate and/or AWS Lambda) for a 1 or 3 year period
  - Up to 66% off (compared to on demand instances)
  - Provides **complete flexibility**:
    - You can change instance family, size, OS, tenancy or AWS Region of your Amazon EC2 instances
    - You can switch between Amazon EC2, AWS Fargate and/or AWS Lambda
- EC2 Instance Savings Plans
  - **Commitment** : I would spend X dollars per hour on Amazon EC2 instances of a specific instance family (General Purpose, for example) within a specific region (us-east-1, for example)
  - Up to 72% off (compared to on demand instances)
  - You can switch operating systems (Windows to Linux, for example)

# EC2 Pricing Models Overview

<https://www.ec2instances.info/>

## Pricing Model

### Usecases

#### On Demand

Spiky workloads.

#### Spot

Cost sensitive, Fault tolerant, Non immediate workloads.

#### Reserved

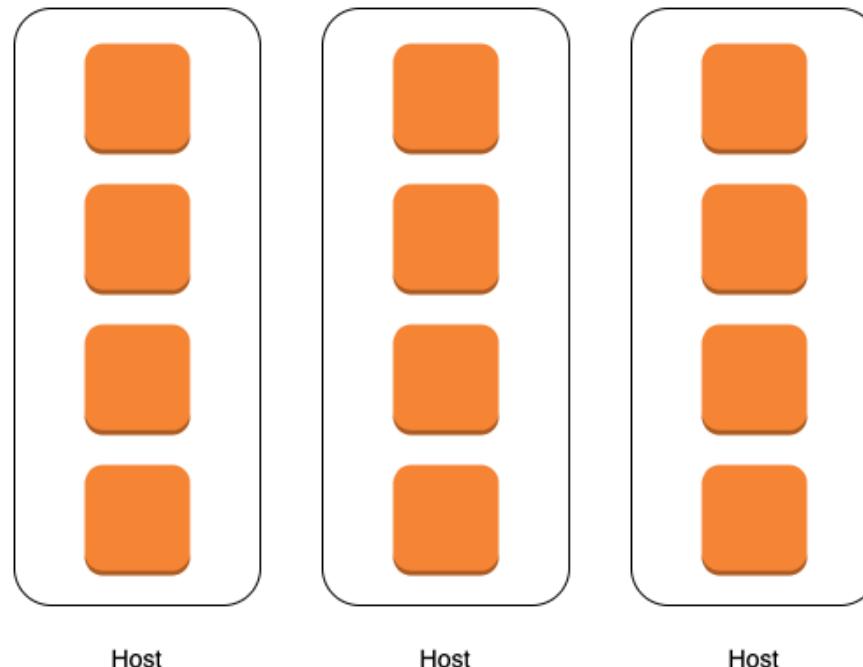
Constant workloads that run all the time.

#### Savings Plans

Constant workloads that run all the time and you want more flexibility.

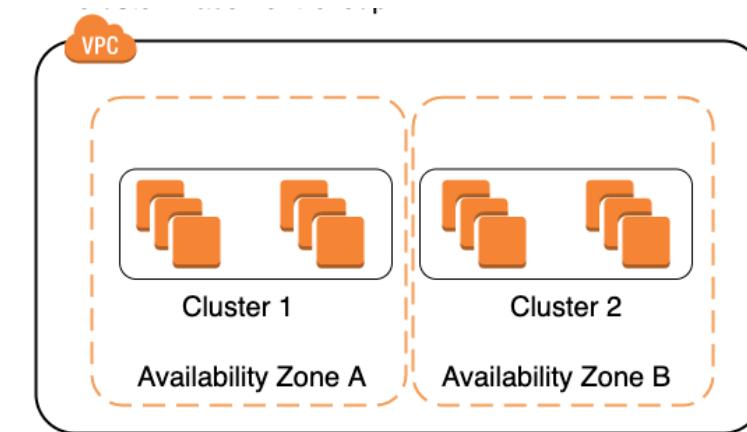
# EC2 Placement Groups

- Certain usecases need control over placement of a group of EC2 instances
  - Low latency network communication
  - High availability
- You DO NOT want EC2 to decide that for you!
- Go for EC2 placement groups!
  - Cluster (low network latency )
  - Spread (avoid simultaneous failures)
  - Partition (multiple partitions with low network latency)



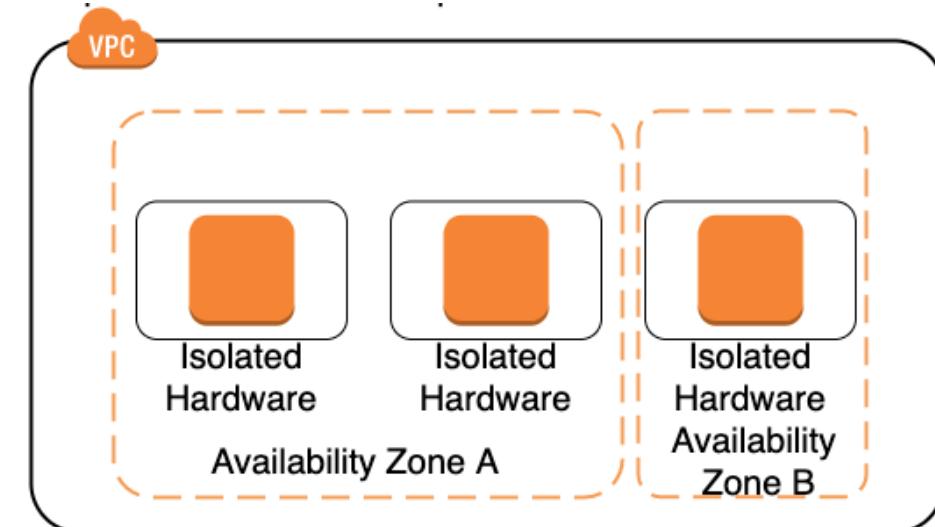
# EC2 Cluster Placement Group

- When low latency network communication between EC2 instances is critical
- Example: Big Data or High Performance Computing needing extreme low latency
- EC2 instances placed near to each other in single AZ
- **High Network Throughput:** EC2 instances can use 10 Gbps or 25Gbps network
- (Disadvantage) Low Availability (Rack crashes => All EC2 instances fail)

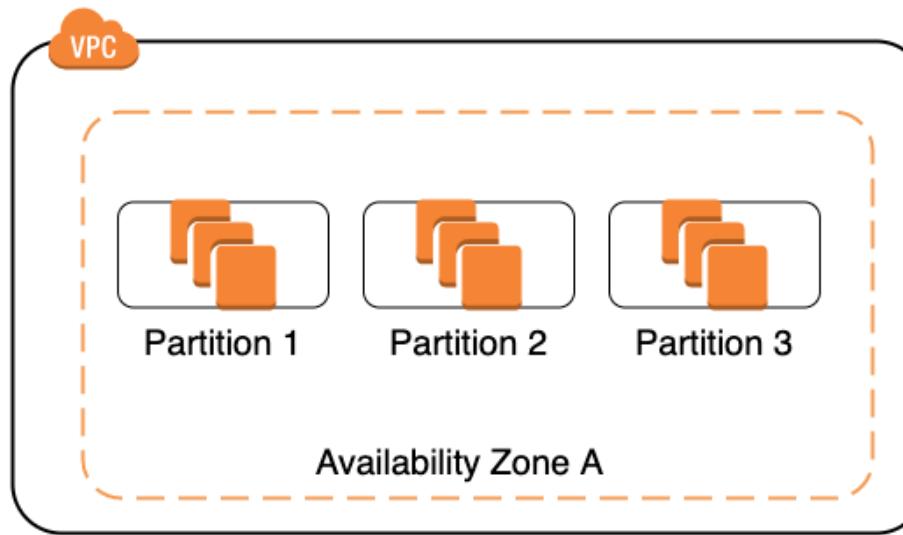


# EC2 Spread Placement Group

- Spread EC2 instances across distinct racks
- Each rack has its own network and power source
- **Avoid simultaneous failures** of EC2 instances
- Can be spread across different AZs in same region
- Maximum of seven running instances per AZ in a spread placement group



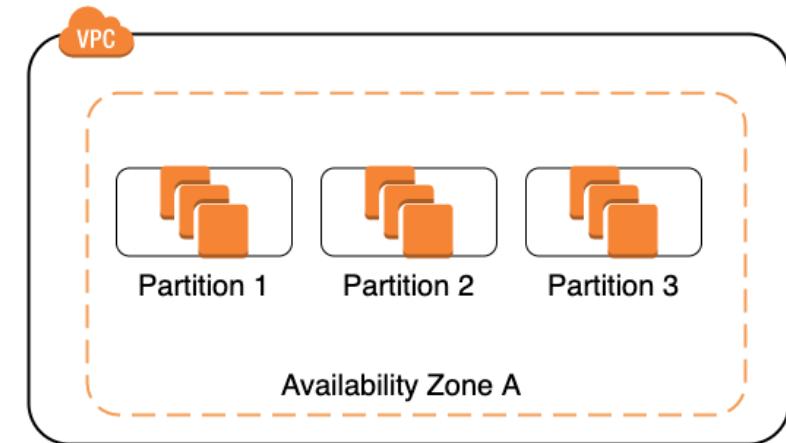
# EC2 Partition Placement Group - Use Case



- In large distributed and replicated workloads (HDFS, HBase, and Cassandra), EC2 instances need to be **divided into multiple groups**:
  - Low latency communication between instances in a group
  - Each group is placed on a different rack

# EC2 Partition Placement Group

- A partition is a group of EC2 instances
- Each partition will be **placed on a different rack**
- You can choose the partition where EC2 instance is launched into
- Can be spread across **different AZs** in same region
- Maximum of seven partitions per Availability Zone per group



# EC2 Placement Groups - Best Practice

- **Insufficient capacity error** can happen when:
  - New instances are added in (OR)
  - More than one instance type is used (OR)
  - An instance in placement group is stopped and started
- If you receive a capacity error:
  - Stop and start all instances in the placement group (OR)
  - Try to launch the placement group again
  - Result: Instances may be migrated to a rack that has capacity for all the requested instances
- **Recommendation:**
  - Have only one instance type in a launch request AND
  - Launch all instances in a single launch request together

# Elastic Network Interface

- Logical networking component that represents a **virtual network card**
- Support IPv4 (110.120.120.145) and IPv6 (2001:0db8:85a3:0000:0000:8a2e:0370:7334)
- Each Elastic Network Interface can provide:
  - One primary and multiple secondary private IP addresses
  - One public address
  - One **Elastic IP address** per private IPv4 address
  - One or more **security groups**



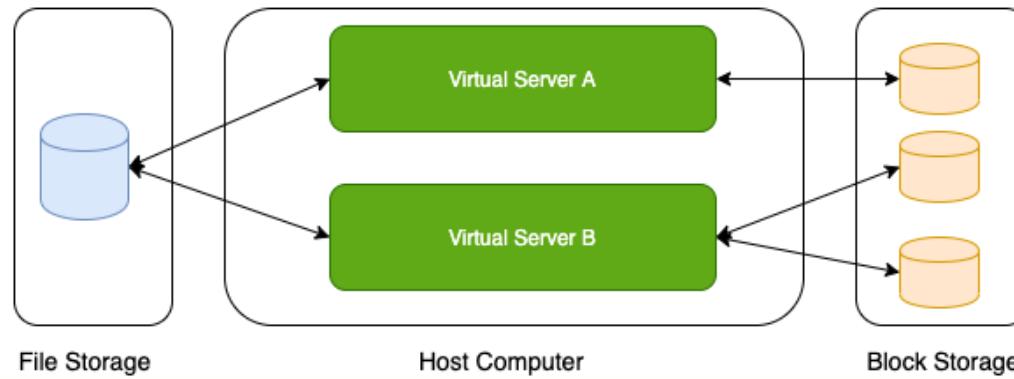
# Elastic Network Interface - Primary and Secondary

- Each EC2 instance is connected to primary network interface (**eth0**)
- You can create and attach a secondary network interface - **eth1**
- Allows an instance to be **dual homed** - present in two subnets in a VPC
- Used to create a **management network** or a low budget high availability solution
- Terminology :
  - **Hot attach**: Attaching ENI when EC2 instance is running
  - **Warm attach**: Attaching ENI when EC2 instance is stopped
  - **Cold attach**: Attaching ENI at launch time of EC2 instance
- Demo



# Storage in Cloud - Block Storage and File Storage

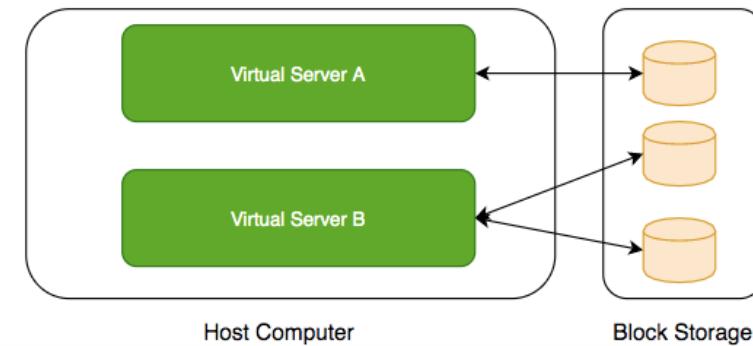
# Storage Types - Block Storage and File Storage



- What is the type of storage of your hard disk?
  - **Block Storage**
- You've created a file share to share a set of files with your colleagues in a enterprise. What type of storage are you using?
  - **File Storage**

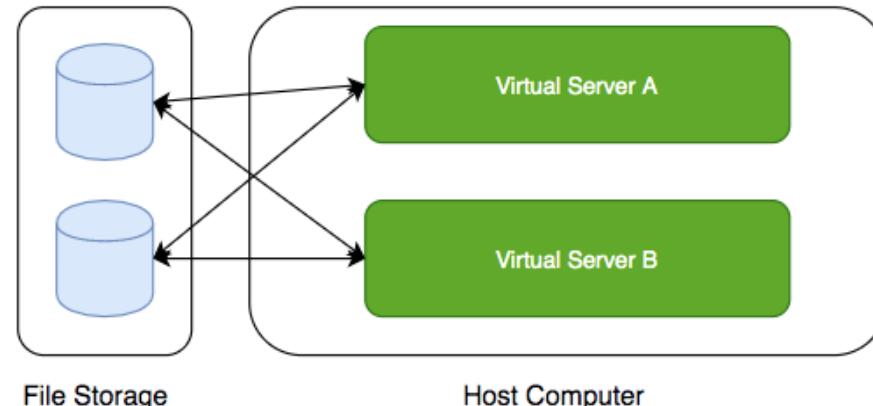
# Block Storage

- Use case: Hard-disks attached to your computers
- Typically, ONE Block Storage device can be connected to ONE virtual server
- HOWEVER, you can connect multiple different block storage devices to one virtual server



# File Storage

- Media workflows need huge shared storage for supporting processes like video editing
- Enterprise users need a quick way to share files in a secure and organized way
- These file shares are shared by several virtual servers



# AWS - Block Storage and File Storage

In 28  
Minutes



Amazon EFS

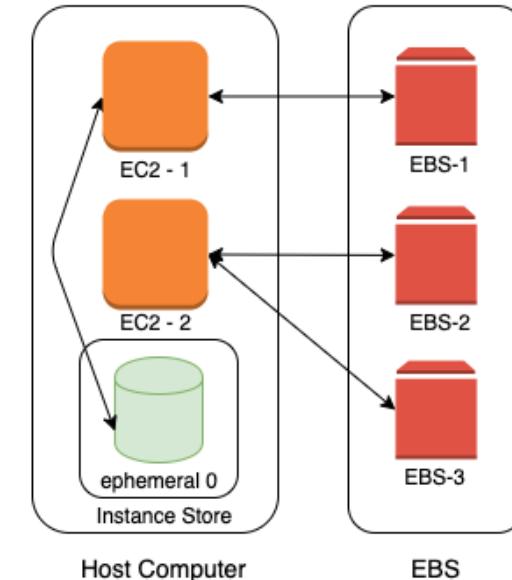


Amazon EBS

- **Block Storage:**
  - Amazon Elastic Block Store (EBS)
  - Instance store
- **File Storage:**
  - Amazon EFS (for Linux instances)
  - Amazon FSx Windows File Servers
  - Amazon FSx for Lustre (high performance use cases)

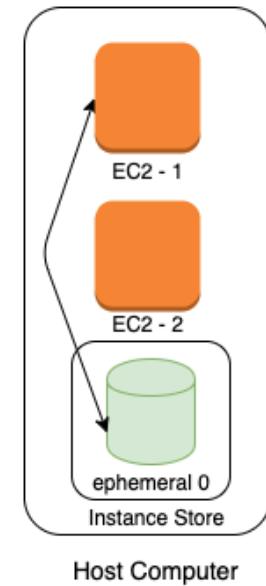
# EC2 - Block Storage

- Two popular types of block storage can be attached to EC2 instances:
  - Elastic Block Store (EBS)
  - Instance Store
- **Instance Stores** are physically attached to the EC2 instance
  - Temporary data
  - Lifecycle tied to EC2 instance
- **Elastic Block Store (EBS)** is network storage
  - More durable
  - Lifecycle NOT tied to EC2 instance



# Instance Store

- Physically attached to your EC2 instance
- Ephemeral storage
  - Temporary data.
  - Data is lost when hardware fails or an instance is terminated.
  - Use case: cache or scratch files
- Lifecycle is tied to EC2 instance
- Only some of the EC2 instance types support Instance Store



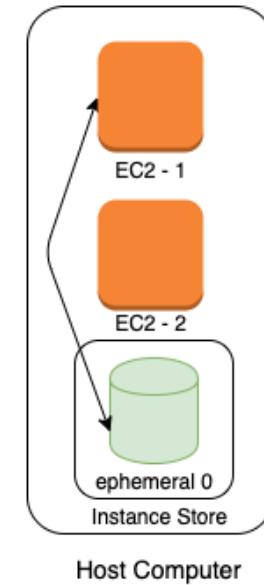
# Instance Store - Advantages and Disadvantages

- **Advantages**

- Very Fast I/O (2-100X of EBS)
- (Cost Effective) **No extra cost**. Cost is included in the cost of EC2 instance
- Ideal for storing **temporary information** - cache, scratch files etc

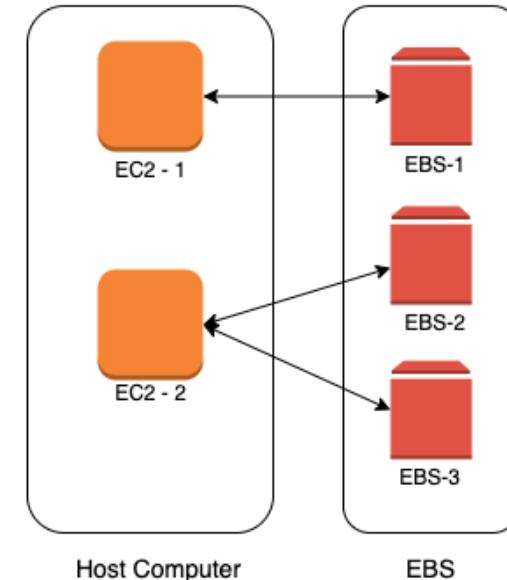
- **Disadvantages**

- Slow boot up (up to 5 minutes)
- **Ephemeral storage** (data is lost when hardware fails or instance is terminated)
- **CANNOT** take a snapshot or restore from snapshot
- Fixed size based on instance type
- You cannot detach and attach it to another EC2 instance



# Amazon Elastic Block Store (EBS)

- Network block storage attached to your EC2 instance
- Provisioned capacity
- Very flexible.
  - Increase size when you need it - when attached to EC2 instance
- Independent lifecycle from EC2 instance
  - Attach/Detach from one EC2 instance to another
- 99.999% *Availability* & replicated within the same AZ
- **Use case :** Run your custom database



# Amazon EBS vs Instance Store

Feature	Elastic Block Store (EBS)	Instance Store
Attachment to EC2 instance	As a network drive	Physically attached
Lifecycle	Separate from EC2 instance	Tied with EC2 instance
Cost	Depends on provisioned size	Zero (Included in EC2 instance cost)
Flexibility	Increase size	Fixed size
I/O Speed	Lower (network latency)	2-100X of EBS
Snapshots	Supported	Not Supported
Use case	Permanent storage	Ephemeral storage
Boot up time	Low	High

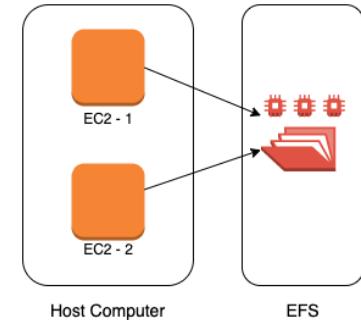
# Hard Disk Drive vs Solid State Drive

*Amazon EBS offers HDD and SSD options!  
How do you choose between them?*

Feature	HDD(Hard Disk Drive)	SSD(Solid State Drive)
Performance - IOPS	Low	High
Throughput	High	High
Great at	Large sequential I/O operations	Small, Random I/O operations & Sequential I/O
Recommended for	Large streaming or big data workloads	Transactional workloads
Cost	Low	Expensive
Boot Volumes	Not Recommended	Recommended

# Amazon EFS

- Petabyte scale, Auto scaling, Pay for use shared file storage
- Compatible with Amazon EC2 Linux-based instances
- (Use cases) Home directories, file share, content management
- (Alternative) Amazon FSx for Lustre
  - File system optimized for performance
  - High performance computing (HPC) and media processing use cases
  - Automatic encryption at-rest and in-transit
- (Alternative) Amazon FSx Windows File Servers
  - Fully managed Windows file servers
  - Accessible from Windows, Linux and MacOS instances
  - Integrates with Microsoft Active Directory (AD) to support Windows-based environments and enterprises.
  - Automatic encryption at-rest and in-transit

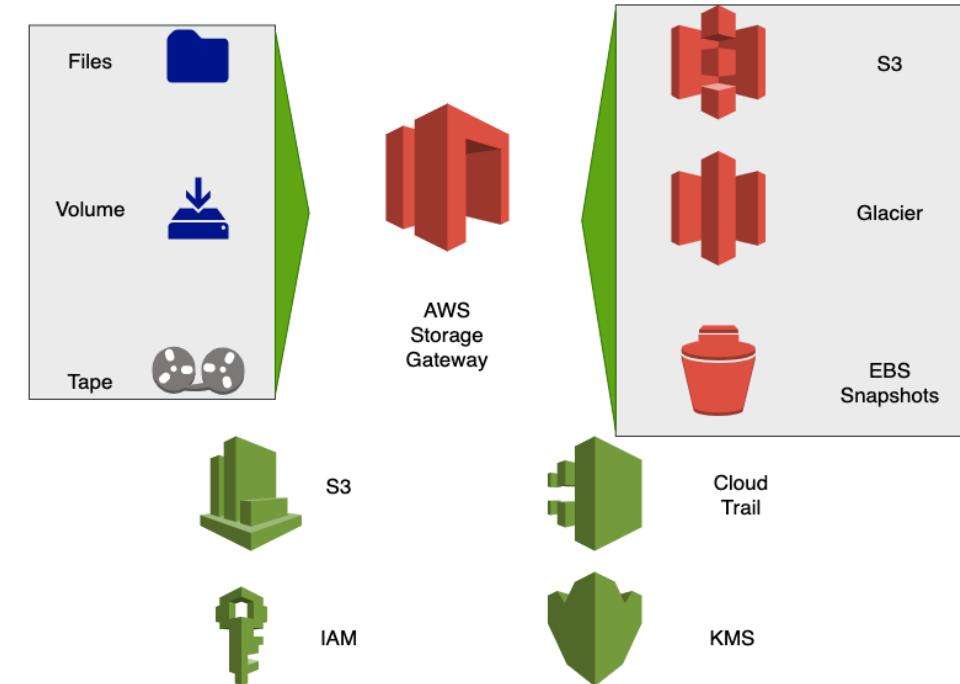


# Review of storage options

Type	Examples	Latency	Throughput	Shareable
Block	EBS, Instance Store	Lowest	Single	Attached to one instance at a time. Take snapshots to share.
File	EFS, FSx Windows, FSx for Lustre	Low	Multiple	Yes
Object	S3	Low	Web Scale	Yes
Archival	Glacier	Minutes to hours	High	No

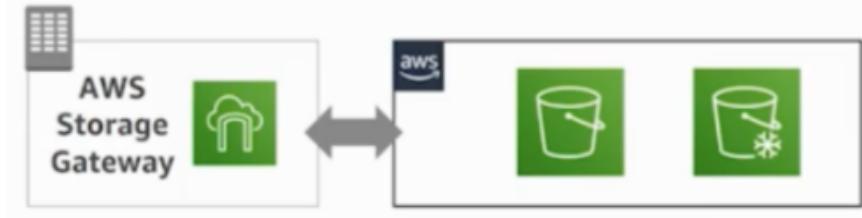
# AWS Storage Gateway

- Hybrid storage (cloud + on premise)
- Unlimited cloud storage for on-premise software applications and users with good performance
- (Remember) Storage Gateway and S3 Glacier **encrypt data by default**
- **Three Options**
  - AWS Storage File Gateway
  - AWS Storage Tape Gateway
  - AWS Storage Volume Gateway



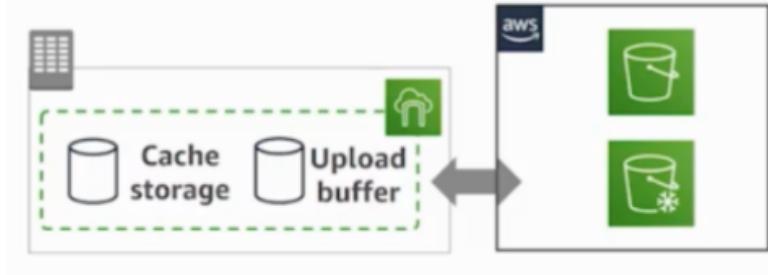
# AWS Storage File Gateway

- **Problem Statement:** Large on-premise file share with terabytes of data
  - Users put files into file share and applications use the files
  - Managing it is becoming expensive
  - Move the file share to cloud without performance impact
- AWS Storage File Gateway provides cloud storage for your file shares
  - Files stored in Amazon S3 & Glacier



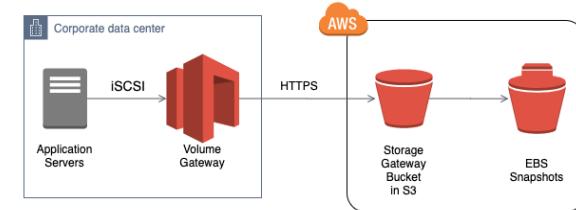
# AWS Storage Tape Gateway

- Tape backups used in enterprises (archives)
  - Stored off-site - expensive, physical wear and tear
- AWS Storage Tape Gateway - Avoid physical tape backups
- No change needed for tape backup infrastructure
- Backup data to virtual tapes (actually, Amazon S3 & Glacier)



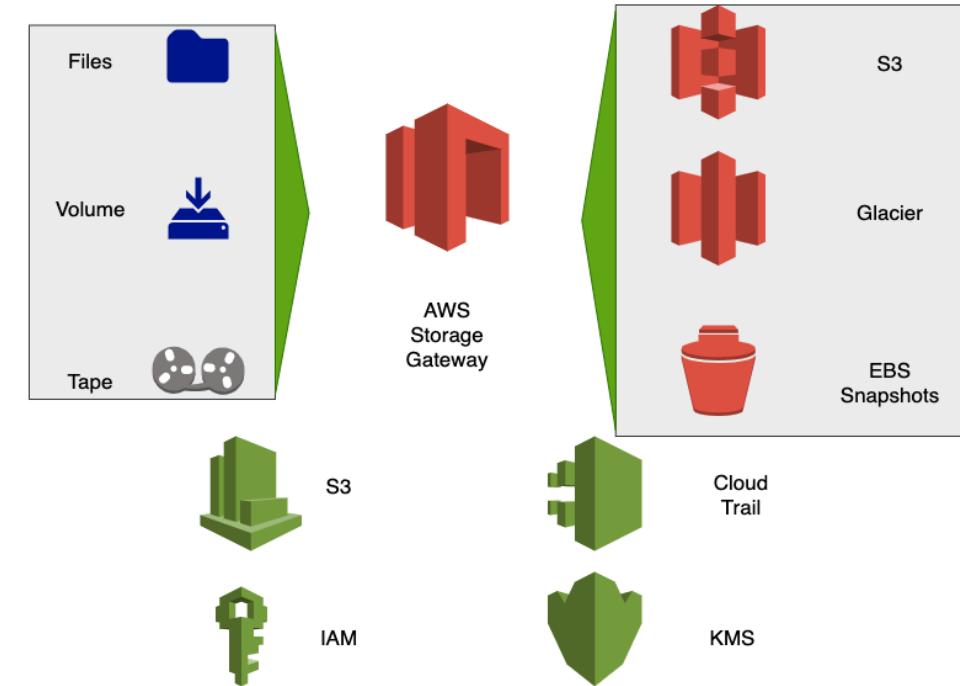
# AWS Storage Volume Gateway

- **Volume Gateway** : Move block storage to cloud
- Automate backup and disaster recovery
- Use cases: Backup and disaster recovery, Migration of application data
- (Option 1) **Cached** (Gateway Cached Volumes):
  - Primary Data Store - AWS - Amazon S3
  - On-premise cache stores frequently accessed data
- (Option 2) **Stored** (Gateway Stored Volumes):
  - Primary Data Store - On-Premises
  - Asynchronous copy to AWS
  - Stored as EBS snapshots



# AWS Storage Gateway - Summary

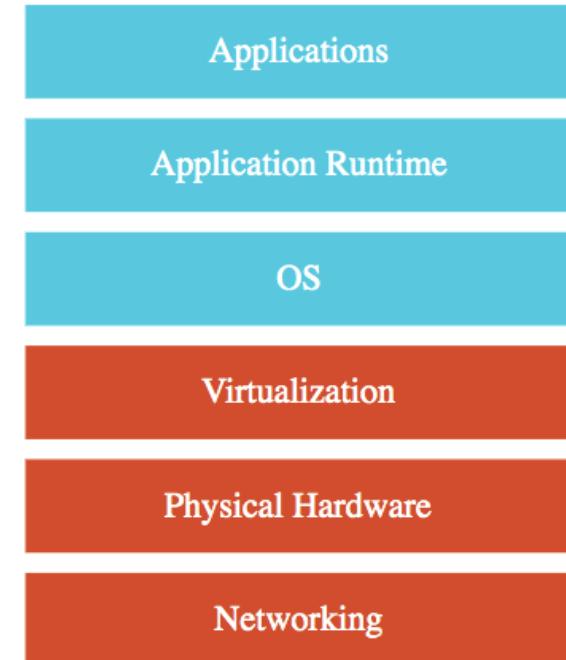
- Key to look for : Hybrid storage (cloud + on premise)
- File share moved to cloud => **AWS Storage File Gateway**
- Tape Backups on cloud => **AWS Storage Tape Gateway**
- Volume Backups on cloud (Block Storage) => **AWS Storage Volume Gateway**
  - High performance => **Stored**
  - Otherwise => **Cached**



# AWS Managed Services

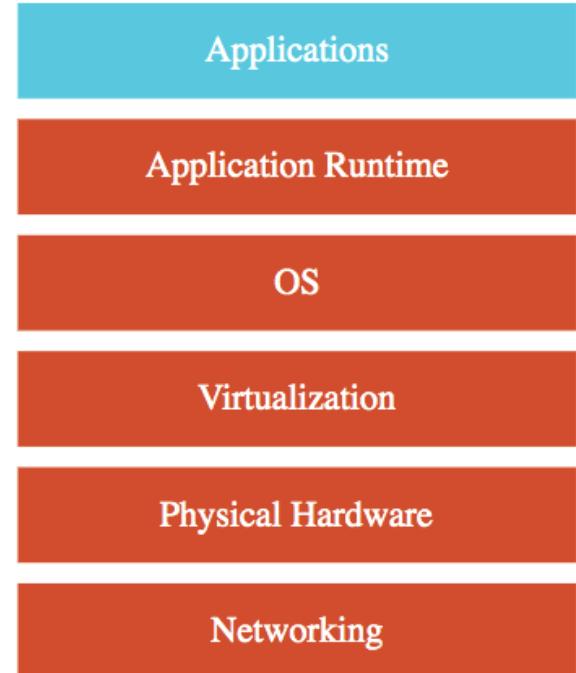
# IAAS (Infrastructure as a Service)

- Use **only infrastructure** from cloud provider
- Also called "**Lift and Shift**"
- **Example:** Using EC2 to deploy your applications or databases
- You are responsible for:
  - Application Code and Runtime
  - Configuring load balancing
  - Auto scaling
  - OS upgrades and patches
  - Availability
  - etc.. ( and a lot of things!)



# PAAS (Platform as a Service)

- Use a platform provided by cloud
- **Cloud provider** is responsible for:
  - OS (incl. upgrades and patches)
  - Application Runtime
  - Auto scaling, Availability & Load balancing etc..
- **You** are responsible for:
  - Application code
  - Configuration
- **CAAS (Container as a Service)**: Containers instead of Applications
- **FAAS (Function as a Service) or Serverless**: Functions instead of Applications



# AWS Managed Service Offerings

In 28  
Minutes



ELB



ECS



Amazon RDS

- **Elastic Load Balancing** - Distribute incoming traffic across multiple targets
- **AWS Elastic Beanstalk** - Run and Manage Web Apps
- **Amazon Elastic Container Service (ECS)** - Containers orchestration on AWS
- **AWS Fargate** - Serverless compute for containers
- **Amazon Elastic Kubernetes Service (EKS)** - Run Kubernetes on AWS
- **Amazon RDS** - Relational Databases - MySQL, Oracle, SQL Server etc
- And a lot more...

# AWS Elastic Beanstalk

- **Simplest way** to deploy and scale your web application in AWS
  - Provides end-to-end web application management
- **Supports:**
  - Programming languages (Go, Java, Node.js, PHP, Python, Ruby)
  - Application servers (Tomcat, Passenger, Puma)
  - Docker containers (Single and Multi Container Options)
- **No usage charges** - Pay only for AWS resources you provision
- **Features:**
  - Automatic load balancing
  - Auto scaling
  - Managed platform updates
  - Application health monitoring

# AWS Elastic Beanstalk Demo

- Deploy an application to cloud using AWS Beanstalk

# AWS Elastic Beanstalk Concepts

- **Application** - A container for environments, versions and configuration
- **Application Version** - A specific version of deployable code (stored in S3)
- **Environment** - An application version deployed to AWS resources. You can have multiple environments running different application versions for the same application.
- **Environment Tier:**
  - For batch applications, use **worker tier**
  - For web applications, use **web server tier**

# AWS Elastic Beanstalk - Web Server Environments

- Single-instance environments: EC2 + Elastic IP
- Load-balanced environments: ELB + ASG + EC2
- (OPTIONAL) Add database to Elastic Beanstalk Env:
  - Use environment properties to connect to database
    - RDS\_HOSTNAME, RDS\_PORT, RDS\_DB\_NAME, RDS\_USERNAME, RDS\_PASSWORD
- Lifecycle of database tied to Elastic Beanstalk Env:
  - If you delete Elastic Beanstalk environment, database also deleted
    - (WORKAROUND): Enable Delete Protection on RDS
    - (WORKAROUND): Take Database Snapshot and Restore
  - NOT RECOMMENDED for Production Deployment



# AWS Elastic Beanstalk - Worker environments

In 28  
Minutes



```
version: 1
cron:
  - name: "audit"
    url: "/audit"
    schedule: "0 23 * * *"
```

- **Architecture:** ASG + EC2 + SQS
- Process messages from SQS queues
- Trigger auto scaling using AWS CloudWatch alarms
- **Schedule tasks using cron.yaml**

# AWS Elastic Beanstalk - Source Bundle

```
|-- app.war
|-- .ebextensions
|   |-- xray.config
`-- healthcheckurl.config
```

- **ONE AND ONLY ONE ZIP file or WAR file**
  - To deploy multiple WAR files, include them in a ZIP file
- **(MAX SIZE) 512 MB**
- **DO NOT have any parent folder**
  - subdirectories are fine

# AWS Elastic Beanstalk - Configuration Files

```
|-- app.war
|-- .ebextensions
|   |-- xray.config
|   '-- healthcheckurl.config

option_settings:
  - namespace: aws:elasticbeanstalk:application
    option_name: Application Healthcheck URL
    value: /health
```

- Customize your Elastic Beanstalk environment:
  - YAML- or JSON-formatted documents
    - .config file extension
    - Example: `xray.config`
  - Placed in a folder named `.ebextensions` at root of your source bundle
    - Example: `.ebextensions\xray.config`, `.ebextensions\elb.config`
  - Configure Elastic Load Balancer, Enable X-Ray etc
  - Supports creating CloudFormation scripts as well!

# AWS Elastic Beanstalk - Deployment methods

- Move from V1 to New Version V2
  - **All at once** – Deploy V2 to all existing instances in a **SINGLE** batch.
  - **Rolling** – Deploy V2 to existing instances in multiple batches. Deployment of next batch starts after current batch is successful.
  - **Rolling with additional batch** – Deploy V2 to new/existing instances in multiple batches. Launches a new batch with V2 first. Each batch with V2 will replace existing instances with V1 deployed.
  - **Immutable** – Second ASG created with V2. New version and Old version serve traffic until all V2 instances pass health checks.
  - **Traffic splitting** – Canary testing approach. Deploy V2 to few new instances. Send a portion of traffic to V2 (While serving majority of users from V1).
  - **(ADDITIONAL OPTION) BLUE GREEN with SWAP URL** - Create New Environment with V2 instances. Test them. SWAP URL of V1 environment with V2 environment. One time switch!
    - You can clone V1 environment and deploy V2 **all at once!**

# AWS Elastic Beanstalk - Deployment methods - AWS Documentation

Method	Impact of failed deployment	Deploy time	Zero downtime	No DNS change	Rollback process	Code deployed to
All at once	Downtime	⌚	🚫 No	✅ Yes	Manual redeploy	Existing instances
Rolling	Single batch out of service; any successful batches before failure running new application version	⌚⌚ †	✅ Yes	✅ Yes	Manual redeploy	Existing instances
Rolling with an additional batch	Minimal if first batch fails; otherwise, similar to Rolling	⌚⌚ ⌚†	✅ Yes	✅ Yes	Manual redeploy	New and existing instances
Immutable	Minimal	⌚⌚ ⌚⌚	✅ Yes	✅ Yes	Terminate new instances	New instances
Traffic splitting	Percentage of client traffic routed to new version temporarily impacted	⌚⌚ ⌚⌚ ††	✅ Yes	✅ Yes	Reroute traffic and terminate new instances	New instances
Blue/green	Minimal	⌚⌚ ⌚⌚	✅ Yes	🚫 No	Swap URL	New instances

# AWS Elastic Beanstalk - Deploying new version

- **STEP 1 : Create Application Version**
  - Application Version is stored in S3
- **STEP 2 : Update Environment to use the new Application Version**
- **Options:**
  - **FROM UI (AWS Management Console)**
    - Upload and Deploy => Select Source Zip
  - **FROM Elastic Beanstalk command line interface (EB CLI)**
    - Create new app version and deploy
      - eb appversion
      - eb deploy
- (Remember) Deployments consumes storage & application version quota
  - Configure **application version lifecycle policy** (from Elastic Beanstalk CLI and APIs) to delete old application versions:
    - Option 1: How many versions do you want to retain?
    - Option 2: Do you want to delete from Amazon S3 as well?

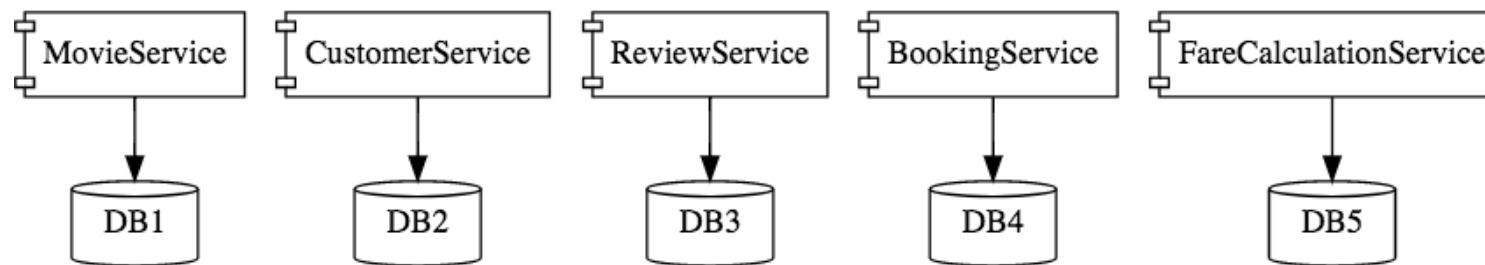
# AWS Elastic BeanStalk - Remember

- You retain **full control** over AWS resources created
- **Ideal for simple web applications**
  - NOT ideal for microservices architectures
- Logs can be stored in Amazon S3 or in CloudWatch Logs
- You can choose to **apply patches and platform updates automatically**
- Metrics are send to Amazon CloudWatch
- You can configure SNS notifications based on health
- Delete your environment!

# Containers and Container Orchestration

# Microservices

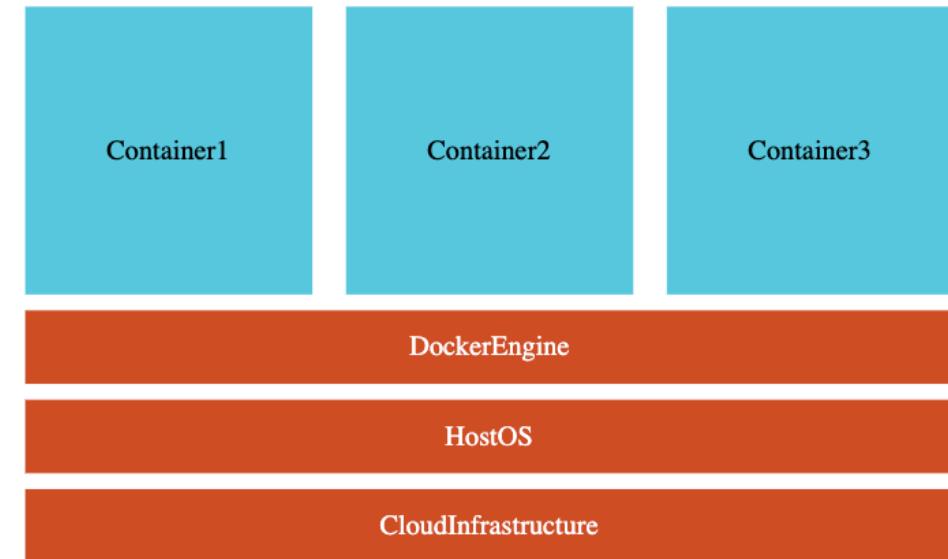
In 28  
Minutes



- Enterprises are heading towards microservices architectures
- Build small focused microservices
- **Flexibility to innovate** and build applications in different programming languages (Go, Java, Python, JavaScript, etc)
- **BUT deployments become complex!**
- How can we have **one way of deploying** Go, Java, Python or JavaScript .. microservices?
  - Enter **containers!**

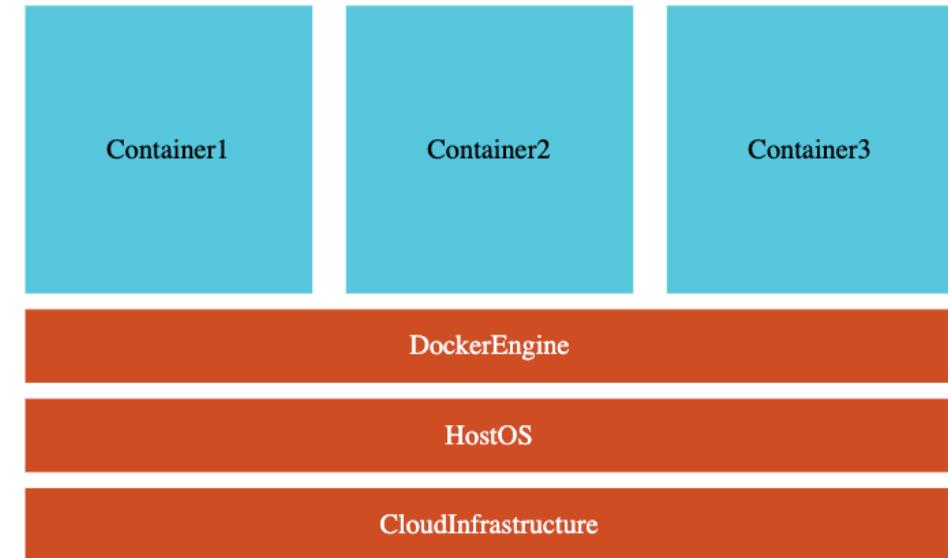
# Docker

- Create Docker images for each microservice
- Docker image **contains everything a microservice needs** to run:
  - Application Runtime (JDK or Python or NodeJS)
  - Application code
  - Dependencies
- You can run these docker containers **the same way** on any infrastructure
  - Your local machine
  - Corporate data center
  - Cloud



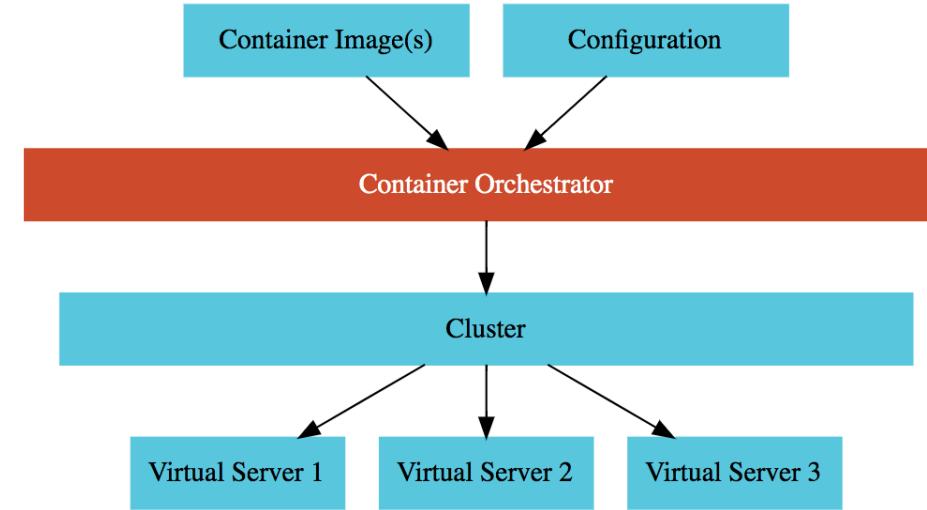
# Docker - Advantages

- Docker containers are **light weight** (compared to Virtual Machines)
- Docker provides **isolation** for containers
- Docker is **cloud neutral**
- (NEW CHALLENGE) How do you manage 1000's of containers belonging to multiple microservices?
  - Enter Container Orchestration!



# Container Orchestration

- **Requirement :** I want 10 instances of Microservice A container, 15 instances of Microservice B container and ....
- **Typical Features:**
  - **Auto Scaling** - Scale containers based on demand
  - **Service Discovery** - Help microservices find one another
  - **Load Balancer** - Distribute load among multiple instances of a microservice
  - **Self Healing** - Do health checks and replace failing instances
  - **Zero Downtime Deployments** - Release new versions without downtime



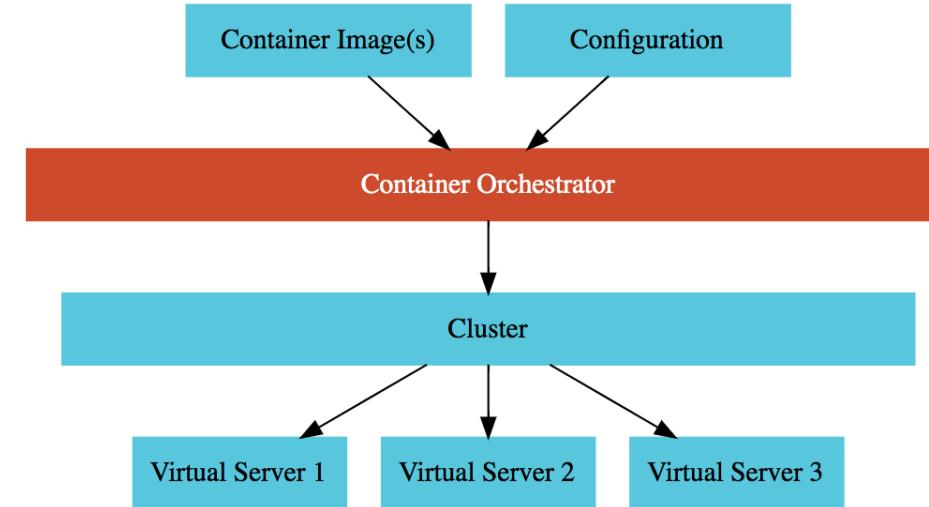
# Container Orchestration Options

- **Cloud Neutral**

- Kubernetes
- AWS service - AWS Elastic Kubernetes Service (EKS)
- EKS does not have a free tier

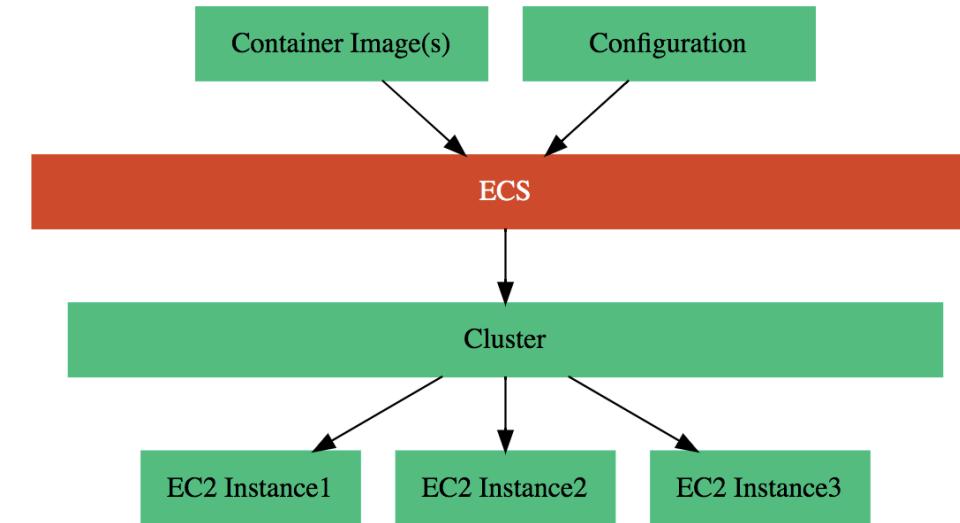
- **AWS Specific**

- AWS Elastic Container Service (ECS)
- AWS Fargate : Serverless version of AWS ECS
- AWS Fargate does not have a free tier



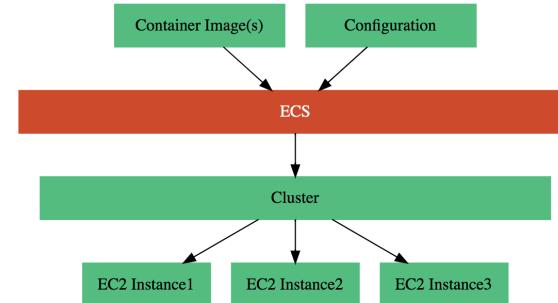
# Amazon Elastic Container Service (Amazon ECS)

- Fully managed service for container orchestration
- Serverless option - AWS Fargate
- Use cases:
  - Microservices Architectures - Create containers for your microservices and orchestrate them using ECS or Fargate
  - Batch Processing. Run batch workloads on ECS using AWS Batch
- DEMO



# Amazon ECS - Cluster

- **Cluster** : Group of one or more EC2 instances
  - AWS Fargate = serverless. DON'T worry about EC2 instances.
- **Container Instance** = EC2 instance + **container agent**
  - Use ECS ready AMIs
  - Communicates with the ECS cluster
    - Define Cluster Info in (`/etc/ecs/ecs.config`)
  - Use On-Demand instances or Spot instances
- AWS Fargate does NOT give you visibility into the EC2 instances in the cluster.
- Amazon ECS capacity providers can be used to implement cluster auto scaling
  - Scale out/Scale in EC2 instances based on load



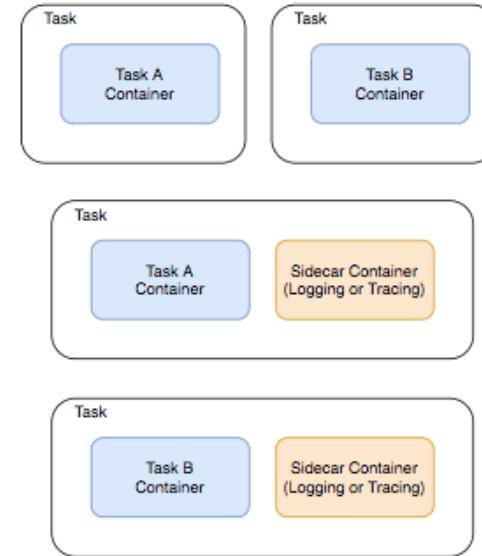
# Amazon ECS - Task Definition Example

In 28  
Minutes

```
{  
    "containerDefinitions": [  
        {  
            "portMappings": [  
                {  
                    "hostPort": 80,  
                    "protocol": "tcp",  
                    "containerPort": 80  
                }  
            ],  
            "cpu": 10,  
            "memory": 300,  
            "image": "httpd:2.4",  
            "name": "simple-app"  
        }  
    ]  
}
```

# Amazon ECS - Task Definition

- **Important Configuration:**
  - Which Docker image is used to create your containers?
  - CPU and memory at task and/or container level
  - **Launch type**
    - EC2 or FARGATE (Which kind of cluster do you want to run this task on?)
  - Logging configuration
  - Data volumes attached to containers
  - Task Permissions
- (Remember) When should you put two containers in same task-definition?
  - Scenario 1 : Common lifecycle with shared data volumes
  - Scenario 2 : Sidecar pattern
    - Deploy a container along side every microservice container to proxy requests and manage metrics and logs

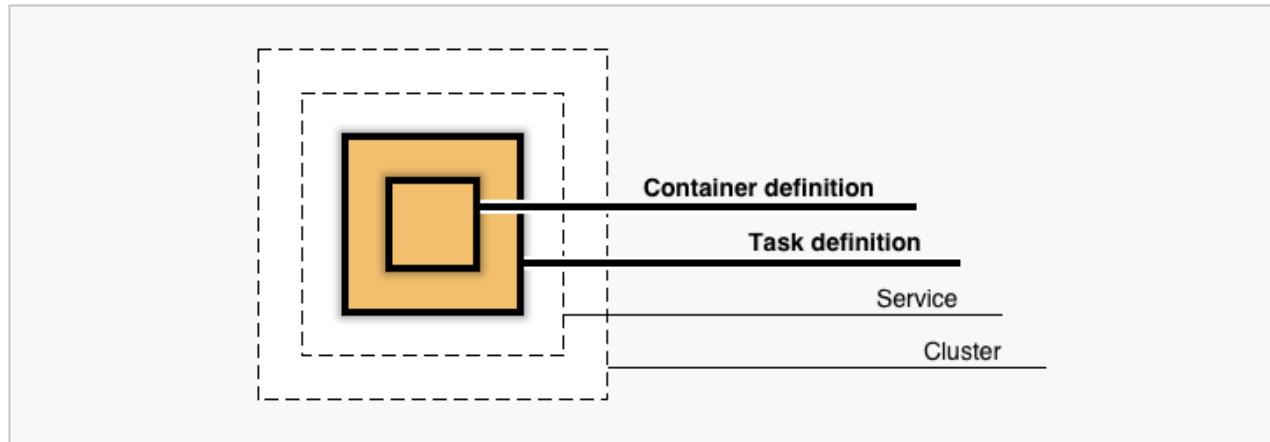


# Amazon ECS - Task Permissions

- **Task IAM Role**
  - Define an IAM role to use in your task definition
  - Specific permissions for your task/application
  - Do you need to talk with a database?
  - (EC2 container agent) Enable it using `ECS_ENABLE_TASK_IAM_ROLE=true`
  - (ADVANTAGE) More secure than using an EC2 instance's role
  - (BEST PRACTISE) 10 Microservices => 10 Task Definitions => 10 Task IAM Roles with individual permissions needed by each microservice
- **Task Execution IAM Role**
  - Provide Amazon ECS container and Fargate agents access to:
    - Pull container images from ECR
    - Publish container logs to CloudWatch

# Amazon ECS - Service

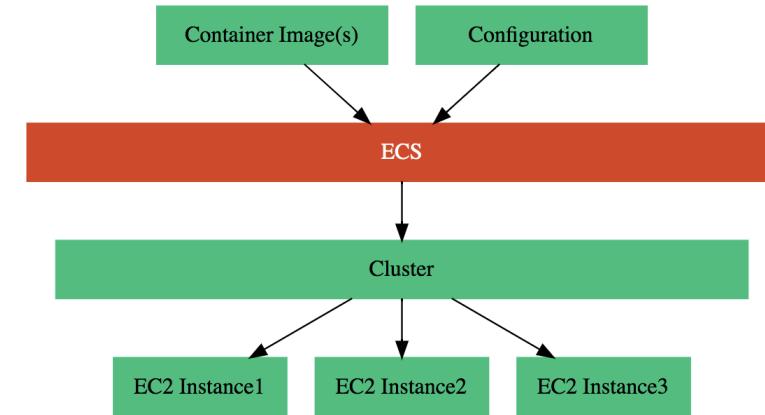
In 28  
Minutes



- **Maintain** specified number ("desired count") of tasks
- **Important Configuration:**
  - Deployment type
    - Rolling update
    - Blue/green deployment (powered by AWS CodeDeploy)
  - Task Placement
  - Task Auto Scaling

# Amazon ECS - Task Placement

- Identify the instances that satisfy
  - CPU, memory, and port requirements
    - From Task Definition
  - Task placement constraints
  - Task placement strategies
- Select the instances for task placement



# Amazon ECS - Strategy types

```
"placementStrategy": [  
    {  
        "field": "attribute:ecs.availability-zone",  
        "type": "spread"  
    },  
    {  
        "field": "memory",  
        "type": "binpack"  
    }  
]
```

- **binpack** - Leave least amount of unused CPU or memory
  - (REMEMBER) Minimizes number of container instances in use
- **random** - Random task placement
- **spread** - Spread evenly based on the specified values. Examples include:
  - Host (instanceId)
  - (OR) Availability Zone(attribute:ecs.availability-zone)
- (ALSO ALLOWED) Combine strategies and prioritize

# Amazon ECS - Task Placement Constraints

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]

//OR

"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type = t2.micro",
        "type": "memberOf"
    }
]
```

- `distinctInstance` - Place each task on a different container instance
- `memberOf` - Place tasks on container instances
  - Use Cluster query language to group objects and define constraints
    - `attribute:ecs.instance-type == t2.micro`
    - `attribute:ecs.availability-zone in [us-east-1c, us-east-1d]`
    - `ec2InstanceId in ['i-abcd1234' 'i-wxvxx7890']`

# Amazon Elastic Container Service - Remember

In 28  
Minutes



ECS



ELB

- Load balancing performed using Application Load Balancers
- Two features of ALB are important for ECS:
  - **Dynamic host port mapping:** Multiple tasks from the same service are allowed per EC2 (container) instance
  - **Path-based routing:** Multiple services can use the same listener port on same ALB and be routed based on path ([www.app.com/microservice-a](http://www.app.com/microservice-a) and [www.app.com/microservice-b](http://www.app.com/microservice-b))
- Delete the cluster!

# AWS Elastic Beanstalk - Docker containers

- Deploy web applications using **Docker containers**
- (**MORE FLEXIBILITY**) Create your own runtime environment or customize an existing platform
- Use images from Docker repositories or Build container images during deployment (include a *Dockerfile*)
  - Supports Docker Compose (*docker-compose.yml*)
  - Also provides custom structure to your define containers (*Dockerrun.aws.json*)
- **Two Options:**
  - Single Container
  - Multi Container (Uses ECS in the background!)

# Running Docker Containers in AWS

- **Elastic Beanstalk**
  - Single container or multiple containers in same EC2 instance
  - Recommended for simple web applications
- **Amazon ECS**
  - AWS specific solution for container orchestration
  - Ideal for microservices
- **Amazon Fargate**
  - Serverless version of Amazon ECS
  - You want to run microservices and you don't want to manage the cluster
- **Amazon EKS**
  - AWS managed service for Kubernetes
  - Recommended if you are already using Kubernetes and would want to move the workload to AWS

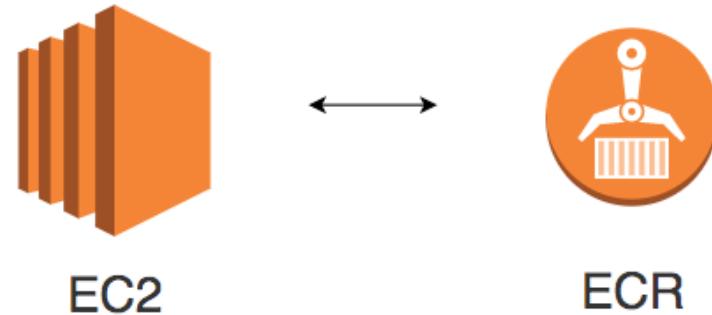
# Amazon ECR (Elastic Container Registry)



- You've created docker images for your microservices:
  - Where do you store them?
- You need a **Container Registry**
- **Amazon ECR** is a Fully-managed Docker container registry provided by AWS
- (Alternative) Docker Hub

# Docker Commands - Quick Reference

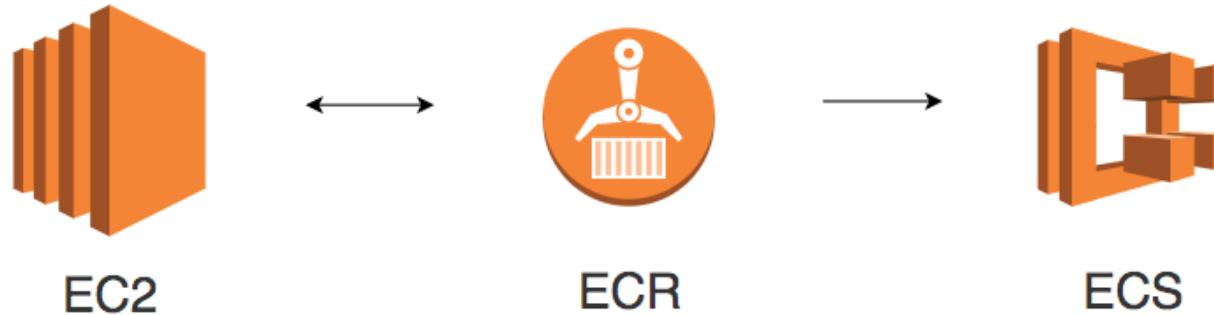
In 28  
Minutes



- `docker build`: Build a docker image for the microservice
- `docker push`: Push the docker image to a container repository
- `docker pull`: Pull an docker image from a container repository to your local machine
- Recommended Watch : <https://www.youtube.com/watch?v=Rt5G5Gj7RP0>

# Docker with ECR

In 28  
Minutes

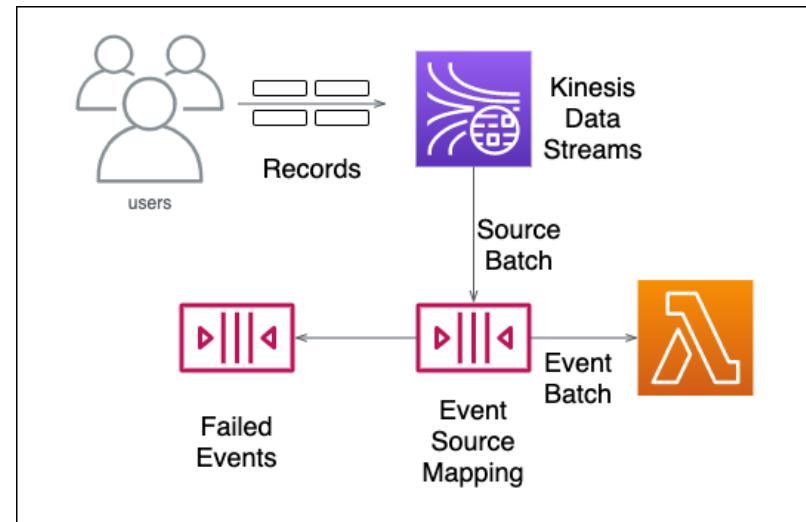


- To pull and push docker images from ECR, you need to login:
  - (DEPRECATED) `aws ecr get-login --region $AWS_DEFAULT_REGION`
  - (REMEMBER) This command returns the command to execute to be able to login to ECR:
    - Sample Output: `docker login -u AWS -p -e none https://dkr.ecr.amazonaws.com`
    - You need execute the output to Login to ECR
  - (NEW VERSION) `aws ecr get-login-password --region $AWS_DEFAULT_REGION`

# More Serverless

# AWS Lambda - Event Source Mapping

- Some AWS services don't invoke Lambda functions directly
  - Example: Events from DynamoDB, Kinesis and SQS
- Event Source Mapping is a Lambda resource:
  - Read from event source & invoke a Lambda function
  - Reads records in batches (shards)
  - Automatic retries in case of failure (ensure in-order processing)
  - On repeated failures, you can send batch details to SQS queue or SNS topic
- (REMEMBER) Services like Amazon S3 and SNS do NOT use Event Source Mapping
  - Configuration is made in Amazon S3 and SNS (NOT on Lambda)



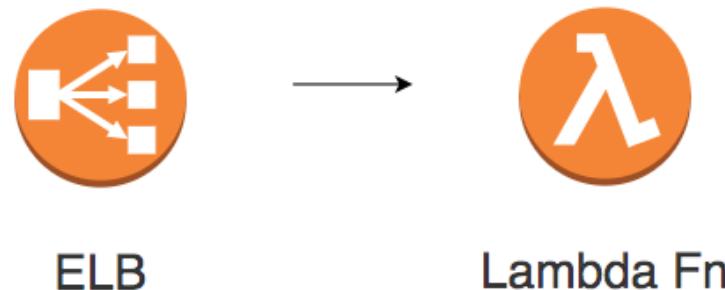
# AWS Lambda Event Source Mapping Example

```
// aws lambda list-event-source-mappings
{
  "EventSourceMappings": [
    {
      "UUID": "20aa21fa-5528-4e87-8176-1e9432f11669",
      "StateTransitionReason": "User action",
      "LastModified": 1602464340.0,
      "BatchSize": 1,
      "State": "Enabled",
      "FunctionArn": "arn:aws:lambda:us-east-1:183686979203:function:ProcessDynamodbStream",
      "EventSourceArn": "arn:aws:dynamodb:us-east-1:183686979203:table/my-todos/stream",
      "LastProcessingResult": "OK"
    }
  ]
}
```

- Stream `my-todos/stream` belonging to a DynamoDB table will be handled by lambda function `ProcessDynamodbStream`

# AWS Lambda & Application Load Balancers

In 28  
Minutes



- Lambda function can be configured as a target for an ALB
- Use ALB rules to route HTTP requests to Lambda function
  - ALB makes a synchronous call
- When an ALB receives an HTTP request, it converts HTTP request to an event
- Grant ALB permission to run the function
  - Add ALB to Lambda function's resource based policy
    - "Principal": {"Service": "elasticloadbalancing.amazonaws.com"}
    - "Action": "lambda:InvokeFunction"
    - "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"

# AWS Lambda with ALB - Request Event Example

```
{  
    "requestContext": {  
        "elb": {  
            "targetGroupArn": "arn:aws:elasticloadbalancing:LAMBDA_TARGET_GROUP"  
        }  
    },  
    "httpMethod": "GET",  
    "path": "/lambda",  
    "queryStringParameters": {  
        "query": "1234ABCD"  
    },  
    "headers": {  
        "accept": "text/html,....",  
        "accept-encoding": "gzip",  
        "host": "lambda-alb-123578498.us-east-2.elb.amazonaws.com",  
        ....  
        "x-forwarded-for": "72.12.164.125",  
        "x-forwarded-port": "80",  
        "x-forwarded-proto": "http",  
        "x-imforwards": "20"  
    },  
    "body": "",  
    "isBase64Encoded": false  
}
```

# AWS Lambda with ALB - Response Document Example

```
{  
    "statusCode": 200,  
    "statusDescription": "200 OK",  
    "isBase64Encoded": False,  
    "headers": {  
        "Content-Type": "text/html"  
    },  
    "body": "<h1>Hello from Lambda!</h1>"  
}
```

# AWS Lambda - Permissions

- **Lambda execution role** - Grants permissions to access AWS services and resources
  - You can assign a role when creating a function
  - Function assumes this role when invoked
- Predefined policies simplify permission configuration:
  - **AWSLambdaBasicExecutionRole** – Permission to upload logs to CloudWatch.
  - **AWSLambdaDynamoDBExecutionRole** – Permission to read records from an Amazon DynamoDB stream.
  - **AWSLambdaSQSQueueExecutionRole** – Permission to read a message from an Amazon Simple Queue Service (Amazon SQS) queue.
  - **AWSLambdaVPCAccessExecutionRole** – Permission to manage elastic network interfaces to connect your function to a VPC.
  - **AWSXRayDaemonWriteAccess** – Permission to upload trace data to X-Ray.



AWS Lambda

# AWS Lambda Resource Based Policies

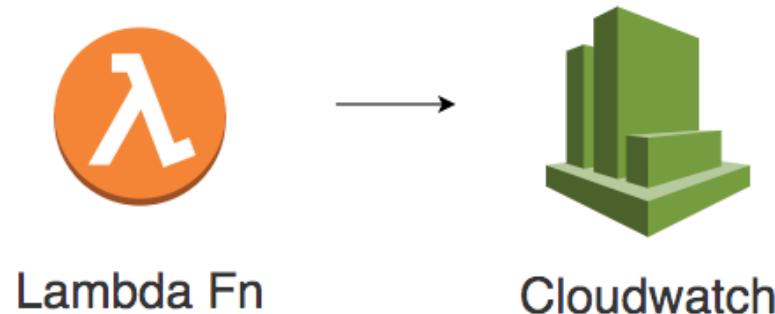
Resource-based policy [Info](#)

```
1 {  
2   "Version": "2012-10-17",  
3   "Id": "default",  
4   "Statement": [  
5     {  
6       "Sid": "lambda-d49ff629-xmpl-454d-8473-04c11fdc424c",  
7       "Effect": "Allow",  
8       "Principal": {  
9         "Service": "s3.amazonaws.com"  
10      },  
11      "Action": "lambda:InvokeFunction",  
12      "Resource": "arn:aws:lambda:us-east-2:123456789012:function:my-function",  
13      "Condition": {  
14        "StringEquals": {  
15          "AWS:SourceAccount": "123456789012"  
16        },  
17        "ArnLike": {  
18          "AWS:SourceArn": "arn:aws:s3:::my-bucket"  
19        }  
20      }  
21    }  
22  ]  
23 }
```

- Allow other accounts or specific AWS services to invoke a Lambda function
- Apply to a single function, version, alias, or layer version

# AWS Lambda Logging

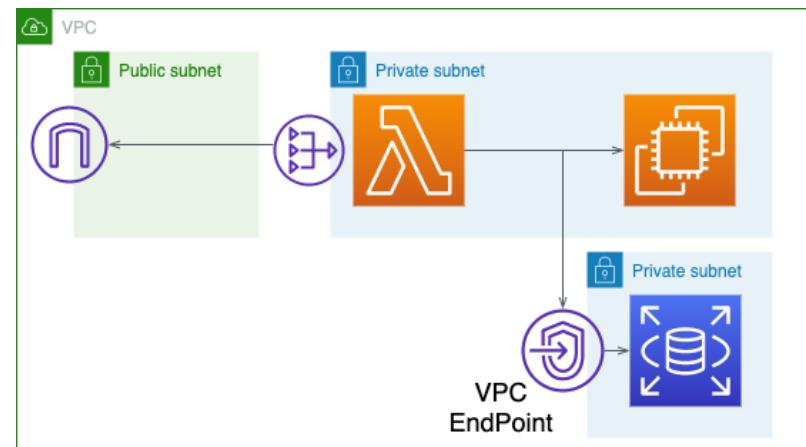
In 28  
Minutes



- Lambda logs are automatically stored in CloudWatch Logs
  - Default log group name: /aws/lambda/function-name
- Helps with Auditing and Troubleshooting
  - Insert additional logs in function code
- If logs are not visible:
  - Check if Lambda Function has permissions to write to CloudWatch Logs(Execution role)

# AWS Lambda inside a VPC

- (Default) Lambda function runs outside a VPC
  - Has access to internet
  - Can't access private resources inside VPC (RDS, EC2..)
- How to provide access to resources in a VPC?
  - Configure Lambda to run inside VPC
  - Lambda creates Elastic Network Interface
    - Lambda execution role should have permissions for
      - ec2:CreateNetworkInterface
      - ec2:DescribeNetworkInterfaces
      - ec2:DeleteNetworkInterface
    - Use **AWSLambdaVPCAccessExecutionRole** managed policy
- How do you provide a Lambda function running inside a VPC access to internet?
  - Ensure that the private subnet has a route to a NAT Gateway



# API Gateway - CORS Configuration

- To enable CORS, REST API resource needs to implement an OPTIONS method returning these headers:
  - Access-Control-Allow-Methods
  - Access-Control-Allow-Headers
  - Access-Control-Allow-Origin
- How to configure CORS?
  - **REST API Lambda custom (non-proxy) integration** - Enable CORS and configure API Gateway method response and integration response settings
  - **REST API Lambda proxy integrations** - Implement logic in lambda function (details on next slide) to return headers (in addition to settings similar to custom integration)
  - **HTTP API** - Enable CORS and configure properties (allowOrigins, allowMethods, allowHeaders)



API Gateway

# Lambda proxy integrations - Enabling CORS

```
exports.handler = async (event) => {
  const response = {
    statusCode: 200,
    headers: {
      "Access-Control-Allow-Headers" : "Content-Type",
      "Access-Control-Allow-Origin": "https://www.yourwebsite.com",
      "Access-Control-Allow-Methods": "OPTIONS,POST,GET,PUT,DELETE"
    },
    body: JSON.stringify('Your Lambda Function'),
  };
  return response;
};
```

# API Gateway - REST API vs HTTP API

Feature	HTTP API	REST API
Authorizers	AWS Lambda, IAM, Amazon Cognito, OAuth 2.0/OIDC	AWS Lambda, IAM, Amazon Cognito
Integrations	HTTP, Lambda, AWS services, Private integration	HTTP, Lambda, AWS services, Private integration, Mock
Usage plans/API keys	X	✓
API caching	X	✓
API type	Regional	Regional/Edge-optimized/Private

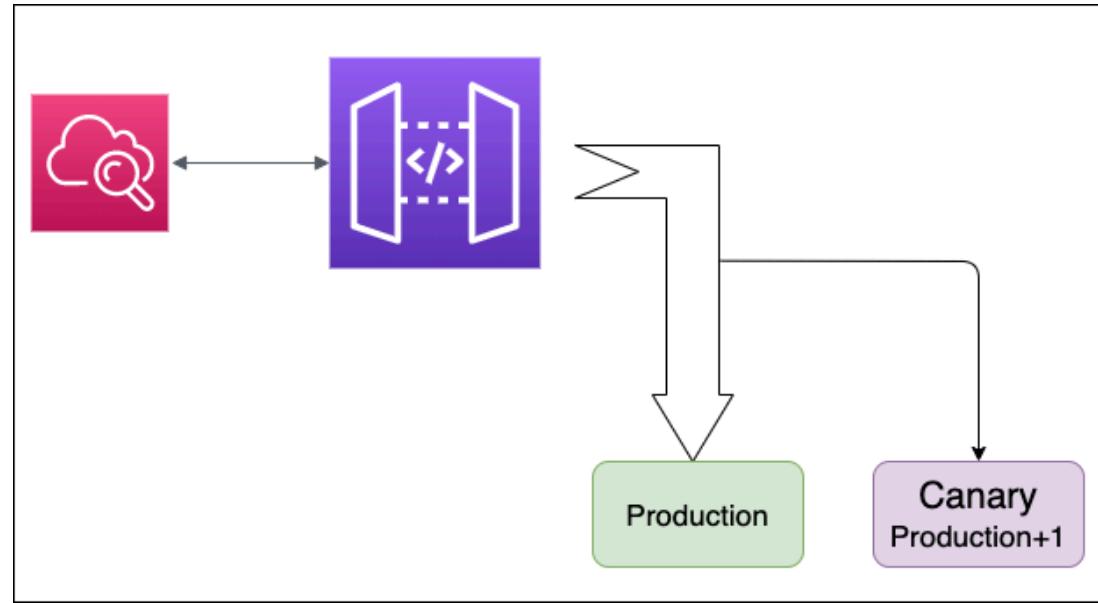
# API Gateway - REST API vs HTTP API - 2

In 28  
Minutes

Feature	HTTP API	REST API
Request transformation	X	✓
Request / response validation	X	✓
Test invocation	X	✓
Automatic deployments	✓	X
Default stage	✓	X
AWS X-Ray	X	✓

# API Gateway - Canary Releases

In 28  
Minutes



- New version of the software is deployed for testing purposes while base version still in production
- Small % of traffic routed to new version to see the behavior in production
- New version is either promoted or rolled back based on the behavior

# Creating a Canary Release

- Step 1 : Create a Canary
  - Configure
    - Percentage of requests to send to Canary
    - Configure Canary Stage Variables (override existing stage variables or add new stage variables for the canary release)
- Step 2 : Deploy new release to Canary
  - Choose Deploy API and choose the stage where canary is enabled
  - Test the Canary release
- Step 3 : Promote Canary to 100%
  - Choose "Promote Canary"
  - Options:
    - Update stage with Canary's deployment
    - Update stage with Canary's stage variables
    - Set Canary percentage to 0.0%

Manage Canary settings here. A Canary is used to test new API deployments and/or changes to stage variables. A Canary can receive a percentage of requests going to your stage. In addition, API deployments will be made to the Canary first before being able to be promoted to the entire stage.

Promote Canary   Delete Canary

Stage's Request Distribution

Percentage of requests directed to Canary 0%

Percentage of requests directed to test 100%

Canary Deployment

Deployment date Oct 14, 2020 10:10:06 AM

Description No description.

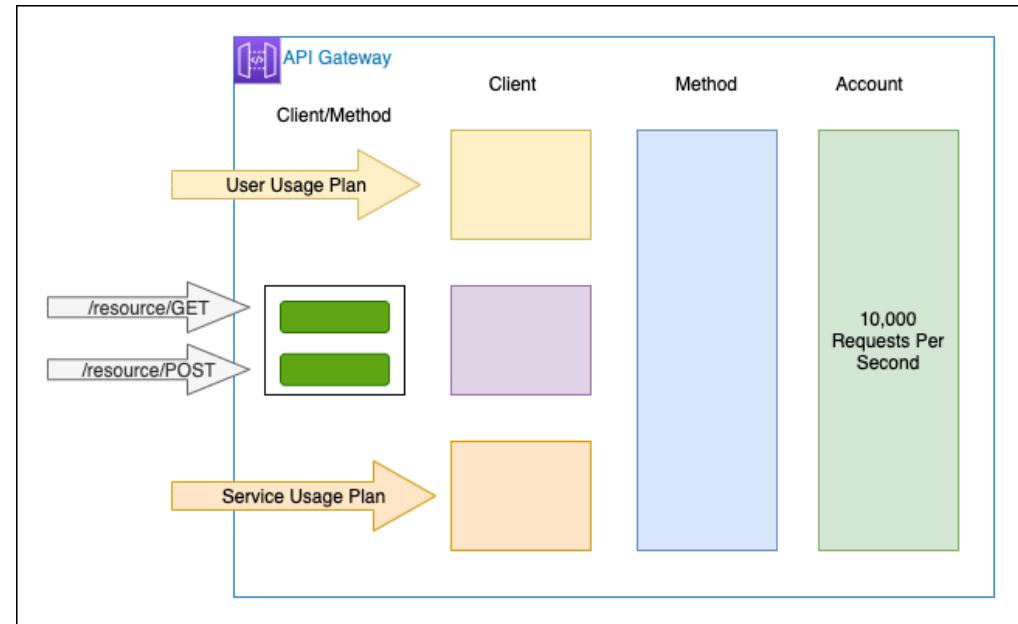
Canary Stage Variables

By default, your Canary inherits stage variables from the stage. You can override these stage variables or add new ones. When promoting a Canary's settings to the stage, the stage is able to update its stage variables to reflect any overridden values and include any new stage variables created by the Canary.

Name	Stage Value	Canary Override Value
------	-------------	-----------------------

# API Gateway - Throttling

- How to prevent clients from sending too many requests to your API Gateway?
  - Set request limits for steady-state and bursts (maximum number of concurrent requests)
    - In case the limits are exceeded, API Gateway sends 429 Too Many Requests error
- Throttling Levels:
  - **Account-level:** 10000 requests per second with a burst of 5000 requests
  - **Method-level:** Configure throttling limits at the level of a resource method
  - **Client-level:** Create client specific keys and usage plans



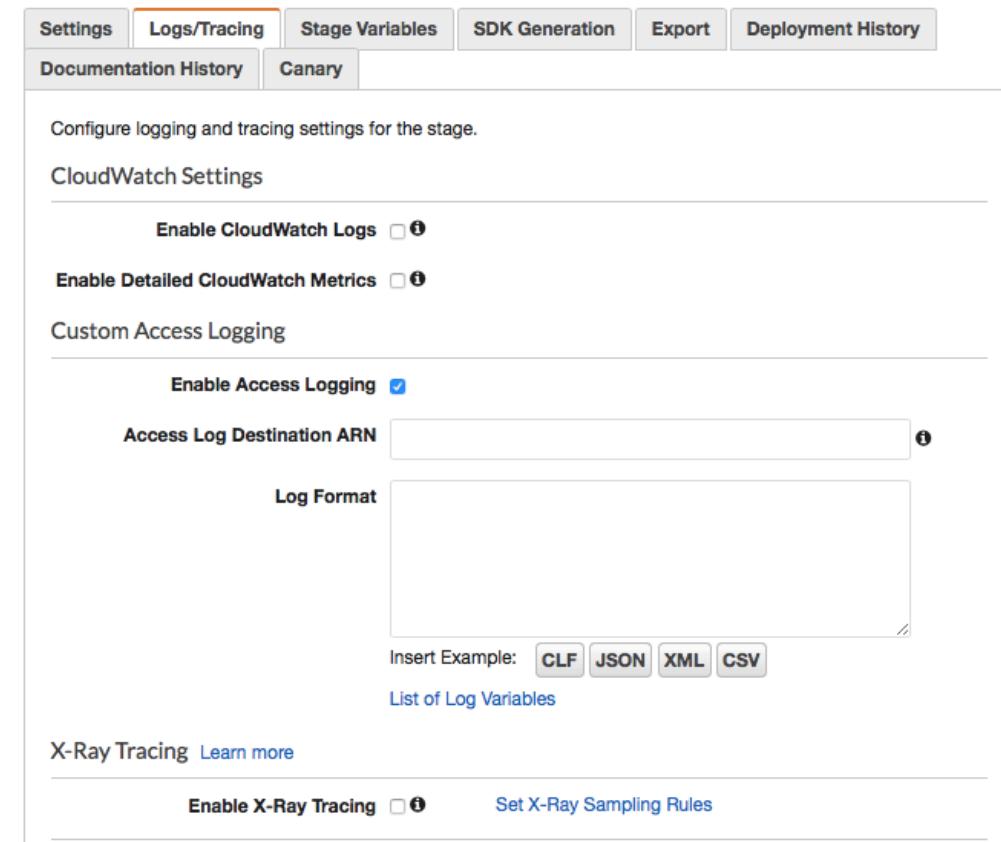
# Control Access - Resource policies

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "execute-api:Invoke",  
      "Resource": "arn:aws:execute-api:region:account-id:*",  
      "Condition": {  
        "NotIpAddress": {  
          "aws:SourceIp": "123.4.5.6/24"  
        }  
      }  
    }  
  ]  
}
```

- Control access to invoke your APIs:
  - Restrict by principal (typically an IAM user or role), Source IP CIDR blocks, VPCs or VPC end points
  - Give access to other AWS Accounts

# API Gateway - Monitoring

- **Amazon CloudWatch metrics** -  
Collects near-real-time metrics
  - Examples: 4XXError (client-side errors), 5XXError(server-side errors), CacheHitCount
- **Amazon CloudWatch Logs** - Debug issues related to request execution
- **AWS CloudTrail** - Record of actions taken by a user, role, or an AWS service in API Gateway
- **AWS X-Ray** - Trace your request across different AWS Services



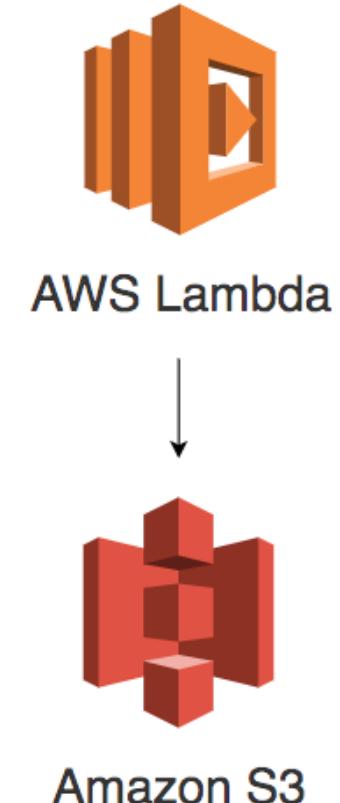
# Lambda CloudFormation - Inline

```
TestLambdaFunction:  
  Type: AWS::Lambda::Function  
  Properties:  
    Handler: lambda_function.lambda_handler  
    Role: !GetAtt LambdaExecutionRole.Arn  
    Code:  
      ZipFile: !Sub |  
              import json  
  
              def lambda_handler(event, context):  
                  print(event)  
                  return {  
                      "statusCode": 200,  
                      "body": "<h1>Hello from Lambda!</h1>"  
                  }  
  Runtime: nodejs8.10
```

- Simple Lambda functions can be defined inline in CloudFormation scripts!
- However most Lambda functions are more complex and depend on other packages!
  - (BEST PRACTICE) Create a deployment package!

# Lambda - Deployment Package

- ZIP archive - Compiled function code + Dependencies
  - (REMEMBER) One and Only One Zip for 1 Lambda function.
    - There CANNOT be a separate package for Dependencies
    - Dependencies should be included in the same package as code or use Layers
- (Option 1) Directly Update Lambda Function
  - Create a Zip with Lambda Code and Dependencies
  - `aws lambda update-function-code --function-name my-function --zip-file fileb://function.zip`
- (Option 2) Upload to Amazon S3 (>50 MB : S3 is mandatory)
  - Create a Zip with Lambda Code and Dependencies
  - Upload to S3
  - `aws lambda update-function-code --function-name my-function --s3-bucket BUCKET_NAME --s3-key OBJECT_KEY`



# Lambda CloudFormation - S3

```
TestLambdaFunction:  
  Type: AWS::Lambda::Function  
  Properties:  
    Handler: HelloWorld.lambda_handler  
    Role: !GetAtt LambdaExecutionRole.Arn  
    Code:  
      S3Bucket: lambda-s3-sample  
      S3Key: HelloWorld.zip  
      S3ObjectVersion: jUzGu2o2NjsILaVxIoY92D5zEUlvn2qx  
    Runtime: python3.6
```

- You can also use CloudFormation to deploy Lambda function code from an S3 Bucket
- (REMEMBER) Make sure that you update at least one of the three for every new version - S3Bucket, S3Key, S3ObjectVersion
  - Otherwise, newest version will NOT be picked in the deployment

# AWS Lambda quotas

In 28  
Minutes

Quota	Limit
<b>Function memory allocation</b>	128 MB to 3,008 MB, in 64 MB increments
<b>Function timeout</b>	900 seconds (15 minutes)
<b>Function environment variables</b>	4 KB
<b>Function layers</b>	5 layers
<b>Function burst concurrency</b>	500 - 3000 (varies per region)
<b>Deployment package size</b>	50 MB (zipped, for direct upload) 250 MB (unzipped, including layers) 3 MB (console editor)
<b>/tmp directory storage</b>	512 MB
<b>Execution processes/threads</b>	1,024

# SAM - Deployment with CodeDeploy

## DeploymentPreference:

Type: Canary10Percent10Minutes

## Alarms:

- !Ref AliasErrorMetricGreaterThanZeroAlarm
- !Ref LatestVersionErrorMetricGreaterThanZeroAlarm

## Hooks:

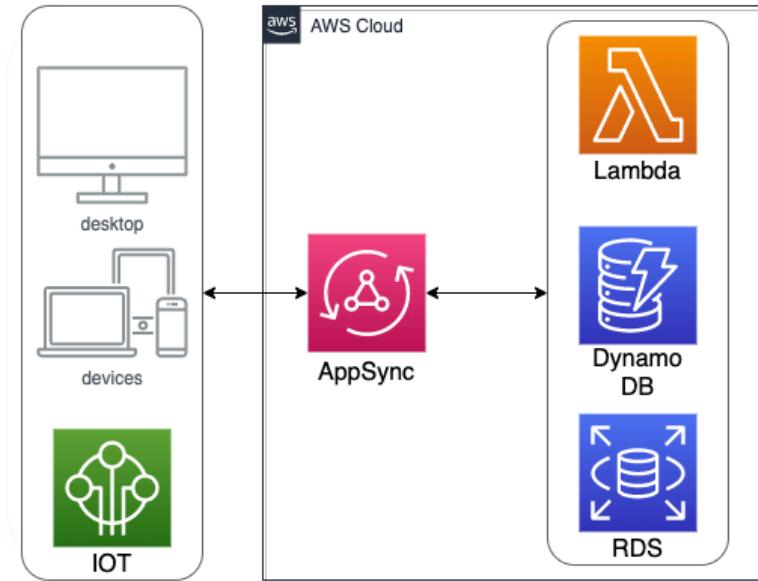
PreTraffic: !Ref PreTrafficLambdaFunction

PostTraffic: !Ref PostTrafficLambdaFunction

- Built-in support for CodeDeploy deployments
  - Canary (Canary10PercentXMinutes)
  - Linear (Linear10PercentEveryXMinutes)
  - All-at-once (AllAtOnce)
- Supports hooks for lambda functions to call before and after traffic shifting
- Rollback if any of the configured CloudWatch alarms are triggered

# AWS AppSync

- We are in multi device world
  - Want to synchronize app data across devices?
  - Want to create apps which work in off-line state?
  - Want to automatically sync data once user is back online?
- Welcome AWS AppSync
- Based on GraphQL
- App data can be accessed from anywhere
  - NoSQL data stores, RDS or Lambda
- (Alternative) Cognito Sync is limited to storing simple key-value pairs
  - AppSync recommended for almost all use cases



# AWS Step Functions

- Create a serverless workflow in 10 Minutes using a visual approach
- Orchestrate multiple AWS services into serverless workflows:
  - Invoke an AWS Lambda function
  - Run an Amazon Elastic Container Service or AWS Fargate task
  - Get an existing item from an Amazon DynamoDB table or put a new item into a DynamoDB table
  - Publish a message to an Amazon SNS topic
  - Send a message to an Amazon SQS queue
- Build workflows as a series of steps:
  - Output of one step flows as input into next step
  - Retry a step multiple times until it succeeds
  - Maximum duration of 1 year



# AWS Step Functions

In 28  
Minutes

- Integrates with Amazon API Gateway
  - Expose API around Step Functions
  - Include human approvals into workflows
- (Use case) Long-running workflows
  - Machine learning model training, report generation, and IT automation
- (Use case) Short duration workflows
  - IoT data ingestion, and streaming data processing
- (Benefits) Visual workflows with easy updates and less code
- (Alternative) Amazon Simple Workflow Service (SWF)
  - Complex orchestration code (external signals, launch child processes)
- Step Functions is recommended for all new workflows  
UNLESS you need to write complex code for orchestration



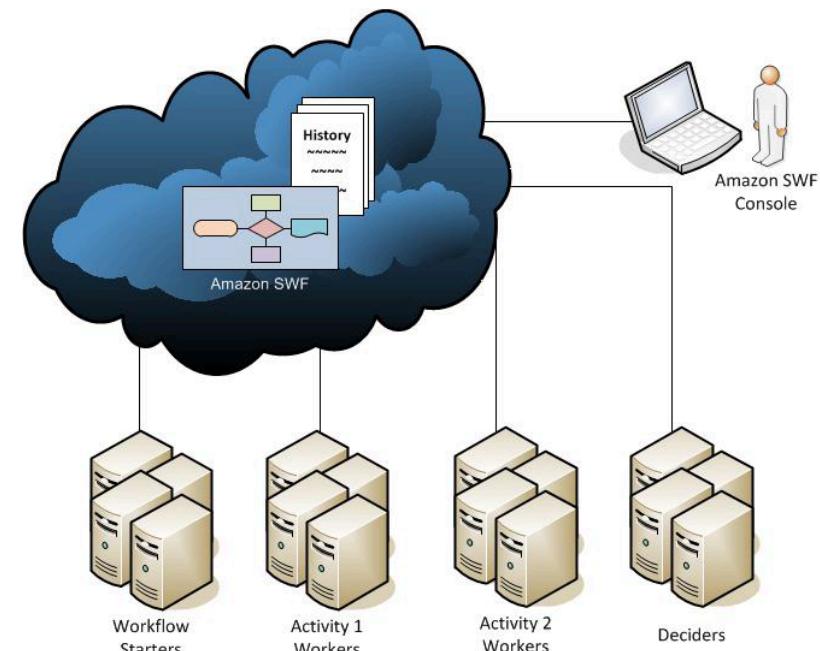
# Amazon Simple Workflow Service (SWF)

- Build and run background jobs with
  - parallel or sequential steps
  - synchronously or asynchronously
  - with human inputs (can indefinitely wait for human inputs)
- (Use cases) Order processing and video encoding workflows
- A workflow can start when receiving an order, receiving a request for a taxi
- Workflows can run upto 1 year
- Deciders and activity workers can use long polling



# Amazon SWF - Order Process

- Key Actors : Workflow starter, Decider and Activity worker
- Workflow starter calls SWF action to start workflow
  - Example: when an order is received
- SWF receives request and schedules a decider
  - Decider receives the task and returns decision to SWF:
    - For example, schedule an activity "Activity 1"
  - SWF schedules "Activity 1"
  - Activity worker performs "Activity 1". Returns result to SWF.
  - SWF updates workflow history. Schedules another decision task.
  - Loop continues until decider returns decision to close workflow
- SWF archives history and closes workflow

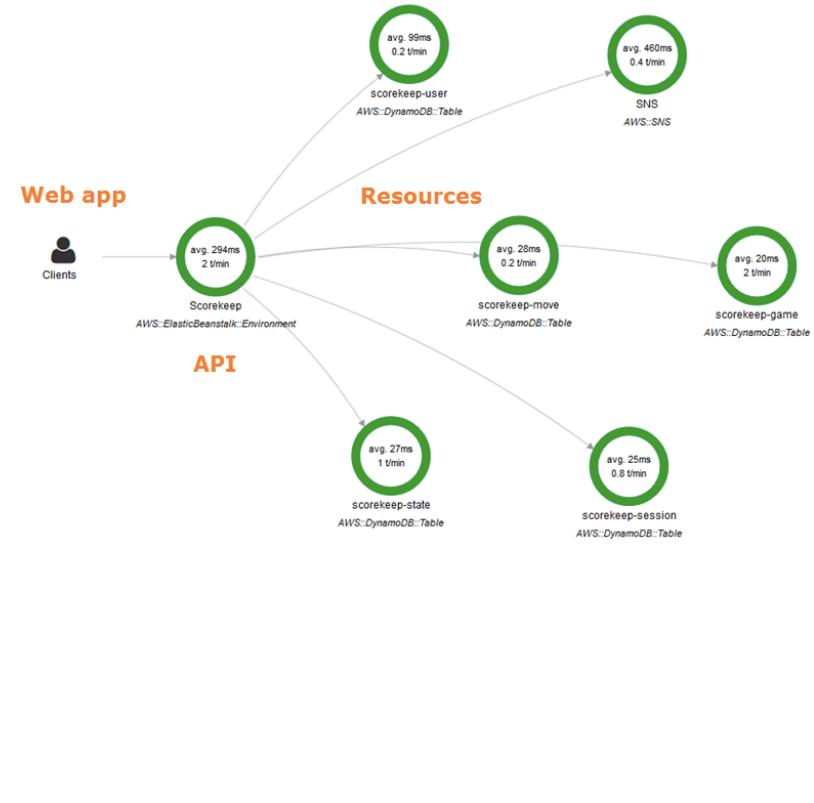


<https://docs.aws.amazon.com/amazonswf/latest/developerguide/swf-dev-actors.html>

# X-Ray

# X-Ray

- One request (especially in microservices architecture) might involve several Applications / Microservices / AWS services:
  - How to troubleshoot errors?
  - How to solve performance issues?
  - How to find out how something is working?
- What if you can trace a request across apps/components/AWS services?
- Enter X-Ray:
  - Gather tracing information
    - From applications/components
    - From AWS Services
  - Tools to view, filter and gain insights (Ex: Service Map)



# How does Tracing work?

- Unique trace ID assigned to every client request
  - X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
- Each service in request chain sends traces to X-Ray using this trace ID
- X-Ray gathers all the information and provides visualization
- Important Questions:
  - How do you reduce performance impact due to tracing?
    - Sampling - Only a sub set of requests are sampled (Configurable)
  - How can you make AWS Services and your applications send data to X-Ray?
    - Step I : Update Application Code Using X-Ray SDK
    - Step II: Use X-Ray agents (Some services make it EASY to use them! Example: AWS Lambda)



# X-Ray - Step 1 - Implement Tracing - X-Ray SDK

- Supports C#, Go, Java, Node.js, Python, Ruby
- **Interceptors** - Trace incoming HTTP requests
  - Example: Enable tracing using a filter in a Java application
- **Client handlers** - Instrument AWS SDK clients used to call other AWS services
  - Example: Enable tracing on calling to DynamoDB using DynamoDB SDK
- **HTTP client** - Instrument calls to other HTTP web services



AWS X-Ray

# X-Ray - Step 1 - Tracing Example - Using SDK

```
var app = express();

var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));

var rules = {
  "default": { "fixed_target": 1, "rate": 0.1 },
  "version": 1
}

AWSXRay.middleware.setSamplingRules(rules);

app.get('/', function (req, res) {
  res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

# X-Ray - Step 2 - Sending Traces - X-Ray Daemon

- Applications do NOT send details to X-Ray directly
  - Traces send to X-Ray Daemon (listens on UDP port 2000)
  - X-Ray Daemon gathers raw data and sends batches to X-Ray
- Using X-Ray Daemon:
  - **AWS Lambda & AWS Elastic Beanstalk:** Enable tracing and Ensure that execution role has permissions to send data to X-Ray
  - **EC2:** Install appropriate version (Download from S3) and Assign role with X-Ray permissions to EC2
  - **On-premises:** Install appropriate version (Download from S3) and create an IAM User with the permissions to send data to X-Ray
    - For example: Configure aws\_access\_key\_id and aws\_secret\_access\_key in `~/.aws/credentials`
  - **Amazon ECS:** Use the Daemon container image (`amazon/aws-xray-daemon`)
  - Minimum Permission Needed:
    - PutTraceSegments: Upload segment documents
      - `PUT /Segments/Records/UploadTraceSegments/Segments`



# X-Ray Trace Hierarchy: Trace > Segment > Sub Segment

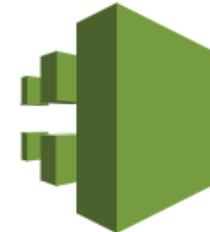
- **Trace** : Track the path of the request across Applications and AWS services (using a unique trace ID)
- **Segment** : All data points for a single component in the chain
  - Segment = System-defined and User-defined data (annotations) + Sub-Segments
- **Subsegment** - Granular details about remote calls made from a component (call to an AWS service, an external HTTP API, or an SQL database)
- **Annotations** - Key value pair with system or user-defined data
  - Searchable as they are indexed. Use them in filter expressions.
- **Metadata** - Key-value pairs with values of any type
  - Not indexed. Can NOT be searched.



# CloudTrail, Config & CloudWatch

# AWS CloudTrail

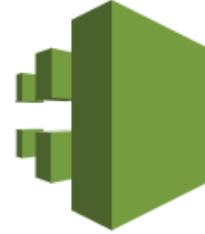
- Track events, API calls, changes made to your AWS resources:
  - Who made the request?
  - What action was performed?
  - What are the parameters used?
  - What was the end result?
- (USE CASE) Compliance with regulatory standards
- (USE CASE) Troubleshooting. Locate a missing resource
- Delivers log files to S3 and/or Amazon cloud watch logs log group ( S3 is default )
- You can setup SNS notifications for log file delivery



AWS CloudTrail

# AWS Cloud Trail Types

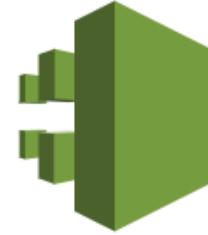
- Multi Region Trail
  - One trail of all AWS regions
  - Events from all regions can be sent to one CloudWatch logs log group
- Single Region Trail
  - Only events from one region
  - Destination S3 bucket can be in any region



AWS CloudTrail

# AWS Cloud Trail - Good to know

- Log files are automatically encrypted with Amazon S3 SSE
- You can configure S3 Lifecycle rules to archive or delete log files
- Supports log file integrity
  - You can prove that a log file has not been altered



AWS CloudTrail

- **Auditing**
  - Create a complete inventory of your AWS resources
- **Resource history and change tracking**
  - Find how a resource was configured at any point in time
  - Configuration of deleted resources would be maintained
  - Delivers history file to S3 bucket every 6 hours
  - Take configuration snapshots when needed
- **Governance**
  - Customize Config Rules for specific resources or for entire AWS account
  - Continuously evaluate compliance against desired configuration
  - Get a SNS notification for every configuration change
- **Consistent rules and compliance across AWS accounts:**
  - Group Config Rules and Remediation Actions into Conformance Packs



AWS Config

# Predefined Config Rule Examples (80+)

- **alb-http-to-https-redirection-check** - Checks whether HTTP to HTTPS redirection is configured on all HTTP listeners of Application Load Balancers
- **ebs-optimized-instance** - Checks whether EBS optimization is enabled for your EC2 instances that can be EBS-optimized
- **ec2-instance-no-public-ip** - Do EC2 instances have public IPs?
- **encrypted-volumes** - Are all EC2 instance attached EBS volumes encrypted?
- **eip-attached** - Are all Elastic IP addresses used?
- **restricted-ssh** - Checks whether security groups that are in use disallow unrestricted incoming SSH traffic



# AWS Config Rules

- (Feature) Create Lambda functions with your custom rules
- (Feature) You can setup auto remediation for each rule
  - Take immediate action on a non compliant resource
  - (Example) Stop EC2 instances without a specific tag!
- Enable AWS Config to use the rules
  - No Free Tier
  - More rules to check => More \$\$\$\$



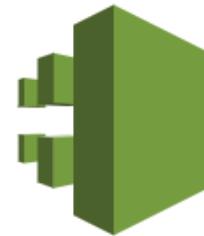
AWS Config

# AWS Config + AWS CloudTrail

In 28  
Minutes



AWS Config



AWS CloudTrail

- AWS Config
  - What did my AWS resource look like?
- AWS CloudTrail
  - Who made an API call to modify this resource?

# Amazon CloudWatch

# Monitoring AWS with Amazon CloudWatch

- Monitoring and observability service
- Collects monitoring and operational data in the form of logs, metrics, and events
- Set alarms, visualize logs, take automated actions and troubleshoot issues
- Integrates with more than 70 AWS services:
  - Amazon EC2
  - Amazon DynamoDB
  - Amazon S3
  - Amazon ECS
  - AWS Lambda
  - and ....



# Amazon CloudWatch Metrics

- How do you know if your application is working fine?
- How do you know if the AWS service is working efficiently?
- How do you know if you are making optimum use of your provisioned capacity?
- Gather metrics using **CloudWatch Metrics**:
  - Most AWS services provide free metrics for resources (Amazon EC2 instances, Amazon EBS volumes, and Amazon RDS DB instances)
  - Enable **detailed monitoring** if needed
  - Create your own application specific **custom metrics**
  - CloudWatch provides search, notification (alarm) and visualization around these metrics



Cloudwatch

# Amazon CloudWatch Metrics - Examples

- **ELB**
  - HTTPCode\_Target\_2XX\_Count, HTTPCode\_Target\_4XX\_Count, UnHealthyHostCount



ELB

- **DynamoDB**
  - AccountProvisionedReadCapacityUtilization, AccountProvisionedWriteCapacityUtilization, ConsumedReadCapacityUnits, ConsumedWriteCapacityUnits



DynamoDB

- **EC2**
  - CPUUtilization, NetworkIn, NetworkOut
  - StatusCheckFailed\_System (AWS related issues - typically Infrastructure Failures)
  - StatusCheckFailed\_Instance(Problem you should address - software/network related)



EC2

- **Lambda**
  - Throttles, Errors, ConcurrentExecutions, Duration



AWS Lambda

- **API Gateway**
  - 4XXError, 5XXError, CacheHitCount, CacheMissCount, Count
  - IntegrationLatency (How long did the backend take to process?)
  - Latency (How long did the total client request to API Gateway take?)



API Gateway

# Amazon CloudWatch Metrics - Terminology

Name	Unit	Timestamp	Value	Dimensions
Utilization	Percentage	2020-10-31T12:30:00Z	85	InstanceId=ec2-1234
Utilization	Percentage	2020-10-31T12:35:00Z	100	InstanceId=ec2-1234

- **Namespace** - Container for CloudWatch metrics.
  - Used to group metrics related to a Service or Application.
  - Group metrics from multiple applications under single namespace to create a multi application dashboard
- **Metric** - Time-ordered data point
- **Dimensions** - name/value pairs associated with a metric

# Amazon CloudWatch Custom Metrics

```
https://monitoring.&api-domain;/doc/2010-08-01/  
?Action=PutMetricData&Version=2010-08-01  
&Namespace=TestNamespace  
&MetricData.member.1.MetricName=buffers  
&MetricData.member.1.Unit=Bytes  
&MetricData.member.1.Value=231434333  
&MetricData.member.1.Dimensions.member.1.Name=InstanceID  
&MetricData.member.1.Dimensions.member.1.Value=i-aaba32d4  
&MetricData.member.1.Dimensions.member.2.Name=InstanceType  
&MetricData.member.1.Dimensions.member.2.Value=m1.small
```

- Publish your own metrics to CloudWatch:
  - Use CloudWatch **PutMetricData** API
    - Add data (MetricName, Unit, Value)
    - Add dimensions (InstanceId=1234, InstanceType=t2.micro)
  - Make sure that the IAM Role has permissions to call this API
- Metrics can use Two Resolutions
  - **Standard (Default)** - 1 minute granularity
  - **High Resolution** - 1 second granularity
    - Expensive (will involve more PutMetricData API calls)
    - Use Case: **High resolution alarms** - Quick alarms in 10 or 20 seconds

# Amazon CloudWatch Metrics - Good to know

- Metrics exists only in the region in which they are created.
- Metrics can't be deleted and expire after 15 months
- EC2 Metrics
  - (DEFAULT) EC2 instances collect metrics every 5 minutes.
    - You can increase it to every one minute
  - CloudWatch does **NOT** have access to **operating system metrics** like memory consumption
    - Install CloudWatch agent to gather metrics around memory



Cloudwatch

# Amazon CloudWatch Logs

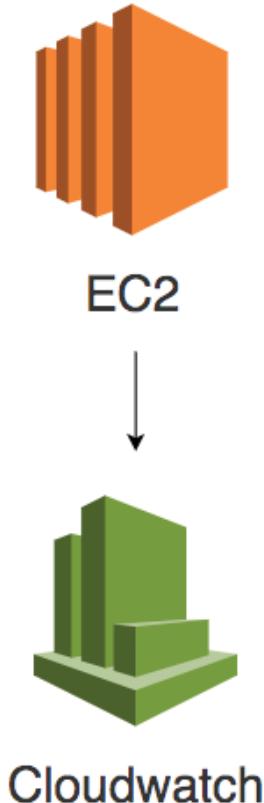
- Monitor and troubleshoot using system, application and custom log files
- **Real time application and system monitoring:**
  - Use **CloudWatch Logs Insights** to write queries and get actionable insights
    - Monitor for patterns in your logs and trigger alerts based on them
    - Example : Errors in a specific interval exceed a certain threshold
  - Use **CloudWatch Container Insights** to monitor, troubleshoot and set alarms for your containerized applications - EKS, ECS and Fargate
- **Long term log retention:**
  - Store logs in CloudWatch Logs for as long as you want
    - Default - forever. Configure expiry of your logs at log group level.
  - Or archive logs to S3 bucket (Typically involves a delay of 12 hours)
  - Or stream real time to Amazon Elasticsearch Service (Amazon ES) cluster using CloudWatch Logs subscription



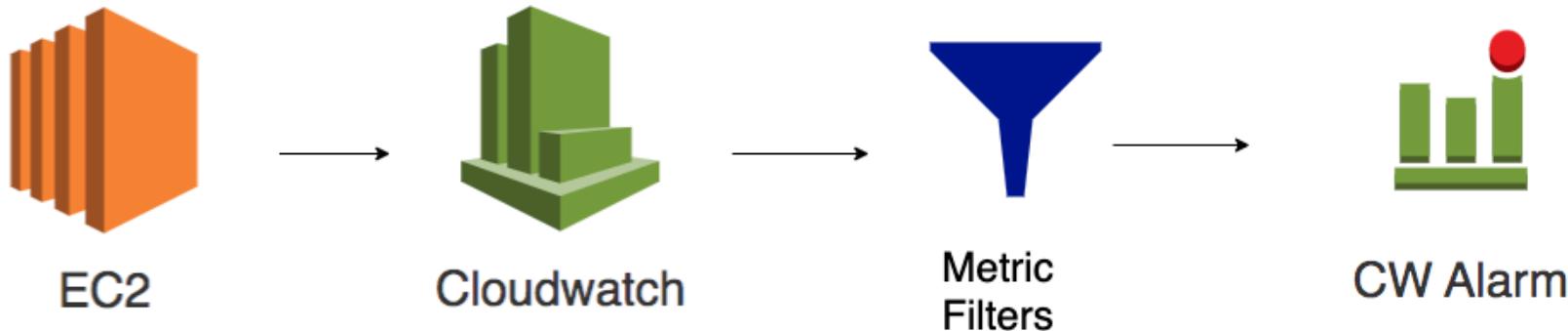
Cloudwatch

# CloudWatch Logs - Collect Logs from EC2/On-premise

- (Option 1) **Unified CloudWatch agent (NEW)**: ONE agent to collect logs and advanced metrics with **Multi OS support**
- (Option 2) **CloudWatch Logs agent (OLD)**: Limited to collection of logs from Linux based systems
- Give permissions to CloudWatch Agent to publish logs:
  - **EC2 Instances** - Attach the CloudWatchAgentServerRole IAM role to the EC2 instance
  - **On-Premise Instances** - Create an IAM User and configure a AmazonCloudWatchAgent profile (using aws\_access\_key\_id and aws\_secret\_access\_key)



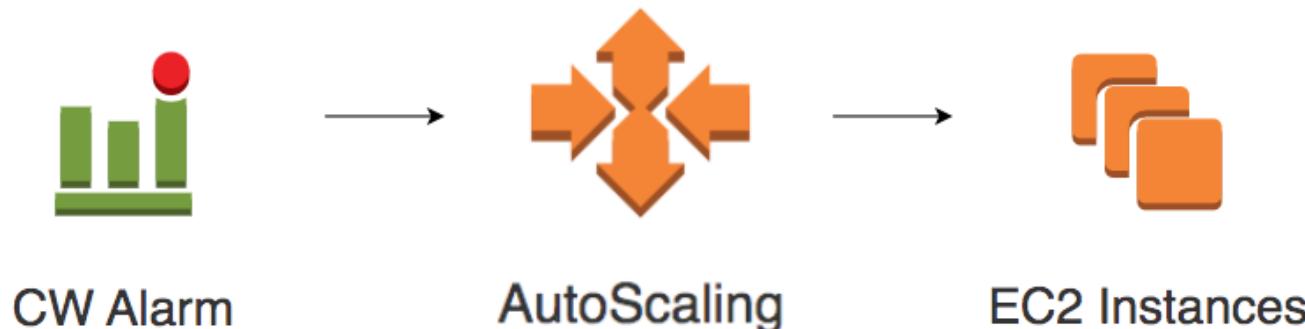
# Amazon CloudWatch Logs Metrics - Filters



- Get real time intelligence from your log files:
  - Your application logs application specific errors
  - You want to create a metric called **ErrorCount**
  - **Metric Filters** will help to achieve this
- Turn log data to CloudWatch metrics
- Retroactive filtering is NOT possible
  - Data captured only for future events (events after configuring filter)

# Amazon CloudWatch Alarms

In 28  
Minutes



- **Create alarms based on:**
  - Amazon EC2 instance CPU utilization
  - Amazon SQS queue length
  - Amazon DynamoDB table throughput or
  - Your own custom metrics
- **Take immediate action:**
  - Send a SNS event notification
    - Send an email using SNS
  - Execute an Auto Scaling policy

# Amazon CloudWatch Alarms - Terminology

- Amazon CloudWatch Alarm States:
  - OK - Metric is within the defined threshold
  - ALARM - Metric is outside the threshold causing an ALARM
  - INSUFFICIENT\_DATA - Not enough data available to determine the state
- Three things to configure an Alarm:
  - Period - Length of the time to evaluate the metric to create one data point
  - Evaluation Period - Number of most recent periods or datapoints to evaluate
  - Datapoints to Alarm - Number of datapoints in the Evaluation Period that should be breaching to cause an Alarm
- You set a CPU Utilization alarm on EC2 instance with a threshold of 80% over 3 periods of 10 minutes. If CPU utilization is 90% for 20 minutes, does the alarm get triggered?



CW Alarm

..

# Amazon CloudWatch Events

- Enable you to **take immediate action based on events on AWS resources**
  - Call a AWS Lambda function when an EC2 instance starts
  - Send event to an Amazon Kinesis stream when an Amazon EBS volume is created
  - Notify an Amazon SNS topic when an Auto Scaling event happens
- (ADDITIONAL FEATURE) **Schedule events** - Use Unix cron syntax
  - Schedule a call to Lambda function every hour or every minute
  - Send a notification to Amazon SNS topic every 3 hours



Cloudwatch

# CloudWatch Events Event Examples

- **Example Use Cases:**
  - Send an email after execution of every stage in a pipeline
  - Send an email if an EC2 instance is stopped
- **Example Events:**
  - CodeBuild (Build State-change)
  - CodeDeploy (Deployment State-change, Instance State-change)
  - CodeCommit (pullRequestCreated, pullRequestStatusChanged, commentOnCommitCreated etc)
  - CodePipeline (Pipeline Execution State Change, Stage Execution State Change)
  - Amazon EC2 State Change Events - stop, start, terminate



Cloudwatch

# Amazon EventBridge vs CloudWatch Events

- Original goal with CloudWatch Events was to help with monitoring usecases specific to AWS services.
- **Amazon EventBridge extends CloudWatch Events - Build event-driven architectures**
  - React to events from Your Applications, AWS services and Partner Services
    - Example: EC2 status change, change in your application or SaaS partner application
    - Structure of events is defined in Schema Registry (Code bindings are available)
  - **Event Targets** can be a Lambda function, an SNS Topic, an SQS queues etc
  - Rules map events to targets (Make sure that IAM Roles have permissions)
  - **Event buses** receive the events:
    - Default event bus (for AWS services), Custom event bus (custom applications), Partner event bus (partner applications)
- **ZERO change** needed for users of CloudWatch Events (Same URL)
- Over time, Amazon EventBridge will replace Amazon CloudWatch Events

# Event & Event pattern

```
//Event
{
    "version": "0","id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type": "EC2 Instance State-change Notification",
    "source": "aws.ec2",
    "account": "111122223333","time": "2017-12-22T18:43:48Z",
    "region": "us-west-1",
    "resources": ["arn:aws:ec2:us-west-1:123456789012:instance/ i-1234567890abcdef0"],
    "detail": {
        "instance-id": " i-1234567890abcdef0","state": "terminated"
    }
}

//Event Patterns
{
    "source": [ "aws.ec2" ],
    "detail-type": [ "EC2 Instance State-change Notification" ],
    "detail": {
        "state": [ "terminated" ]
    }
}
```

- Event patterns are used to identify events and route them to targets

# AWS CLI

# AWS CLI - Getting Started

In 28  
Minutes

```
aws ec2 describe-instances --output table --region us-east-1
```

- How do you run actions against your AWS services from the command line?
- How do you script action against AWS services?
  - Use AWS CLI
- Command Structure:
  - aws <command> <subcommand> [options/parameters]
  - aws <command> wait <subcommand> [options/parameters]
    - Waits until a command is complete
    - **aws deploy wait deployment-successful --deployment-id ABCDEFGH** : Wait for a deployment to be successful
  - Examples:
    - *aws s3 ls*
    - *aws ec2 describe-instances*

- Get Help:
  - `aws help` or `aws s3 help` or `aws ec2 describe-instances help`
- Important Options:
  - **--output**: Change format of output (json/yaml/text/table)
  - **--no-paginate**: Display only first page
    - (DEFAULT) CLI retrieves all pages at one time.
  - **--page-size**: Avoid "timed out" errors
    - Default Page Size is 1000. Using page-size option makes CLI request smaller number of items in each AWS API call.
  - **--debug**: Debug Mode
  - **--dry-run**: Checks whether you have the required permissions for the action, without actually making the request

# Using AWS CLI - Logging In

```
$ aws configure
AWS Access Key ID [None]: xxx
AWS Secret Access Key [None]: xxx
Default region name [None]: us-east-1
Default output format [None]: json

//CREDENTIALS FILE
[default]
aws_access_key_id=xxx
aws_secret_access_key=xxx
```

- **Access Keys** are used for programmatic login:
  - Command to use: **aws configure**
  - Creates a credential file:
    - Linux or macOS : `~/.aws/credentials`
    - Windows: `C:\Users\USERNAME.aws\credentials`
  - A config file is also created with default region and output format
- **(REMEMBER) DO NOT Use Access Keys on EC2 instances**
  - **(BEST PRACTICE)** Assign Roles to EC2 instances and use Instance Profiles.

# Using AWS CLI - Profiles

```
$ aws configure --profile produser
AWS Access Key ID [None]: yyy
AWS Secret Access Key [None]: yyy
Default region name [None]: us-east-1
Default output format [None]: text

$ aws s3 ls --profile produser
```

- How do you handle access to multiple aws accounts?
  - Use profiles
- Use `--profile` argument to call `aws configure` and other aws commands

# Using AWS CLI - Configuration settings & precedence

## 1. Command line options

- Use --region, --output, and --profile to override defaults)

## 2. Environment variables

- export (or setx) AWS\_ACCESS\_KEY\_ID
- Other examples: AWS\_SECRET\_ACCESS\_KEY, AWS\_DEFAULT\_REGION

## 3. CLI credentials file (Created when you use aws configure)

## 4. CLI configuration file (Created when you use aws configure)

## 5. Container credentials (from IAM role assigned to your Amazon ECS Tasks - Task Definitions)

## 6. Instance profile credentials (IAM roles assigned to your EC2 instances)

# Using an IAM role in the AWS CLI - On-Premise

```
[profile myrole]
role_arn = arn:aws:iam::XYZ:role/myrole
source_profile = your_user_profile
# credential_source = Ec2InstanceMetadata or EcsContainer or Environment

[your_user_profile]
aws_access_key_id=xxx
aws_secret_access_key=xxx
```

- Define a profile for the role in the `~/.aws/config` file
  - Create an IAM Role
    - Using `aws iam create-role` passing `--role-name` and the role policy (`--assume-role-policy-document`)
    - Or AWS Management Console

# Using an IAM role in the AWS CLI - 2

```
//Trust Policy
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::XYZ:user/my_user"},
>Action": "sts:AssumeRole"
"Condition": { "Bool": { "aws:multifactorAuthPresent": true } }

//IAM Policy
"Effect": "Allow",
>Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::XYZ:role/myrole"
```

- Execute a command using the profile configured with role
  - `aws s3 ls --profile myrole`
- When you use a profile :
  - CLI looks up the credentials for linked profile (your\_user\_profile) or credential\_source (IAM role attached to the instance profile or the container)
  - CLI uses the **sts:AssumeRole** operation
    - IAM user/role should have access to AssumeRole operation
    - Trust Policy associated with the IAM Role should allow IAM user to assume the role

# AWS Security Token Service (STS)

In 28  
Minutes

- Provide trusted users with **temporary security credentials** to access your AWS resources
- (ADVANTAGE) Tokens are **temporary!**
- Supports **identity federation**:
  - Web identity federation (OIDC) and
  - Corporate identity federation (SAML based)
- Provides a Global service (<https://sts.amazonaws.com>) from US East (N. Virginia) Region
- (RECOMMENDED) Use Regional AWS STS endpoints for Low Latency and Better Performance
  - Example: <https://sts.eu-west-1.amazonaws.com>

# AWS Security Token Service (STS) - APIs

- **AssumeRole** - Cross-account federation with a custom identity broker
- **AssumeRoleWithSAML** - Users authenticated with enterprise identity store
- **AssumeRoleWithWebIdentity** - For users authenticated with Amazon Cognito, Social, or any OpenID Connect-compatible identity provider
- **GetSessionToken** - Allows Same Account Access with MFA.
  - RECOMMENDED if MFA is needed for making API call
    - Example: Use MFA to protect programmatic calls to specific AWS API operations like Amazon EC2 StopInstances
- **DecodeAuthorizationMessage** - Decodes additional information about the authorization status of a request from encoded response
  - If your API call failed and you get an error - you can use DecodeAuthorizationMessage to debug
- **GetCallerIdentity** - Get IAM user or role whose credentials are used to call the operation

# AWS STS API - APIs - 2

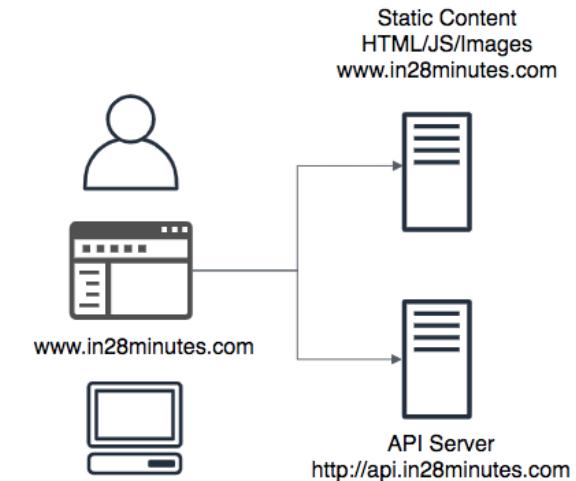
In 28  
Minutes

AWS STS API	Who can call	MFA	Def Expiry	Usecase
<b>AssumeRole</b>	IAM user or IAM role with existing temporary security credentials	Yes	1 hr	Cross-account delegation & federation (custom identity broker)
<b>AssumeRoleWithSAML</b>	Any user; Pass a SAML authentication response	No	1 hr	Federation (SAML identity provider)
<b>AssumeRoleWithWebIdentity</b>	Any user; Pass a web identity token	No	1 hr	Federation (web identity provider)
<b>GetFederationToken</b>	IAM user or AWS account root user	No	12 hrs	Federation (custom identity broker) Login to Console
<b>GetSessionToken</b>	IAM user or AWS account root user	Yes	12 hrs	Temporary credentials (users in untrusted environments)

# CORS, Configuration Management and Caching

# Cross-Origin Resource Sharing (CORS)

- How can you tell a browser that
  - Front end application running on one origin (<http://www.in28minutes.com>) can access resources (or APIs) from a different origin (<http://api.in28minutes.com>) ?
  - Cross-Origin Resource Sharing (CORS)
- Browsers send a preflight OPTIONS request:
  - Resources (APIs or Backend) should respond with headers:
    - Access-Control-Allow-Origin: <http://www.in28minutes.com>
    - Access-Control-Allow-Methods: POST, GET, OPTIONS
    - Access-Control-Allow-Headers: Authorization
  - (NOT RECOMMENDED) Using \* to allow everything:
    - Access-Control-Allow-Origin:\*
    - Access-Control-Allow-Methods:\*
    - Access-Control-Allow-Headers:\*



# Cross-Origin Resource Sharing (CORS) - S3

```
<CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>http://frontend.in28minutes.com</AllowedOrigin>

    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
</CORSConfiguration>
```

- Scenario:
  - Front-end web application hosted on a Amazon S3 bucket (front-end-bucket)
  - Images are hosted on a different S3 bucket (image-bucket)
- Configure CORS in S3:
  - Make CORS configuration on image-bucket allowing access from front-end-bucket URL

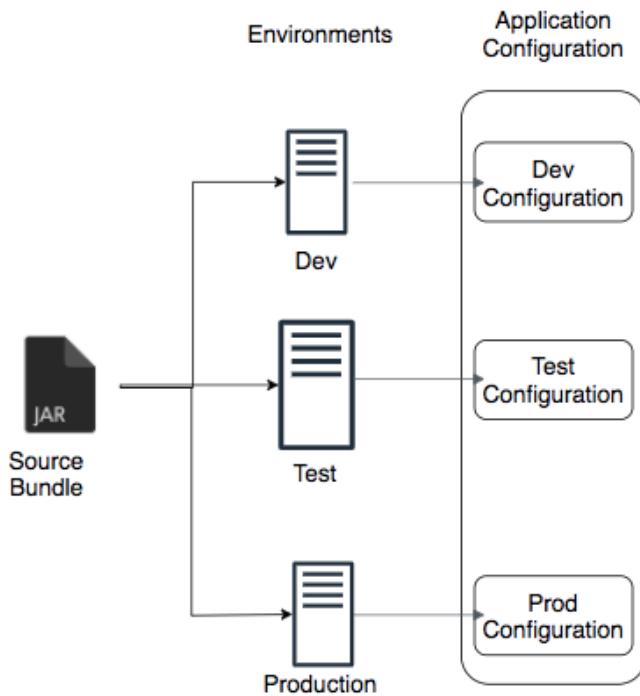
# Cross-Origin Resource Sharing (CORS) - API Gateway

- Front-end web application hosted on a Amazon S3 bucket (front-end-bucket)
- Your APIs are exposed through an API Gateway
- Make CORS configuration on API Gateway allowing access from front-end-bucket URL
  - Enable CORS on API Gateway
- How to configure CORS?
  - REST API Lambda custom (non-proxy) integration - Enable CORS (Configure API Gateway method response and integration response settings)
  - REST API Lambda proxy integrations - Enable CORS on API Gateway. Make (write code) lambda function return headers.
  - HTTP API - Enable CORS (configure properties - allowOrigins, allowMethods, allowHeaders)



# Configuration Management

- You want to connect to a different database in different environments
  - How do you externalize database configuration from the application?
  - How do you decouple your application from the specific configuration needed in a specific environment?
- Considerations:
  - Is the configuration secure?
    - Are the configuration values encrypted?
    - Can you store passwords?
  - How can application retrieve the configured values?
  - What is involved in changing the configuration values?



# AWS Lambda - Environment variables

- Key value pairs directly associated with a Lambda function!
- Code to read environment variable:
  - JavaScript - `process.env.ENV_VAR_NAME`
  - Java - `System.getenv("ENV_VAR_NAME")`
- Reserved Environment Variables are set during initialization:
  - Examples : AWS\_REGION, AWS\_LAMBDA\_FUNCTION\_NAME, AWS\_LAMBDA\_FUNCTION\_VERSION, AWS\_LAMBDA\_LOG\_GROUP\_NAME
- (REMEMBER) Environment variables are locked when a Lambda version is published
- (CONSTRAINT) Publish new Lambda version to change env variables
- (REMEMBER) Integrates with AWS KMS



AWS Lambda

# AWS Systems Manager Parameter Store

- Manage Application Configuration and Secrets
  - Supports hierarchical structure
  - Maintains history of configuration over a period of time
- Multi language SDK support to retrieve configuration:
  - `ssm.get_parameters(Names= ['LambdaSecureString' ] )`
- Simplified Operations:
  - Configuration can be changed without releasing a new Lambda version!
  - Monitoring (CloudWatch), Notifications(SNS) and Auditing(AWS CloudTrail)
- Integrates with:
  - **AWS KMS** - Encrypt your configuration values
  - **Amazon EC2, Amazon ECS, AWS Lambda** - Use configured values from your code
  - **AWS Secrets Manager** - More powerful management for secrets (Automatic Rotation)
  - **CloudFormation, CodeBuild, CodePipeline, CodeDeploy** - Enhanced build & deployment

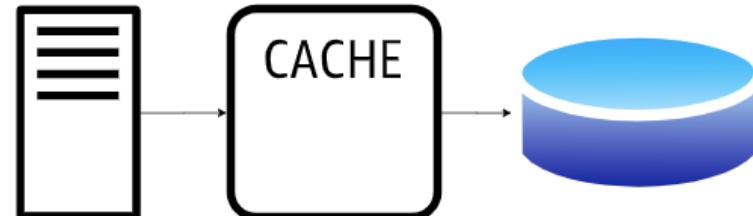
# AWS Secrets Manager

In 28  
Minutes

- Service dedicated to secrets management
  - Rotate, Manage and retrieve database credentials, API keys, and other secrets for your applications
  - Encrypt your secret data using KMS
  - (\$\$\$) Pay for use (**NOT FREE**)
- Simplified Operations:
  - (KEY FEATURE) Rotate secrets automatically without impacting applications
    - Supported for Amazon RDS, Amazon Redshift, and Amazon DocumentDB
  - Configuration can be changed without releasing a new Lambda version!
  - Monitoring (CloudWatch), Notifications(SNS) and Auditing(AWS CloudTrail)
- (RECOMMENDED Workloads) Automatic rotation of secrets for compliance

# Caching

- How can reduce the load on your data stores?
- How can you improve application performance?
  - Use Caching
- Questions to ask when Caching?
  - How often does the data change?
    - Caching is amazing if the data does not change frequently!
  - Am I OK with some stale data?
    - What should be TTL (Time to Live)?
- Caching Usecases:
  - Caching infrequently changing data in Database
  - Caching user sessions from applications



# Caching Strategies - Lazy Loading or Cache Aside

```
get_data(id)

    record = cache.get(id)

    if (record == null)
        record = db.query("YOUR_QUERY", id)
        cache.set(id, record)

    return record
```

- Application sees if data is found in cache.
  - (Cache Hit) If data is found, value from cache is used.
  - (Cache Miss) If data is NOT found, data is retrieved from database and added to cache

# Caching Strategies - Write Through

```
get_data(id)
    return cache.get(id)

save_data(id, value)
    record = db.query("YOUR_QUERY", id, value)
    cache.set(id, record)
    return success
```

- Cache is in sync with backend
  - Cache and database updated at the same time

# Caching Strategy - Comparison

Factor	Lazy Loading	Write Through
Stale data	Possible - Configure TTL	Data is never stale
Node failures	NOT fatal - Increased Latency for subsequent requests	Causes failures (Mitigate using replication)
Cache size	Low - Only requested data is cached	High - All data is in Cache
Reads	Can involve 3 Steps	Directly from Cache
Writes	Update Database Only	2 Steps - Update Cache. Update Database



ElastiCache

# Amazon ElastiCache

- Managed service providing highly scalable and low latency in-memory data store
- Used for distributed caching
- **Two Options:**
  - Redis
  - Memcached
- **Supports:**
  - Lazy Loading
  - Write-Through



ElastiCache

# Amazon ElastiCache for Redis

- Highly scalable and low latency in-memory data store
- Can be used as a **cache, database or message broker**
- Automatic failover with Multi-AZ deployments (if enabled)
- Supports **backup and restore**
- Supports **encryption at-rest (KMS) and in-transit**
- **Use cases:**
  - Caching
  - Session Store
  - Chat and Messaging
  - Gaming Leader boards
  - Geospatial Apps (Ride hailing, restaurant recommendations)
  - Queues



ElastiCache

# Amazon ElastiCache for Memcached

- Simple caching layer to speed up dynamic web applications
  - Non-persistent Pure cache
  - Simple key-value storage
  - Can be used as a **transient session store**
  - **Ideal caching solution for data stores like RDS**
    - DynamoDB Accelerator (DAX) is recommended for DynamoDB
- Features:
  - Create upto 20 cache nodes
- Limitations:
  - Backup and restore NOT supported
  - Does NOT support encryption or replication
  - Does NOT support snapshots
    - When a node fails, all data in the node is lost
    - Reduce impact of failure by using large number of small nodes



ElastiCache

# ElastiCache Memcached vs Redis

- Use ElastiCache Memcached for
  - Low maintenance simple caching solution
  - Easy horizontal scaling with auto discovery
  - Use Case: Fast Session Store
  - Use Case: Cache for Read Only Data (or very infrequently changing data)
- Use ElastiCache Redis for
  - Persistence
  - Publish subscribe messaging
  - Read replicas and failover
  - Encryption



ElastiCache

# ElastiCache vs DAX



DynamoDB



ElastiCache

- DAX is customized for DynamoDB.
  - Very few code changes are needed to cache data from DynamoDB.
- ElastiCache is generic cache.
  - Needs a lot of code changes.

# Caching Application Sessions - ElastiCache vs DynamoDB

In 28  
Minutes



DynamoDB



ElastiCache

- How to create a distribute session store in AWS?
- ElastiCache - MemCached
  - Fast micro second response
  - You will lose session information if a node crashes
- ElastiCache - Redis
  - Can withstand node failures (Replication/Backup/Restore options)
- DynamoDB
  - Millisecond responses

# More AWS Services

# AWS Service Quotas

- AWS account has Region-specific default quotas or limits for each service
  - You don't need to remember all of them :)
- Service Quotas allows you to manage your quotas for over 100 AWS services, from one location

# AWS Directory Service

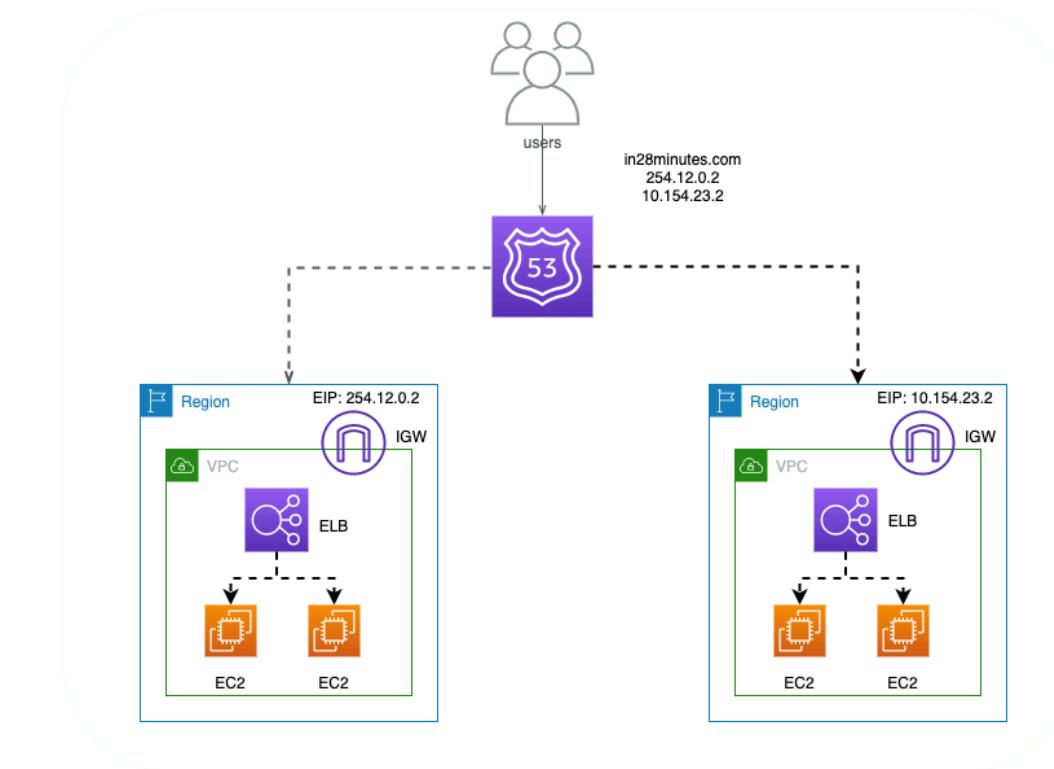
- Provide AWS access to on-premise users without IAM users
- Managed service deployed across multiple AZs
- Option 1 : AWS Directory Service for Microsoft AD
  - More than 5000 Users
  - Trust relationship needed between AWS and on-premise directory
- Option 2 : Simple AD
  - Less than 5000 users
  - Powered by Samba4 and compatible with Microsoft AD
  - Does not support trust relationships with other AD domains
- Option 3 : AD Connector
  - Use your existing on-premise directory with AWS cloud services
  - Your users use existing credentials to access AWS resources



Directory Service

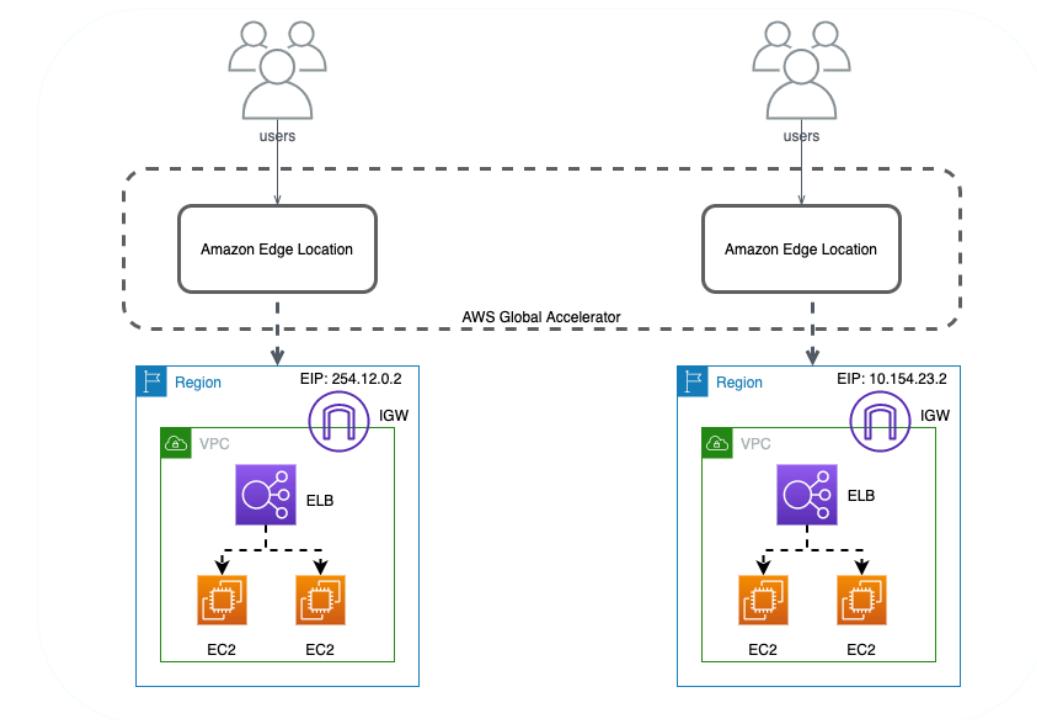
# Need for AWS Global Accelerator

- Cached DNS answers
  - clients might cache DNS answers causing a delay in propagation of configuration updates
- High latency
  - users connect to the region over the internet



# AWS Global Accelerator

- Directs traffic to optimal endpoints over the AWS global network
- Global Accelerator provides you with two static IP addresses
- Static IP addresses are anycast from the AWS edge network
  - Distribute traffic across multiple endpoint resources in multiple AWS Regions
- Works with Network Load Balancers, Application Load Balancers, EC2 Instances, and Elastic IP addresses



# S3 Bucket Policy - Advanced Configuration

```
{  
  "Id": "ExamplePolicy", "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowSSLRequestsOnly",  
      "Action": "s3:*",  
      "Effect": "Deny",  
      "Resource": [ "arn:aws:s3:::your-bucket/*" ],  
      "Condition": {  
        "Bool": { "aws:SecureTransport": "false" }  
      },  
      "Principal": "*"  
    }  
  ]  
}
```

- Bucket policies can be used to:
  - Enforce use of HTTPS
    - Above example - "aws:SecureTransport": "false"
  - Enforce use of encryption with KMS
    - "StringNotLikeIfExists": {"s3:x-amz-server-side-encryption-aws-kms-key-id": "YOUR-KMS-KEY-ARN"}

# Architecture and Best Practices

# Well Architected Framework

- Helps cloud architects build application infrastructure which is:
  - Secure
  - High-performing
  - Resilient and
  - Efficient
- Five Pillars
  - Operational Excellence
  - Security
  - Reliability
  - Performance Efficiency
  - Cost Optimization



# Operational Excellence

In 28  
Minutes



AWS Lambda



CloudFormation



Codepipeline



AWS Config



Cloudwatch

- Avoid/Minimize effort and problems with
  - Provisioning servers
  - Deployment
  - Monitoring
  - Support

# Operational Excellence - Solutions and AWS services

- Use Managed Services
  - You do not need to worry about managing servers, availability, durability etc
- Go serverless
  - Prefer Lambda to EC2!
- Automate with Cloud Formation
  - Use Infrastructure As Code
- Implement CI/CD to find problems early
  - CodePipeline
  - CodeBuild
  - CodeDeploy
- Perform frequent, small reversible changes



AWS Lambda



CloudFormation



Codepipeline



Codebuild



Codedeploy

# Operational Excellence - Solutions and AWS services

- Prepare: for failure
  - Game days
  - Disaster recovery exercises
  - Implement standards with AWS Config rules
- Operate: Gather Data and Metrics
  - CloudWatch (Logs agent), Config, Config Rules, CloudTrail, VPC Flow Logs and X-Ray (tracing)
- Evolve: Get intelligence
  - Use Amazon Elasticsearch to analyze your logs



AWS Config



Cloudwatch



AWS CloudTrail



AWS X-Ray



Amazon ES

# Security Pillar

In 28  
Minutes



AWS IAM



AWS Shield



AWS WAF



AWS KMS



Cloud HSM

- Principle of least privilege for least time
- Security in Depth - Apply security in all layers
- Protect Data in Transit and at rest
- Actively monitor for security issues
- Centralize security policies for multiple AWS accounts

# Security Pillar - Principle of least privilege for least time

In 28  
Minutes



AWS IAM

- Use temporary credentials when possible (IAM roles, Instance profiles)
- Use IAM Groups to simplify IAM management
- Enforce strong password practices
- Enforce MFA
- Rotate credentials regularly

# Security Pillar - Security in Depth

- VPCs and Private Subnets
  - Security Groups
  - Network Access Control List (NACL)
- Use hardened EC2 AMIs (golden image)
  - Automate patches for OS, Software etc
- Use CloudFront with AWS Shield for DDoS mitigation
- Use WAF with CloudFront and ALB
  - Protect web applications from CSS, SQL injection etc
- Use CloudFormation
  - Automate provisioning infrastructure that adheres to security policies



VPC



EC2 AMI



AWS Shield



AWS WAF



CloudFormation

# Security Pillar - Protecting Data at Rest

- Enable Versioning (when available)
- Enable encryption - KMS and Cloud HSM
  - Rotate encryption keys
- Amazon S3
  - SSE-C, SSE-S3, SSE-KMS
- Amazon DynamoDB
  - Encryption Client, SSE-KMS
- Amazon Redshift
  - AWS KMS and AWS CloudHSM
- Amazon EBS, Amazon SQS and Amazon SNS
  - AWS KMS
- Amazon RDS
  - AWS KMS, TDE



AWS KMS



Cloud HSM

# Security Pillar - Protecting Data in Transit

In 28  
Minutes



Certificate Manager

- Data coming in and going out of AWS
- By default, all AWS API use HTTPS/SSL
- You can also choose to perform client side encryption for additional security
- Ensure that your data goes through AWS network as much as possible
  - VPC Endpoints and AWS PrivateLink

# Security Pillar - Detect Threats

In 28  
Minutes



Cloudwatch



Organizations

- Actively monitor for security issues:
  - Monitor CloudWatch Logs
  - Use Amazon GuardDuty to detect threats and continuously monitor for malicious behavior
- Use AWS Organization to centralize security policies for multiple AWS accounts

# Reliability

In 28  
Minutes



AWS Lambda



Amazon SQS



Amazon SNS



API Gateway



AutoScaling

- Ability to
  - Recover from infrastructure and application issues
  - Adapt to changing demands in load

# Reliability - Best Practices

In 28  
Minutes



AutoScaling



CW Alarm



Cloudwatch

- Automate recovery from failure
  - Health checks and Auto scaling
  - Managed services like RDS can automatically switch to standby
- Scale horizontally
  - Reduces impact of single failure
- Maintain Redundancy
  - Multiple Direct Connect connections
  - Multiple Regions and Availability Zones

# Reliability - Best Practices

In 28  
Minutes



AWS Lambda



Amazon SQS



Amazon SNS



API Gateway

- Prefer serverless architectures
- Prefer loosely coupled architectures
  - SQS, SNS
- Distributed System Best Practices
  - Use Amazon API Gateway for throttling requests
  - AWS SDK provides retry with exponential backoff

# Loosely coupled architectures

- ELB
  - Works in tandem with AWS auto scaling
- Amazon SQS
  - Polling mechanism
- Amazon SNS
  - Publish subscribe pattern
  - Bulk notifications and Mobile push support
- Amazon Kinesis
  - Handle event streams
  - Multiple clients
  - Each client can track their stream position



ELB



Amazon SNS



Amazon SQS



Kinesis

# Troubleshooting on AWS - Quick Review

Option	Details	When to Use
Amazon S3 Server Access Logs	S3 data request details - request type, the resources requested, and the date and time of request	Troubleshoot bucket access issues and data requests
Amazon ELB Access Logs	Client's IP address, latencies, and server responses	Analyze traffic patterns and troubleshoot network issues
Amazon VPC Flow Logs	Monitor network traffic	Troubleshoot network connectivity and security issues

# Troubleshooting on AWS - Quick Review

Option	Details	When to Use
Amazon CloudWatch	Monitor metrics from AWS resources	Monitoring
Amazon CloudWatch Logs	Store and Analyze log data from Amazon EC2 instances and on-premises servers	Debugging application issues and Monitoring
AWS Config	AWS resource inventory. History. Rules.	Inventory and History
Amazon CloudTrail	History of AWS API calls made via AWS Management Console, AWS CLI, AWS SDKs etc.	Auditing and troubleshooting. Determine who did what, when, and from where.

# Performance Efficiency

- Meet needs with minimum resources (efficiency)
- Continue being efficient as demand and technology evolves

# Performance Efficiency - Best Practices

In 28  
Minutes



AWS Lambda



API Gateway



Cloudwatch



Amazon SQS

- Use Managed Services
  - Focus on your business instead of focusing on resource provisioning and management
- Go Serverless
  - Lower transactional costs and less operational burden
- Experiment
  - Cloud makes it easy to experiment
- Monitor Performance
  - Trigger CloudWatch alarms and perform actions through Amazon SQS and Lambda

# Performance Efficiency - Choose the right solution

- Compute
  - EC2 instances vs Lambda vs Containers
- Storage
  - Block, File, Object
- Database
  - RDS vs DynamoDB vs RedShift ..
- Caching
  - ElastiCache vs CloudFront vs DAX vs Read Replicas
- Network
  - CloudFront, Global Accelerator, Route 53, Placement Groups, VPC endpoints, Direct Connect
- Use product specific features
  - Enhanced Networking, S3 Transfer Acceleration, EBS optimized instances



AWS Lambda



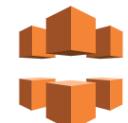
Amazon S3



DynamoDB



ElastiCache



CloudFront

# Cost Optimization

- Run systems at lowest cost



AutoScaling



AWS Lambda



Trusted Advisor



Cloudwatch



CloudFront

# Cost Optimization - Best Practices

- Match supply and demand
  - Implement Auto Scaling
  - Stop Dev/Test resources when you don't need them
  - Go Serverless
- Track your expenditure
  - Cost Explorer to track and analyze your spend
  - AWS Budgets to trigger alerts
  - Use tags on resources



AutoScaling



AWS Lambda



Trusted Advisor



Cloudwatch



CloudFront

# Cost Optimization - Choose Cost-Effective Solutions

- Right-Sizing : Analyze 5 large servers vs 10 small servers
  - Use CloudWatch (monitoring) and Trusted Advisor (recommendations) to right size your resources
- Email server vs Managed email service (charged per email)
- On-Demand vs Reserved vs Spot instances
- Avoid expensive software : MySQL vs Aurora vs Oracle
- Optimize data transfer costs using AWS Direct Connect and Amazon CloudFront



AutoScaling



AWS Lambda



Trusted Advisor



Cloudwatch



CloudFront

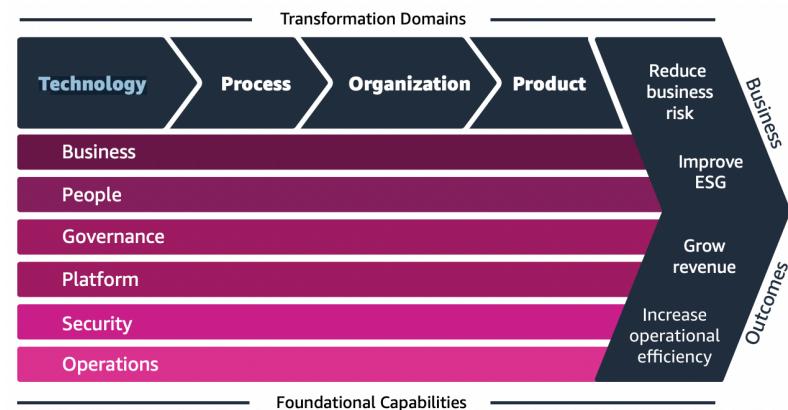
# [ADDITION] AWS Well-Architected Pillar - Sustainability

- **Sustainability:** "meeting the needs of the present without compromising the ability of future generations to meet their own needs." (United Nations Definition)
  - **Various Reports:** Moving to cloud helps you reduce your carbon footprint upto 80%
  - **Design principles for sustainability in the cloud**
    - 1: Understand your impact (AWS Customer Carbon Footprint Tool)
    - 2: Establish sustainability goals
    - 3: Maximize utilization
    - 4: Adopt new, more efficient hardware & software offerings
      - Make your designs flexible
    - 5: Use managed services
    - 6: Reduce the downstream impact of your cloud workloads
      - Reduce need for device upgrades



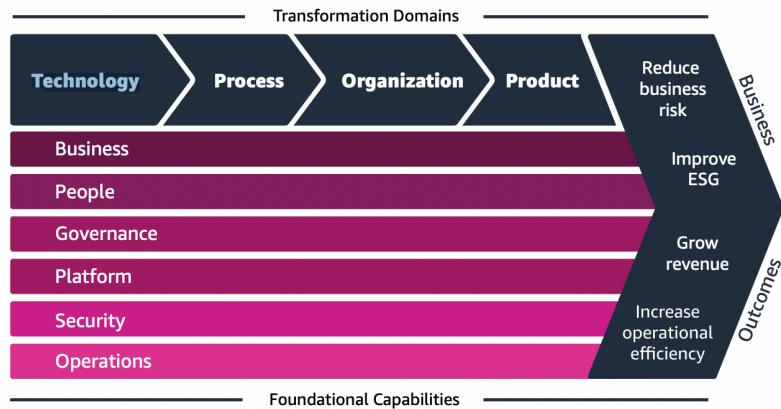
# Exploring AWS Cloud Adoption Framework

- You have decided to move to the cloud:
  - How do you ensure that you are following the best practices while adopting the cloud?
  - **Solution:** The AWS Cloud Adoption Framework (AWS CAF)
- Leverage AWS experience and best practices
- Transformation Domains:
  - **Technology:** Platform modernization
  - **Process:** Automate, and optimize your business operations (using AI/ML insights,...)
  - **Organization:** Restructure business & tech teams
  - **Product:** Re-imagine your business model

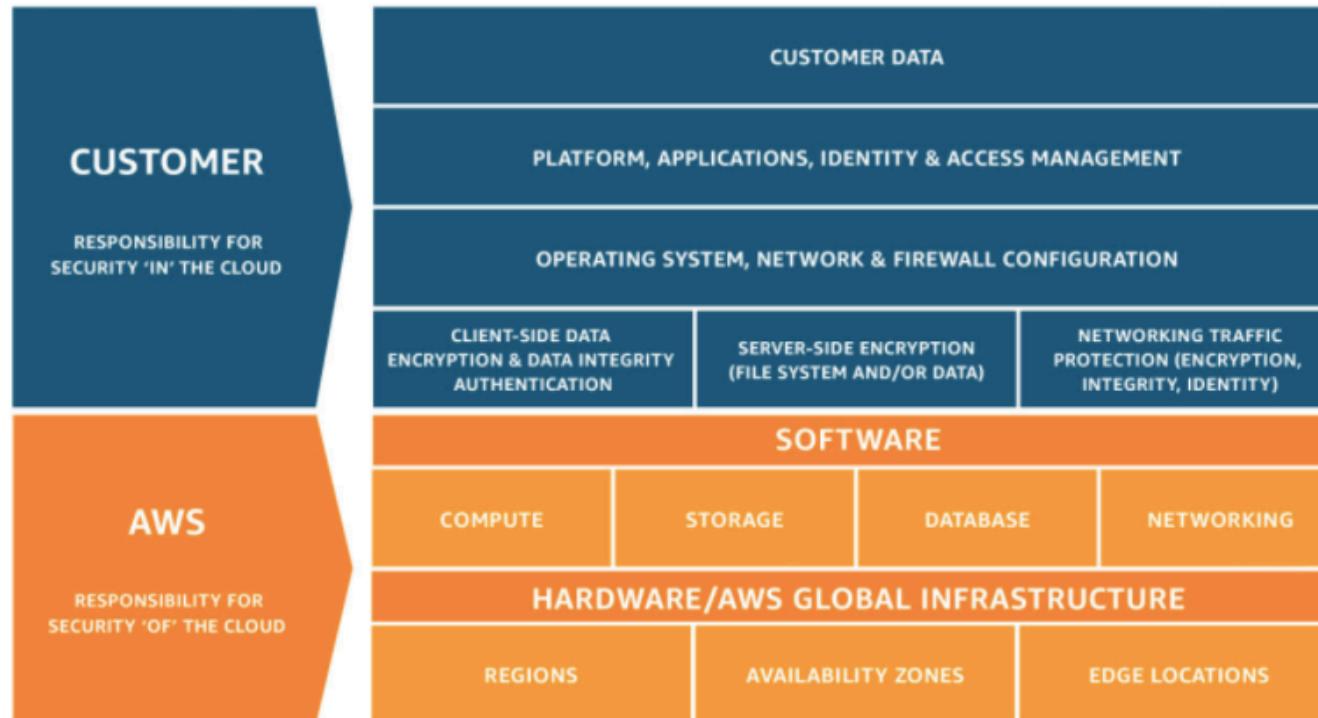


# AWS Cloud Adoption Framework (AWS CAF) - An Overview

- Capabilities grouped in **SIX perspectives**
  - **1: Business:** Strategy, Product Mgmt,..
  - **2: People:** Cloud Fluency, Culture Evolution, ..
  - **3: Governance:** Program/Project/Risk/Data Mgmt ..
  - **4: Platform:** Modern App. development (Container orchestration, Serverless), CI/CD/IaaS/Observability...
  - **5: Security:** IAM, Threat Detection, Incident Response..
  - **6: Operations:** Observability, Incident/Release Mgmt..
- **4 Key Phases**
  - **Envision phase:** Identify & prioritize opportunities
  - **Align phase:** Identify capability gaps
  - **Launch phase:** Deliver pilot initiatives
  - **Scale phase:** Expand & deliver business value



# Shared Responsibility Model



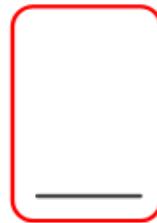
<https://aws.amazon.com/compliance/shared-responsibility-model/>

Security & Compliance is shared responsibility between AWS and customer

# Shared Responsibility Model - Amazon EC2



EC2



Security Group

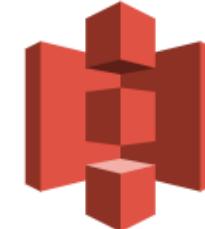


EC2 AMI

- Amazon EC2 instances is Infrastructure as a Service (IaaS)
- You are responsible for
  - Guest OS (incl. security patches)
  - Application software installed
  - Configuring Security Groups (or firewalls)
- AWS is responsible for infrastructure layer only

# Shared Responsibility Model - Managed Services

- Amazon S3 & DynamoDB are managed services
- AWS manages infrastructure layer, OS, and platform
- You are responsible for
  - Managing your data
  - Managing security of data at rest(encryption)
  - Managing security of data in transit
    - Mandating SSL/HTTPS
    - Using the right network - AWS global network or dedicated private network when possible
  - Managing access to the service
    - Configure right permissions (IAM users/roles/user policies/resource policies)
    - (FOR AWS RDS) Managing in database users
    - Configuring the right security groups (control inbound and outbound traffic)
    - Disabling external access (public vs private)



Amazon S3



DynamoDB

# DAA C03 Updates

# One Enterprise - Many AWS accounts - AWS Control Tower

- Having **multiple AWS accounts** helps businesses meet their business, governance, security, and operational requirements:
  - 1: Group workloads based on their business purpose
  - 2: Create environment specific security controls
  - 3: Make cost management easier
    - Each business unit responsible for their costs
  - **AWS Organizations:** Centralized Governance of multiple AWS accounts
    - Ex: Billing, Config Rules, Service control policies (SCPs), Firewall Manager rules, CloudTrail data
- **AWS Control Tower:** Easiest way to set up and govern a secure, multi-account AWS environment (also called a landing zone)
  - Sets up AWS Organizations
  - Ensures that all new accounts adhere to organization policies
  - Configure guardrails (AWS Config Rules, Service control policies (SCPs) etc) and check compliance



# More Security Services in AWS

Service	Description
Certificate Manager	Provision, manage, deploy, and renew SSL/TLS certificates on the AWS platform
AWS Firewall Manager	Central management of firewall rules across multiple AWS accounts 1: AWS WAF rules 2: AWS Shield Advanced protections 3: Security groups and NACLs
AWS Artifact	No cost, self-service portal for on-demand access to AWS's compliance reports (Service Organization Control (SOC) reports, Payment Card Industry (PCI) reports...)
AWS Audit Manager	Enterprise infrastructure undergoes regular auditing(SOC, PCI DSS, GDPR...) Reduce Manual effort in generating auditor-friendly reports

# More Security Services in AWS - 2

Service	Description
<b>Amazon GuardDuty</b>	Continuously monitor AWS environment for suspicious activity (Intelligent Threat Detection) Analyze AWS CloudTrail events, VPC Flow Logs etc
<b>Amazon Inspector</b>	Scan AWS workloads - EC2, Containers - for software vulnerabilities and unintended network exposure with a single click
<b>AWS Security Hub</b>	Single dashboard for comprehensive view of your security state within AWS Consolidated findings from enabled security services - Amazon GuardDuty, Amazon Inspector, Amazon Macie, ..
<b>Amazon Detective</b>	Analyze, and quickly identify the root cause of potential security issues Collects log data from your AWS resources Uses machine learning enabling you to perform more efficient security investigations.

# Hybrid Cloud - Compute

- Many enterprises are going hybrid-cloud and multi-cloud
- How about running AWS services on customer-managed infrastructure?
  - **Amazon ECS Anywhere:** Run ECS applications in on-premises environments and the cloud with centralized management
  - **Amazon EKS Anywhere:** Create and operate Kubernetes clusters on customer-managed infrastructure
    - Makes use of **Amazon EKS Distro:**
      - Amazon EKS customized Kubernetes distribution
      - Provides extended support even after community support expires
  - **Advantages:**
    - Meet your compliance needs
    - Use familiar ECS and EKS tooling
    - Intermediate step before workloads can be shifted to cloud



# Databases - A Quick Review

In 28  
Minutes

Service	Description
RDS	Managed Relational Database Service
DynamoDB	Managed NoSQL Database
Amazon DocumentDB	Fully-managed MongoDB-compatible database service
ElastiCache	In-Memory Cache
Amazon Keyspaces	Serverless Cassandra-compatible Column-family database
Neptune	Fast, reliable graph database built for the cloud
Amazon Timestream	Fast, scalable, and serverless time series database for IoT and operational applications
Amazon Quantum Ledger Database (Amazon QLDB)	Fully managed ledger database Immutable, cryptographically verifiable log of data changes Use cases: Financial transactions, System-of-record applications

# Big Data & Datawarehousing in AWS

Service	Scenario
Amazon Redshift	Run complex queries (SQL) against data warehouse - housing structured and unstructured data pulled in from a variety of sources
Amazon EMR	Managed Hadoop. Large scale data processing with high customization (machine learning, graph analytics) Important tools in Hadoop ecosystem are natively supported (Pig, Hive, Spark or Presto)
Amazon Kinesis	Data Streams + Firehose (ingestion) + Analytics + Video Streams
Amazon Managed Streaming for Apache Kafka (Amazon MSK)	Managed Kafka Service in AWS
Amazon Timestream	Fast, scalable, and serverless time series database for IoT and operational applications

# Big Data & Datawarehousing in AWS - 2

Service	Scenario
Amazon S3	Can be used as a Data Lake
AWS Lake Formation	Makes it easy to set up a secure data lake 1: Collect data from databases and object storage 2: Move it to S3 3: Clean data and secure access
Amazon Redshift Spectrum	Run queries directly against S3 without worrying about loading entire data from S3 into a data warehouse. Scale compute and storage independently.
Amazon Athena	Quick ad-hoc queries without provisioning a compute cluster (serverless) Amazon Redshift Spectrum is recommended if you are executing queries frequently against structured data
AWS Glue	Fully managed extract, transform, and load (ETL) service Simplify data preparation (capturing metadata) for analytics Transform data between formats (.csv to .parquet)

# Observability in AWS

- **Amazon CloudWatch, AWS X-Ray:** Monitoring, Alarms,.. & Tracing
- **AWS Distro for OpenTelemetry:** AWS-supported OpenTelemetry dist.
  - Instrument your apps with it > Automate all telemetry (metrics, logs, and traces)
- **Amazon Managed Service for Prometheus** - Fully managed service for Prometheus.
  - Monitoring & alerting for containerized applications (Amazon EKS, Amazon ECS)
- **Amazon OpenSearch Service** - Perform interactive log analytics, real-time application monitoring, website search (ELK stack)
- **Amazon Managed Grafana** - Fully managed service for Grafana
  - Visualize and analyze your metrics, logs, and traces
  - Integrate with multiple data sources in your observability stack ( CloudWatch, Amazon Managed Service for Prometheus, Amazon Elasticsearch/OpenSearch)!



- Enhanced IaaC for serverless & containerized applications
  - Infrastructure, CI/CD pipeline, observability tools
- Create Std. Environments & Services across Enterprise:
  - Steps:
    - 1: Create Environment Templates
    - 2: Create Environment
    - 3: Create Service Template and Map a Source Repo
    - 4: Commit code
    - 5: Pipeline triggered
    - 6: Service deployed

# Get Ready

# Certification Resources

Title	Link
Certification - Home Page	<a href="https://aws.amazon.com/certification/certified-developer-associate/">https://aws.amazon.com/certification/certified-developer-associate/</a>
AWS Architecture Home Page	<a href="https://aws.amazon.com/architecture/">https://aws.amazon.com/architecture/</a>
AWS FAQs	<a href="https://aws.amazon.com/faqs/">https://aws.amazon.com/faqs/</a> (Lambda, API Gateway, Dynamo DB, Cognito etc)

# Certification Exam

- **Multiple Choice Questions**
  - Type 1 : Single Answer - 4 options and 1 right answer
  - Type 2 : Multiple Answer - 5 options and 2 right answers
- **No penalty for wrong answers**
  - Feel free to guess if you do not know the answer
- **65 questions and 130 minutes**
  - Ask for 30 extra minutes BEFORE registering if you are **non native English speaker**
- Result immediately shown after exam completion
- Email with detailed scores (a couple of days later)

# Certification Exam - My Recommendations

- Read the entire question
- Read all answers at least once
- Identify and write down the **key parts of the question:**
  - Features: serverless, key-value, relational, auto scaling
  - Qualities: cost-effective, highly available, fault tolerant
- If you do NOT know the answer, **eliminate wrong answers first**
- **Flag questions** for future consideration and review them before final submission

# You are all set!

# Let's clap for you!

- You have a lot of patience! **Congratulations**
- You have put your best foot forward to be an AWS Certified Developer Associate
- Make sure you prepare well (Use practice tests)
- Good Luck!

# Do Not Forget!

- Recommend the course to your friends!
  - Do not forget to review!
- **Your Success = My Success**
  - Share your success story with me on LinkedIn (Ranga Karanam)
  - Share your success story and lessons learnt in Q&A with other learners!

# What next?

- Go Deeper into AWS!
  - Three things I would recommend
    - Serverless (Lambda, API Gateway, DynamoDB)
    - Elastic Beanstalk
    - ECS
- Learn other Cloud Platforms:
  - Gartner predicts a multi cloud world soon
  - Get certified on AWS, Azure and Google Cloud
- Learn DevOps (Containers and Container Orchestration)
- Learn Full Stack Development



AWS Lambda



ECS



API Gateway

# What Next?

# FASTEST ROADMAPS

in28minutes.com

