

AWS CERTIFIED DEVELOPER ASSOCIATE

<< Exam Review >>

Getting Started

- AWS has 200+ services. Exam expects knowledge of 40+ services.
- You might have used these services before.
- But you need to remember all details when attending the exam
 - How do you review everything before the exam?
- **Our Goal :** Enable you to quickly review for **AWS Certified Developer Associate** exam
 - (Remember) This is a crash review course!
- **Our Approach:** Quick Review Videos with Presentations and Comparisons
 - (Recommended) Do not hesitate to replay videos!
 - (Recommended) Have Fun!



EC2



DynamoDB



AWS Lambda



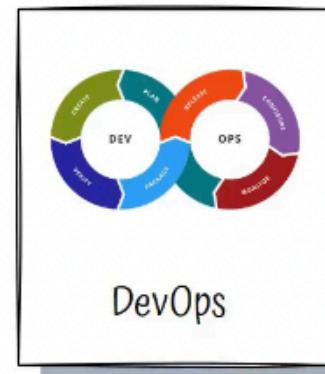
API Gateway



Amazon S3

FASTEST ROADMAPS

in28minutes.com



Getting Started

In 28
Minutes

What?	Description
Cloud	<p>Elasticity. On-demand resource provisioning.</p> <p>Trade "capital expense (capex)" for "variable expense (opex)" (Pay-as-you-go)</p> <p>"Go global" in minutes</p>
Availability	<p>Are the applications available when the users need them?</p> <p>99.99% availability = 4 minutes of downtime in a month</p>
AWS Regions	<p>20+ regions around the world (us-east-1,eu-west-2)</p> <p>High Availability and Low Latency</p>
Availability Zones	<p>Each Region has multiple AZs</p> <p>Discrete data centers, with redundant power, networking, and connectivity</p> <p>Increase availability of applications in the same region</p> <p>us-east-1 has 6 AZs - us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, us-east-1f</p> <p>eu-west-2 has 3 AZs - eu-west-2a, eu-west-2b, eu-west-2c</p>

Compute Services - Overview

Service	Description
Amazon EC2 + ELB	Traditional Approach (Virtual Servers + Load Balancing)
AWS Elastic Beanstalk	Simplify management of web applications and batch applications. Automatically creates EC2 + ELB (load balancing and auto scaling)
AWS Elastic Container Service (ECS)	Simplify running of microservices with Docker containers. Run containers in EC2 based ECS Clusters
AWS Fargate	Serverless version of ECS
AWS Lambda	Serverless - Do NOT worry about servers

Serverless Fundamentals - Lambda and API Gateway

Going Serverless with AWS Lambda

- **Serverless - Don't worry about servers. Focus on building your app**
 - Remember: Serverless does NOT mean "No Servers"
 - **Serverless for me:**
 - You don't worry about infrastructure
 - Flexible scaling and automated high availability
 - Pay for use NOT FOR SERVERS
 - **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 - Write and Scale Your Business Logic**
 - Supports Node.js (JavaScript), Java, Python, Go, C# and more..
 - Don't worry about servers or scaling or availability
 - Pay for Use: Number of requests, Duration of requests and Memory Configured
 - Free tier - 1M free requests per month
 - Integrates with AWS X-Ray(tracing), AWS CloudWatch (monitoring and logs)



AWS Lambda



API Gateway



DynamoDB

AWS Lambda - Remember

- Allocate memory in 64MB increments from 128MB to 3GB
 - More Memory => More Cost and More CPU
- Maximum allowed execution time - 900 seconds (default - 3 seconds)
- **Lambda execution role** - Grants permissions to AWS Resources
 - Assigned when creating a function
 - Function assumes this role when invoked
 - Predefined policies simplify permission configuration:
 - `AWSLambdaBasicExecutionRole` – Upload logs to CloudWatch.
 - Others include `AWSLambdaDynamoDBExecutionRole`, `AWSLambdaSQSQueueExecutionRole`, `AWSLambdaVPCAccessExecutionRole`, `AWSXRayDaemonWriteAccess`
- Lambda logs are automatically stored in CloudWatch Logs
 - Default log group name: `/aws/lambda/function-name`
 - If logs are not visible:
 - Check if Lambda Function has permissions to write to CloudWatch Logs(Execution role)



AWS Lambda Execution Context

In 28
Minutes

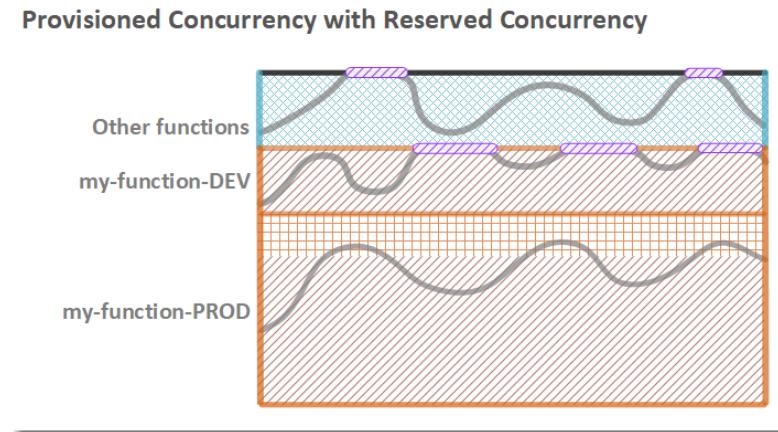
```
const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event, context) => {
```

- **Execution Context** - Temp runtime environment used by Lambda function
 - Lambda tries to reuse execution context when possible
 - Objects declared outside handler functions remain initialized (dynamo in above example)
 - Each execution context has /tmp directory with 512 MB disk space
 - Reused across invocations using same execution 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
- **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 - Concurrency

- **Function concurrency** - 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 lambda function always runs?
 - Use **Reserved Concurrency**
 - Other functions use remaining concurrency from regional quota
- How to avoid cold starts?
 - Use **Provisioned Concurrency**
 - Runs continually (More expensive)
 - Can be configured on a Lambda function version or an alias
- When requests exceed allowed concurrency:
 - Throttling error (429 status code)



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

AWS Lambda - Invocation Types

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

- **Synchronous Invocation:**

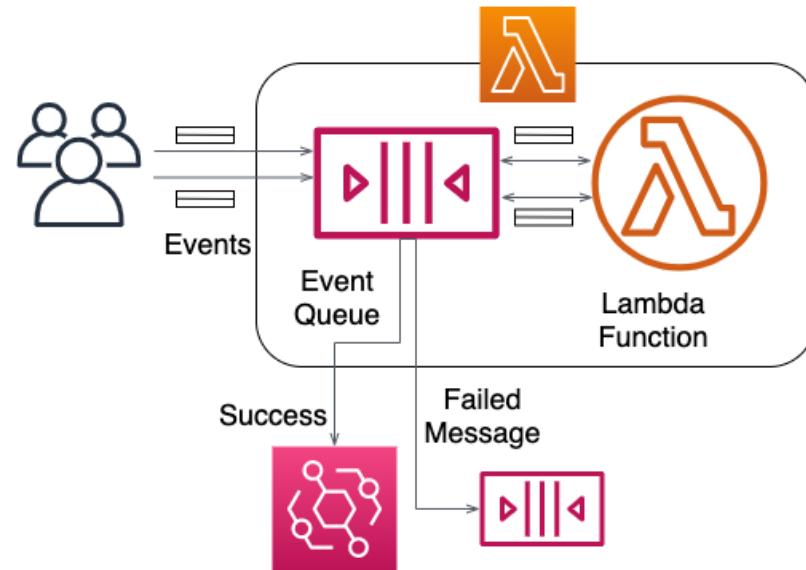
- Lambda runs the function and waits for response
- Lambda returns the response with additional data (version of executed lambda function)
- Example services: AWS API Gateway, Amazon CloudFront

- **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)

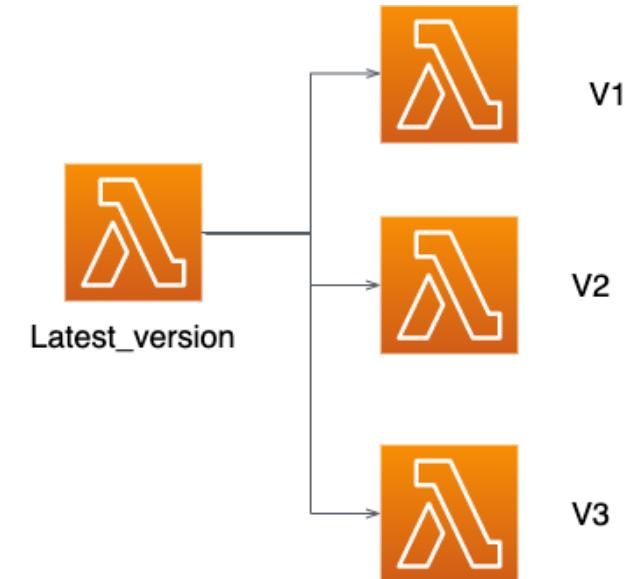
AWS Lambda - Asynchronous Invocation - Errors

- 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)



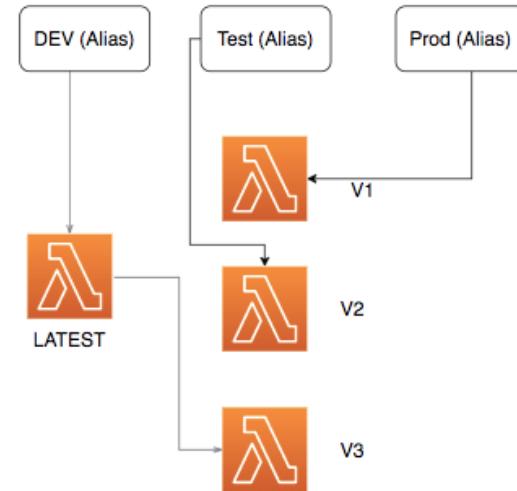
AWS Lambda - Versioning

- 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 versions



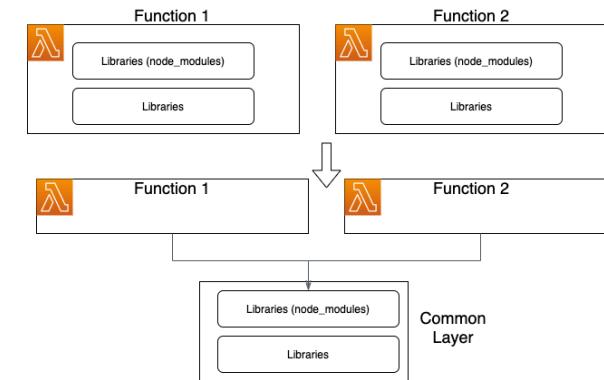
AWS Lambda - Alias

- 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 (Canary 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
Does Lambda scale up or out when it receives multiple requests?	Lambda scales out (multiple instances)
How can you increase the CPU available to a Lambda function?	Increase available memory
How do you enable tracing in Lambda functions?	<ol style="list-style-type: none">1. Give Permissions to Execution Role2. Enable Tracing with X-Ray
How can you make a Lambda function run faster?	Increase memory
Where can you store a temporary file of 100 MB when executing a Lambda?	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
Event notifications from an S3 bucket trigger Lambda function to create thumbnail files from large images in the S3 bucket	Create an Alias for your Lambda function and use it from the S3 bucket

Amazon API Gateway

- 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**" - "front door" to your APIs
 - Fully managed - "publish, maintain, monitor, and secure APIs at any scale"
 - Integrates with AWS Lambda or any web application
 - Supports HTTP(S) and WebSockets
 - Serverless. Pay for use (API calls and connection duration)



API Gateway - API Types

- **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**
 - Also used to build RESTful API
 - Newer, Simpler, Cheaper and Low Latency
 - Automatic Deployments and Default Stage
- **WebSocket API**
 - Persistent connections with clients
 - Allows full-duplex communication
- Names are little confusing



REST API - API Gateway

- API Gateway acts as a front-door for different backend systems

- Integrations:

- 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



- Endpoint Types:

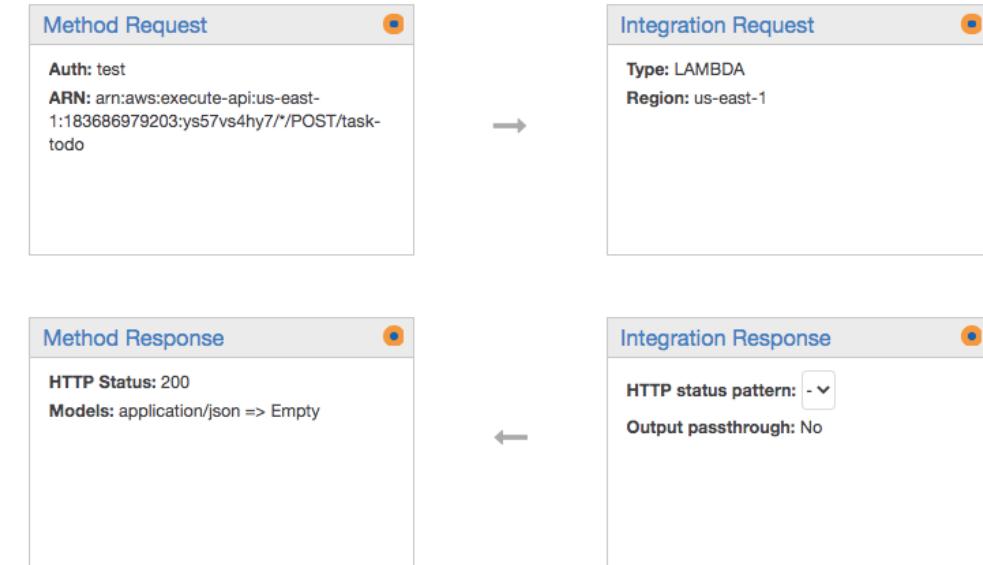
- Edge Optimized (default) - Recommended for geographically distributed clients
 - API requests are routed to the nearest CloudFront Edge Location
 - Regional - Recommended for clients in a single region
 - Private - Can only be accessed from your VPC using an interface VPC endpoint

- Deployment Stages:

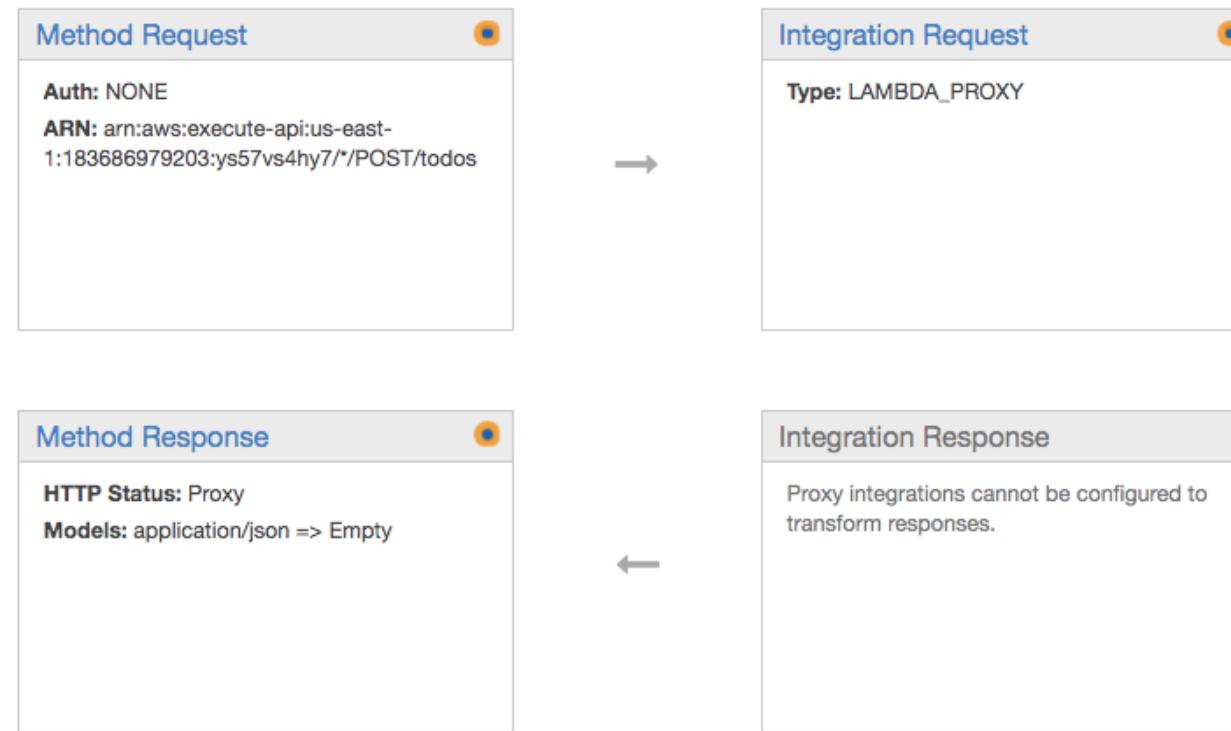
- Deploy API Gateway to different environments - Dev, Test, UAT, Prod etc.
 - Use Stage variables for environment configuration:
 - Example: Connect to different Lambda aliases in different stages

REST API Gateway - Custom Integration (Default)

- **Integrations** define request/response transformation to/from lambda
- **Request Transformation:** Configure Mapping Template in Integration Request
- **Response Transformation:** Configure 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

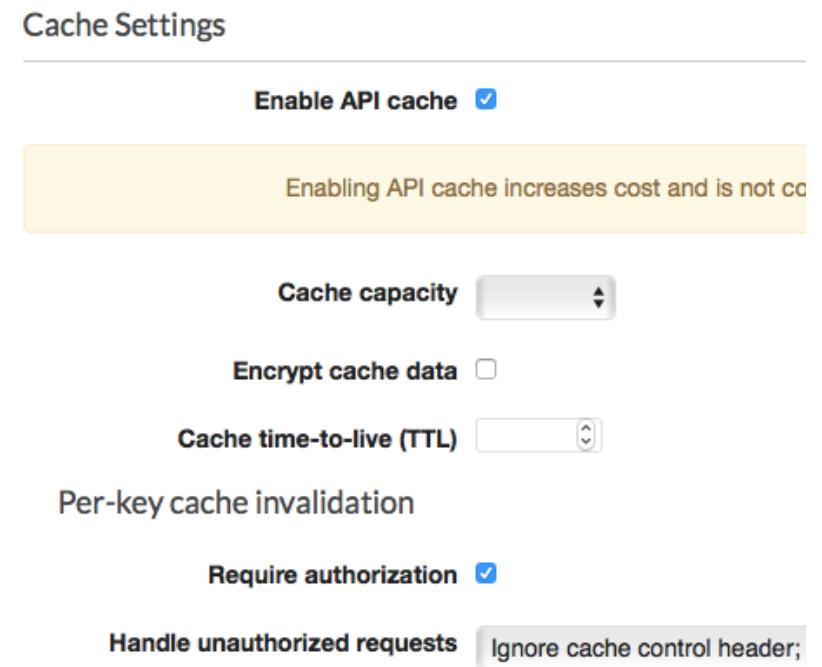
```
{  
  "message": "Welcome"  
}
```

Response Headers

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

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



API Gateway - Canary Releases

- Test new version of the software in production while base version is still live
 - Small % of traffic routed to new version
 - New version is either promoted or rolled back
- Creating a Canary Release
 - Step 1 : Create a Canary: Configure
 - Step 2 : Deploy new release to Canary
 - Step 3 (After Testing) : Promote Canary to 100%
- Configuration:
 - Percentage of requests to send to Canary
 - Configure Canary Stage Variables
 - Override existing stage variables or add new stage variables

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.

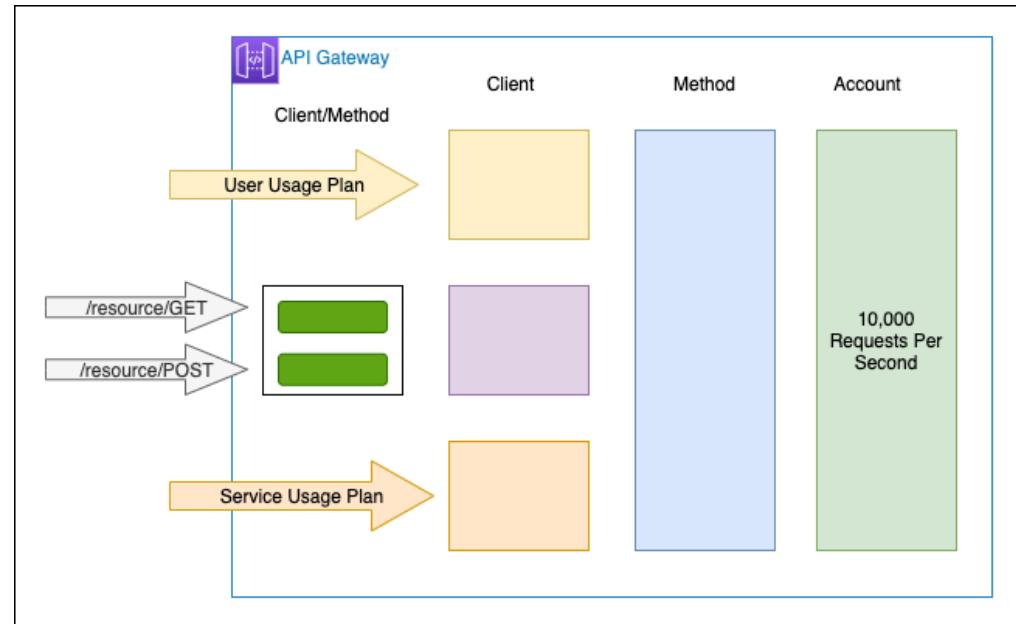
Canary Stage Variables

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



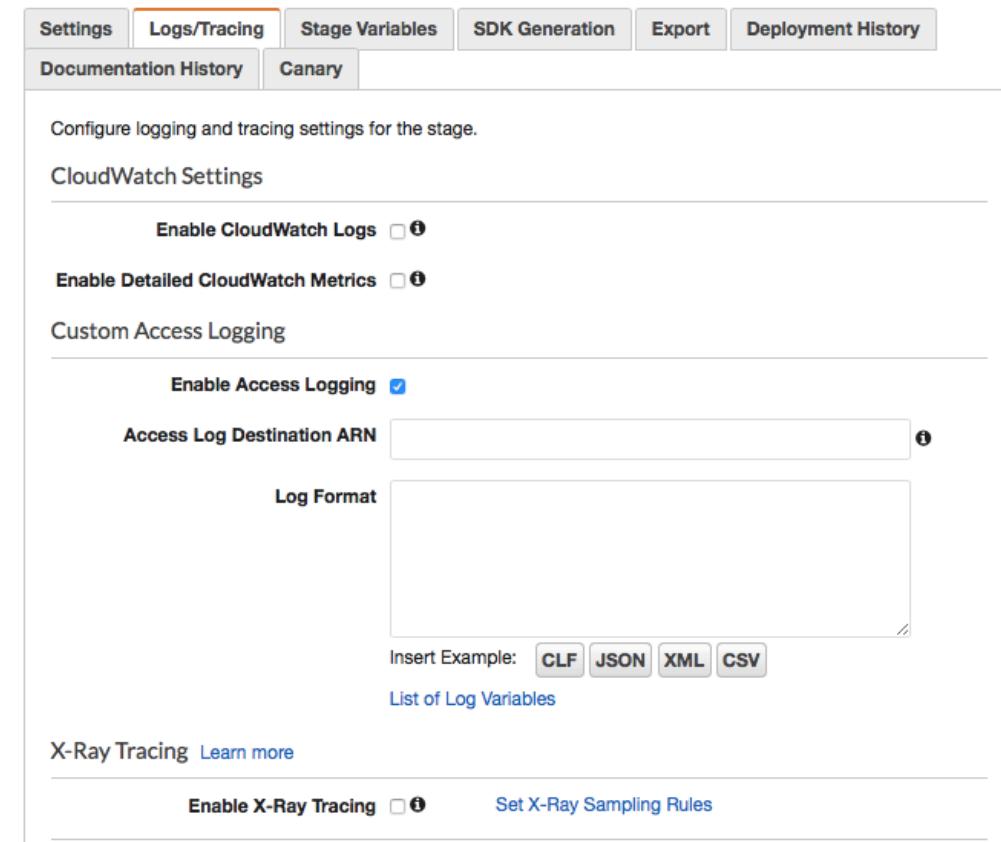
Control Access - Resource policies

```
{  
  "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



HTTP API - API Gateway

- Problems with REST API - API Gateway:
 - Lot of features very few AWS customers made use of
 - Little complex to setup (transformations etc)
- "HTTP API" - Simpler API Gateway (Confusing Naming):
 - Newer, Cheaper, Low Latency and Simpler: Less features, Easier to setup
 - Two payload versions - 1.0 and 2.0
 - 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 - Return a valid response JSON return {"message": "Welcome"}



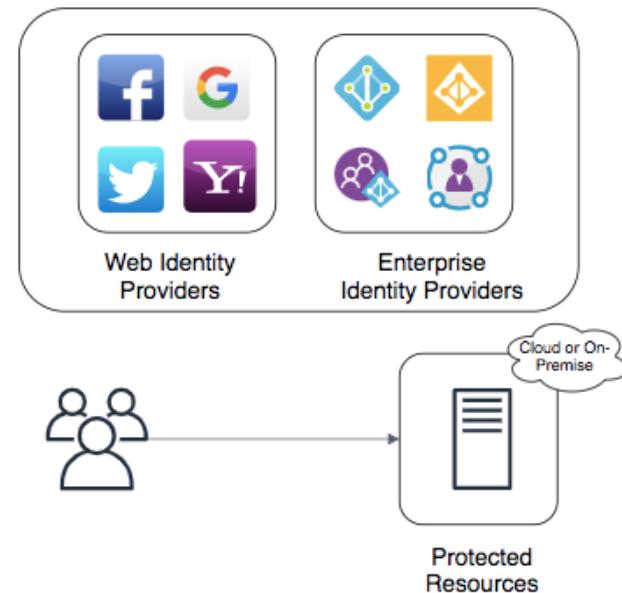
API Gateway - Scenario Questions

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 - map to different Lambda versions
Expose API around a backend SOAP web service	Use Mapping Templates to convert JSON to XML
You are releasing an API with breaking change. You do NOT want to impact existing clients.	Deploy new version to a new stage
An API Gateway is invoking a Lambda. What happens 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?	By using header <i>Cache-Control:max-age=0</i> . User Policy allows <i>execute-api: InvalidateCache</i>
Create customized plans for API Consumers - Basic, Premium, Full	Use Usage Plans

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

- Add authentication and authorization to your mobile and web apps
 - Integrate with web identity providers (ex: Google, Facebook)
 - Add multi-factor authentication (MFA), phone and email verification
 - Sync user data across devices, platforms, and applications
- **User Pools:** Create your own secure and scalable user directory
 - Create sign-up (or registration) pages
 - Customizable web UI to sign in users (with option to social sign-in)
 - Integrates with Application Load Balancer and API Gateway
 - Provides triggers to customize workflow - Pre Authentication Lambda Trigger, Pre Sign-up Lambda Trigger, Post Confirmation Lambda Trigger etc
- **Identity pools:** Provide access to AWS resources to your users
 - Integrate with your own user pool or OpenID Connect provider (Amazon, Apple, Facebook, Google+, Twitter) or SAML identity providers (Corporate)
 - Allows multiple authentication (identity) providers



Amazon Cognito

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

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

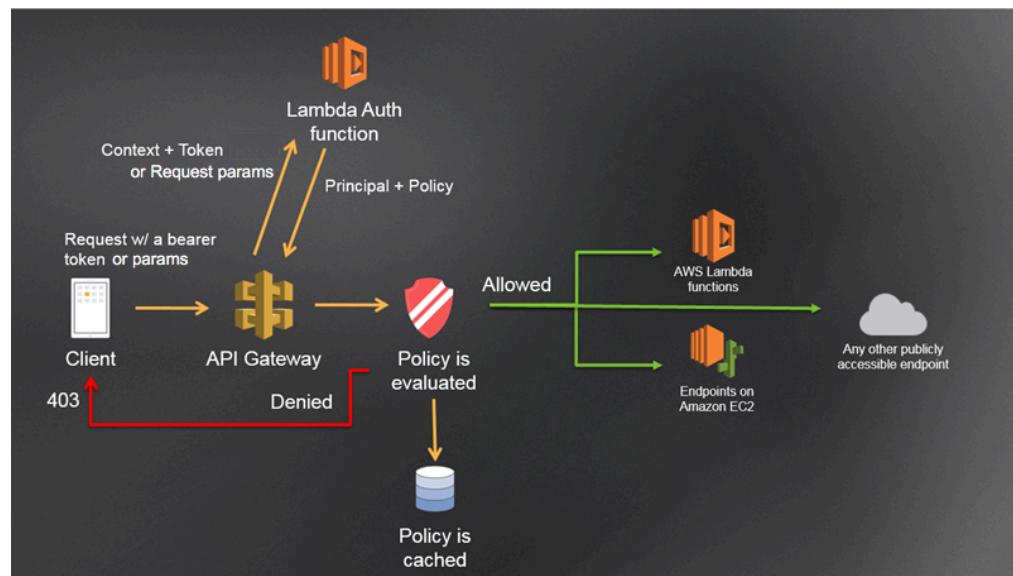
Cognito User Pool * us-east-1

Token Source * Token Validation

Create **Cancel**

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/"
        }
    ]
}
```

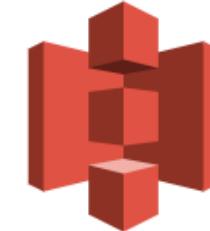
Amazon S3 Fundamentals

Storage in AWS - Overview

Type	Description
Object	Amazon S3 (Very Flexible) Store large objects using a key-value approach
Block	Storage connected to one EC2 instance. Your Hard Disks. Elastic Block Store (EBS - Permanent) Instance Store (Ephemeral)
File	File Share. Share storage between EC2 instances. EFS (Linux) FSx Windows FSx for Lustre (High Performance)
Archival	Amazon Glacier Extremely low cost storage for archives and long-term backups.
Hybrid	AWS Storage Gateway Cloud + On Premise

Amazon S3 (Simple Storage Service)

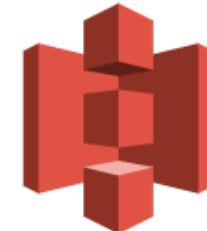
- Most popular, very flexible & inexpensive storage service
- Store large objects using a **key-value** approach
- Also called **Object Storage**
- Provides REST API to access and modify objects
- Provides **unlimited storage**:
 - (S3 storage class) 99.99% availability & (11 9's - 99.999999999) durability
 - 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 - Objects and Buckets

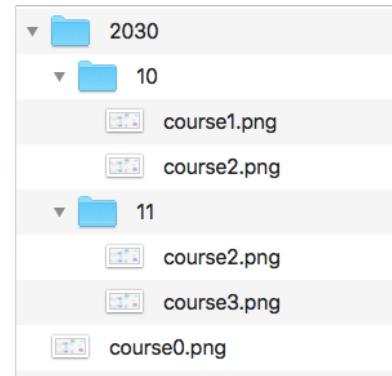
- Amazon S3 is a **global service**. NOT associated with a region.
 - HOWEVER a bucket is created in a specific AWS region
- Objects are stored in buckets:
 - Bucket names are **globally unique** and used as part of object URLs
 - Can contain ONLY lower case letters, numbers, hyphens and periods
 - Unlimited 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**
- Amazon S3 Versioning(**Optional - Enabled at bucket level**):
 - Protects against **accidental deletion**
 - You can enable versioning on existing bucket (Old objects => version null)
 - You cannot turn off versioning on a versioned bucket
 - You can only **suspend** versioning



Amazon S3

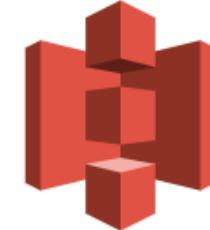
Amazon S3 - Prefix

- 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

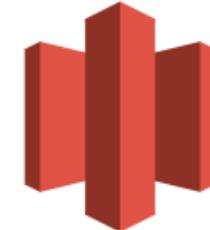


Amazon S3 Storage Classes - Introduction

- 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

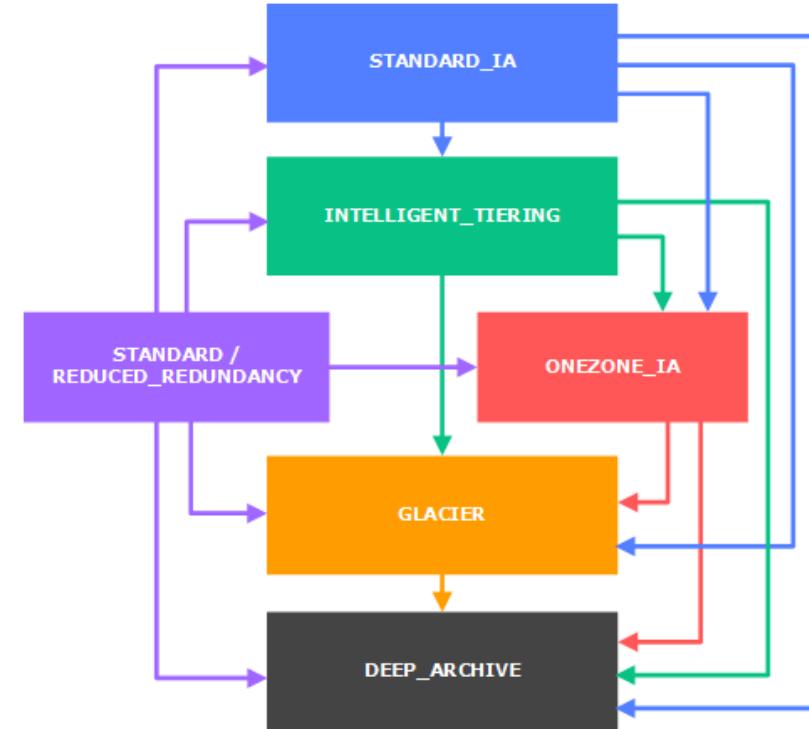
Storage Class	Scenario	AZs
Standard	Frequently accessed data	>=3
Standard-IA	Long-lived, infrequently accessed data (backups for disaster recovery)	>=3
One Zone-IA	Long-lived, infrequently accessed, non-critical data (Easily re-creatable data - thumbnails for images)	1
Intelligent-Tiering	Long-lived data with changing or unknown access patterns	>=3
Glacier	Archive data with retrieval times ranging from minutes to an hour	>=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 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 - Remember

- 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
- Tags : **Key-value pairs** associated with S3 objects - Environment=Dev, Classification=Secure, Project=A etc
 - Used for **automation, security (policies), cost tracking** etc
 - Can be used in creating **lifecycle policies**
- Event Notifications: Configure **notifications** when **certain events** happen
 - **Event Sources:** New object created events, Object removal events , Reduced Redundancy Storage (RRS) object lost events, Replication events etc.
 - **Event Destinations:** Amazon SNS topic, Amazon SQS queue, AWS Lambda function etc.

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

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 - Security

- **Presigned URL** : Grant time-limited permission (few hours to 7 days) to download objects
 - Avoid web site scraping and unintended access
 - 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
- You can configure these at **individual object level** (overriding bucket level configuration):
 - Encryption
 - Objects ACLs
 - Storage class

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 Scenarios - Costs

Scenario	Description
Important pricing elements	Cost of Storage (per GB), (If Applicable) Retrieval Charge (per GB), Monthly tiering fee (Only for Intelligent Tiering), Data transfer fee
Is Data Transfer Free?	Nope. Some of free things include Data transfer into Amazon S3, From Amazon S3 to Amazon CloudFront, From Amazon S3 to services in the same region
Reduce Costs	Use proper storage classes. Configure lifecycle management.
Analyze storage access patterns and decide the right storage class	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 Scenarios - Performance

In 28
Minutes

Scenario	Solution
Improve S3 bucket performance	Use Prefixes . 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 Multipart Upload API . Advantages: 1. Quick recovery from any network issues 2. Pause and resume object uploads 3. Begin an upload before you know the final object size. Recommended for files >100 MB and mandatory for files >4 GB
Get part of the object	Use Byte-Range Fetches - Range HTTP header in GET Object request Recommended: GET them in the same part sizes used in multipart upload
Is this recommended: EC2 (Region A) <-> S3 bucket (Region B)	No. Same region recommended . Reduce network latency and data transfer costs
Faster Data Transfer to S3	Consider Transfer acceleration - Enable fast, easy and secure transfers of files to and from your bucket

Amazon S3 Scenarios - Features

Scenario	Solution
Make user pay for S3 requests and data transfer	Requester pays - The requester (instead of the bucket owner) will pay for requests and data transfer.
Create an inventory of objects in S3 bucket	Use S3 inventory report
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	Generate S3 inventory report Perform S3 Batch Operations using the report
Need S3 Bucket (or Object) Access Logs	Enable S3 Server Access Logs (default: off). Configure the bucket to use and a prefix (logs/).

Amazon Glacier

Amazon Glacier

- In addition to existing as a S3 Storage Class, Amazon 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	Amazon 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

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": "*", //["s3:Get*", "s3>List*"],  
            "Resource": "*" //arn:aws:s3:::mybucket/somefolder/*  
        }  
    ]  
}
```

- 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
 - Give Read Only Access to S3 buckets - "Action": ["s3:Get*", "s3>List*"]

IAM Scenarios

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	

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
- What happens in the background?
 - **Instance Profile:** A Container (A Box) for an IAM role
 - Used to pass role information to an EC2 instance
 - Creation:
 - 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

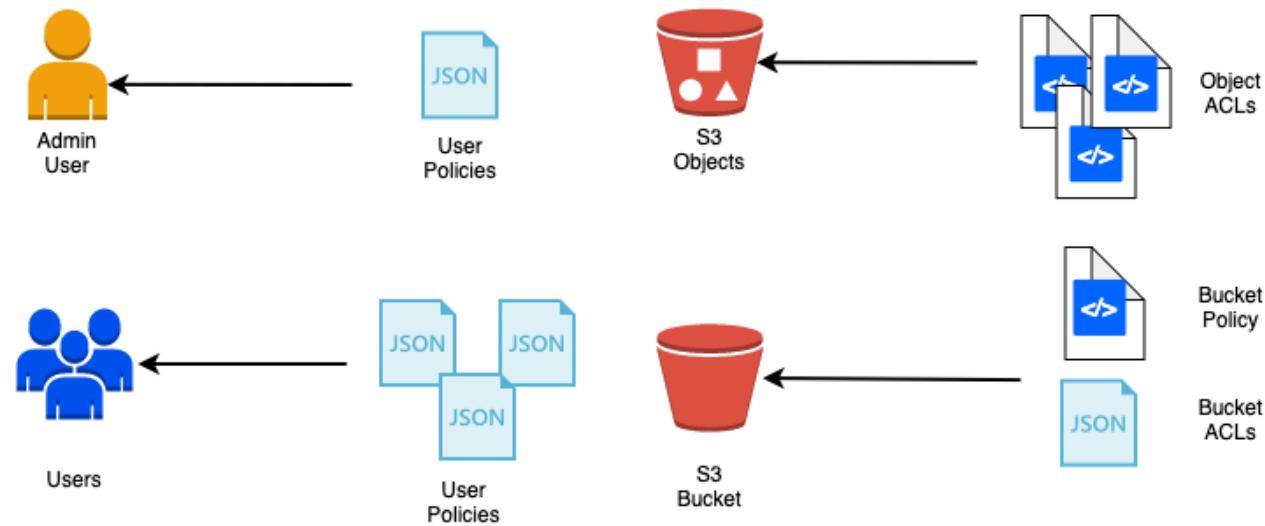


IAM Role Use case 2: Cross Account Access

- 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!



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 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}

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}

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"}

IAM Scenario Questions

In 28
Minutes

Scenario	Solution
How to rotate access keys without causing problems?	Create new access key Use new access key in all apps Disable original access key Test and verify Delete original access key
How are multiple permissions resolved in IAM Policy?	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
Which region are IAM users created in ?	IAM Users are global entities . Can use AWS services in any geographic region
What is the difference between IAM user, Federated user and Web identity federation user?	IAM users - created and maintained in your AWS account Federated users - External Users outside AWS Web identity federation users - Amazon Cognito, Amazon, Google, or any OpenID Connect-compatible provider Accounts

Authentication with IAM - Remember

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

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
 - (Hardware device - Gemalto, Virtual device - An app on a smart phone)
- **Roles** – Use IAM roles for Amazon EC2 instances
- **Sharing** – Use IAM roles to share access
- **Rotate** – Rotate security credentials regularly
- **Root** – Reduce or remove use of root

Data Encryption

KMS and Cloud HSM

- Generate, store, use and replace your keys(symmetric & asymmetric)
- **KMS: Multi-tenant Key Management Service**
 - KMS integrates with all storage and database services in AWS
 - Define key usage permissions (including **cross account** access)
 - **Automatically rotate master keys** once a year
 - **Schedule key deletion** to verify if the key is used
 - Mandatory minimum wait period of 7 days (max-30 days)
- **CloudHSM: Dedicated single-tenant HSM for regulatory compliance**
 - (Remember) AWS KMS is a Multi-tenant service
 - **AWS CANNOT access your encryption master keys in CloudHSM**
 - (**Recommendation**) Be ultra safe with your keys. Use two or more HSMs in separate AZs.
 - 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
 - **Use Cases:** (Web servers) Offload SSL processing, Certificate Authority etc



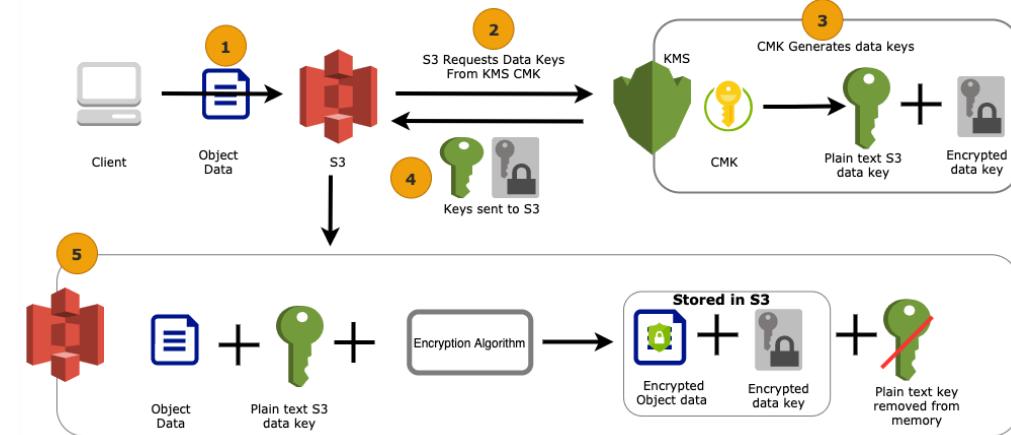
AWS KMS



Cloud HSM

KMS - How does encryption and decryption happen?

- Customer Master Key (CMK) created in KMS and mapped to S3
- Encryption Steps:
 - Data sent to S3
 - S3 receives data keys from KMS
 - S3 encrypts data
 - Stores encrypted data & data key
- Decryption Steps:
 - S3 sends encrypted data key to KMS
 - KMS decrypts using CMK. Returns data key.
 - S3 uses plain text data key to decrypt data
 - Remove data key from memory asap
- Also called Envelope Encryption



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

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)
- **ListKeyPolicies, GetKeyPolicy, PutKeyPolicy** - Play with Key Policies
- **ScheduleKeyDeletion, CancelKeyDeletion** - Schedule and cancel deletion of keys



AWS KMS

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!

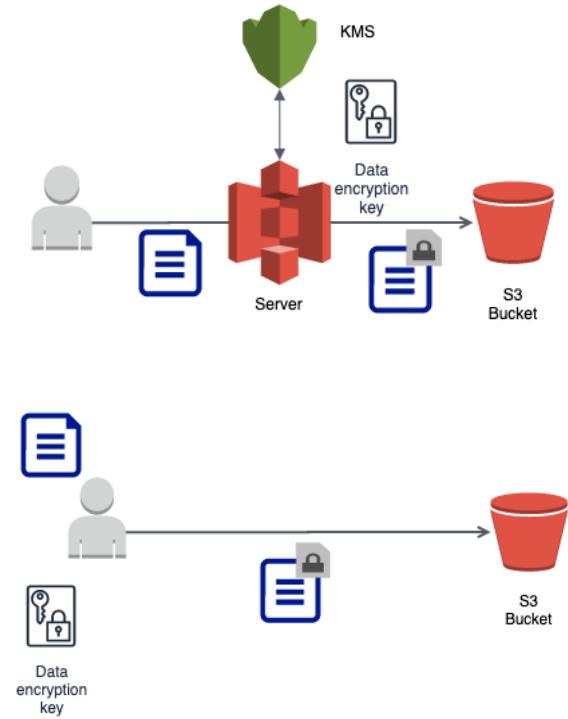
- **What could go wrong?**

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



Server Side vs Client Side Encryption - Amazon S3

- **Server Side Encryption:** S3 <-> KMS to encrypt data
 - **SSE-S3:** AWS S3 manages its own keys (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 key with request (HTTPS mandatory)
 - S3 performs encryption and decryption without storing the key
 - **Use HTTPS endpoints** (secure data in transit)
 - All AWS services (including S3) provides HTTPS endpoints
- **Client Side Encryption:** Client manages encryption
 - Client sends encrypted data to AWS service
 - AWS will not be aware of master key or data key
 - AWS service stores data as is
 - Use a client library (Amazon S3 Encryption Client)



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.

KMS - Remember

- KMS encrypts small pieces of data (usually data keys) MAX - 4 KB
 - Use Envelope Encryption for larger objects (CMK never leaves KMS)
 - Generate a data key (plain-text and encrypted) from KMS (GenerateDataKey)
 - Use data key to perform encryption/decryption on the object (within the service or client-side)
- You can assign an **encryption context** with cryptographic operations
 - (TIP) If encryption context is different, decryption will NOT succeed
- 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 or Retry with Exponential Backoff
- Usage of KMS CMKs can be tracked in **CloudTrail**
- Key policies control access to CMKs (incl. cross account access)
- Use **AWS Encryption SDK** to interact with KMS(Provides Data Key Caching)

Networking in AWS

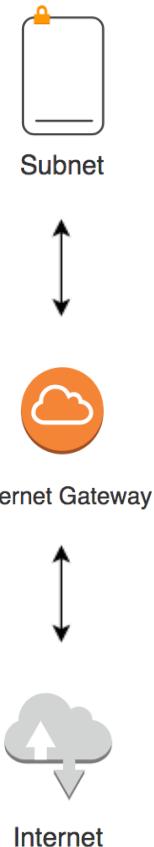
Amazon VPC (Virtual Private Cloud) and Subnets

- 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 AWS resources **in a VPC**
 - Secure resources from unauthorized access AND
 - Enable secure communication between your cloud resources
 - Each VPC is created in a Region
- **Subnet - Separate public resources from private resources in a VPC**
 - **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 Subnet is created in an Availability Zone
 - VPC - us-east-1 => Subnets - AZs us-east-1a or us-east-1b or ..



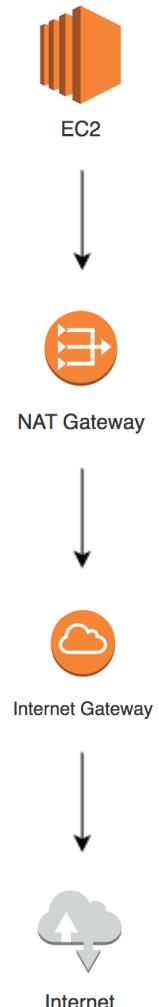
Public Subnet vs Private Subnet

- **Public Subnet:** Communication allowed - Internet to Subnet
- An **Internet Gateway** enables internet communication for public subnets
 - **Public Subnet:** Subnet having a route to an internet gateway
 - **Private Subnet:** Subnet **DOES NOT** have route to an internet gateway



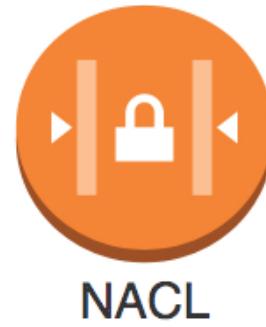
Private Subnet - NAT Devices - Download Patches

- Allow instances in a private subnet to download software patches while denying inbound traffic from internet
- Three Options:
 - **NAT Instance:** Install a EC2 instance with specific NAT AMI and configure as a gateway
 - You are taking complete responsibility of availability
 - **NAT Gateway:** Managed Service (PREFERRED - No maintenance, more availability & high bandwidth)
 - **Egress-Only Internet Gateways:** For IPv6 subnets (NAT Gateway supports IPv4 ONLY)



Network Access Control List

- Security groups control traffic to a specific resource in a subnet
- NACL provides **stateless firewall** at subnet level
 - Stop traffic from even entering the subnet
- 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
Level	Assigned to a specific instance(s)/resource(s)	Configured for a subnet. Applies to traffic to all instances in a subnet.
Rules	Allow rules only	Both allow and deny rules

VPC - Important Concepts

- **VPC Peering** - Connect VPCs from same or different AWS accounts (across regions)
 - Allows private communication between the connected VPCs
 - Peering uses a request/accept protocol (Owner of requesting VPC sends a request)
 - Peering is not transitive.
- **VPC Endpoint** - Securely connect your VPC to another service
 - Gateway endpoint: Securely connect to Amazon S3 and DynamoDB
 - Interface endpoint: Securely connect to AWS services EXCEPT FOR Amazon S3 and DynamoDB
 - Powered by PrivateLink (keeps network traffic within AWS network)
 - (Avoid DDoS & MTM attacks) Traffic does NOT go thru internet
 - (Simple) Does NOT need Internet Gateway, VPN or NAT

VPC Flow Logs

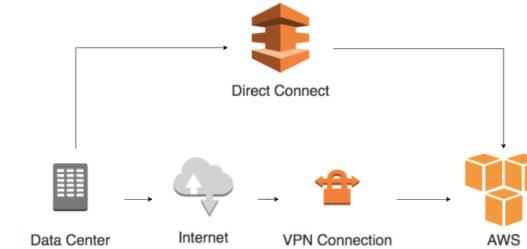
- Monitor network traffic
- Troubleshoot connectivity issues (NACL and/or security groups misconfiguration)
- Capture traffic going in and out of your VPC (network interfaces)
- Can be created for
 - a VPC
 - a subnet
- Publish logs to Amazon CloudWatch Logs or Amazon S3
- Flow log records contain ACCEPT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



VPC Flow Logs

AWS and On-Premises - Overview

- **AWS Managed VPN:** Tunnels from VPC to on premises
 - 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 (DX):** Private dedicated network connection to on premises
 - (Advantage) Reduce your (ISP) bandwidth costs
 - (Advantage) Consistent Network performance (private network)
 - Connection options: Dedicated (1 Gbps or 10 Gbps) or Hosted (Shared 50Mbps to 10 Gbps)
 - (Caution) Establishing DC connection takes a month
 - (Caution) Establish a redundant DC for maximum reliability
 - (Caution) Data is NOT encrypted (Private Connection ONLY)



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 another VPC
- **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

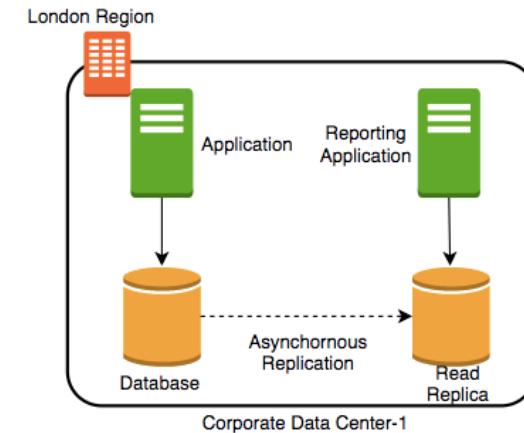
Databases in AWS

Databases - Overview

Database Type	AWS Service	Description
Relational OLTP databases	Amazon RDS	Transactional usecases needing predefined schema and very strong transactional capabilities
Relational OLAP databases	Amazon Redshift	Datewarehouse, reporting, analytics & intelligence apps
Document & Key Databases	Amazon DynamoDB	Apps needing quickly evolving semi structured data (schema-less) Terabytes of data with millisecond responses for millions of TPS Content management, catalogs, user profiles, shopping carts, session stores and gaming applications
Graph Databases	Amazon Neptune	Store and navigate data with complex relationships Social Networking Data (Twitter, Facebook), Fraud Detection
In memory store/caches	Amazon ElastiCache	Applications needing microsecond responses Redis - persistent data Memcached - simple caches

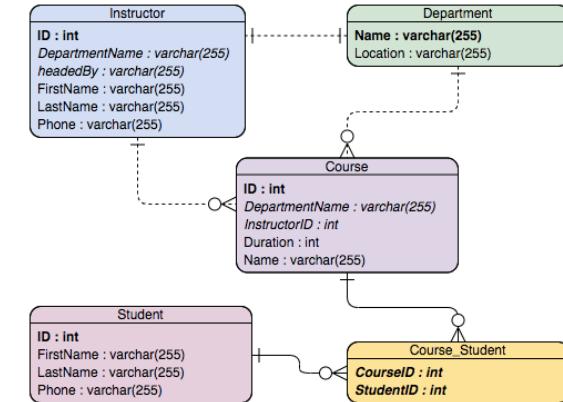
Consistency

- How to ensure that data in multiple database instances 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 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 : Facebook status messages, Twitter tweets etc
- **Read-after-Write consistency** - Inserts are immediately available. Updates and deletes are eventually consistent (Ex: Amazon S3)



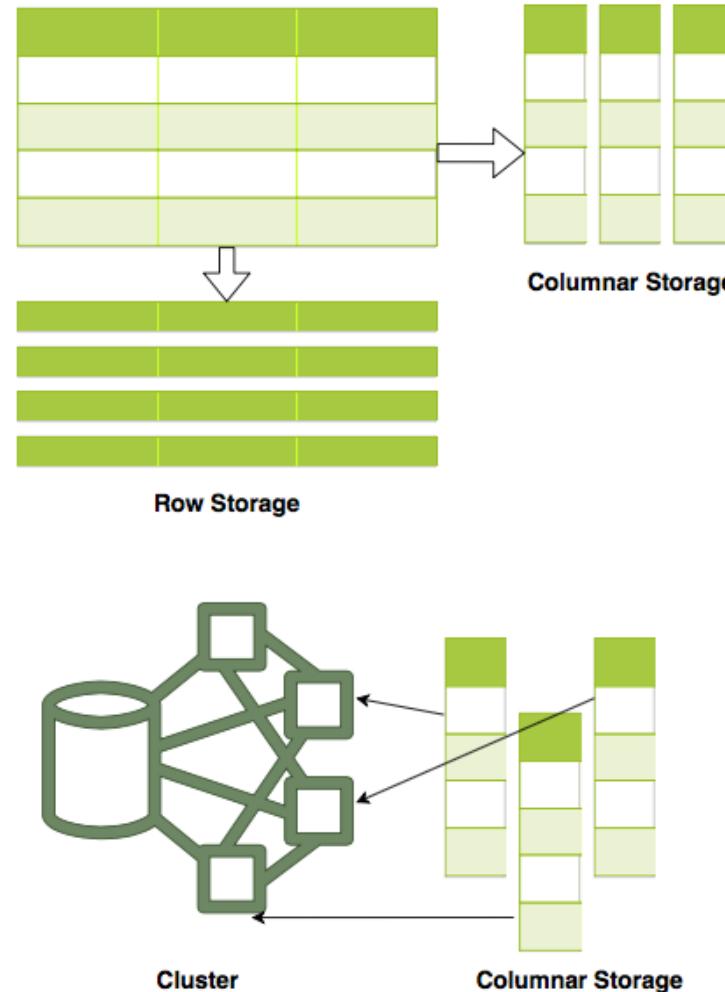
Relational Databases

- Predefined schema with tables and relationships
- Very strong transactional capabilities
- OLTP (Online Transaction Processing) Databases:
 - Databases to support large number of small transactions
 - AWS Managed Service: Amazon RDS (Amazon Aurora, PostgreSQL, MySQL, MariaDB (Enhanced MySQL), Oracle Database, and SQL Server)
- OLAP (Online Analytics Processing) Databases:
 - Analyze petabytes of data (Ex : Reporting applications, Datawarehouses)
 - Data is consolidated from multiple (transactional) databases
 - AWS Managed Service: - Amazon Redshift
 - Petabyte-scale distributed data ware house based on PostgreSQL



Relational Databases - OLAP vs OLTP

- OLAP and OLTP use similar data structures
- BUT different approach to storing data
- 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



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)

- **Managed relational database service for OLTP use cases**
 - Manage setup, backup, scaling, replication and patching of your relational databases
 - Supports Amazon Aurora, PostgreSQL, MySQL (InnoDB storage engine full supported), MariaDB (Enhanced MySQL), Oracle Database and Microsoft SQL Server
- **Features:**
 - Multi-AZ deployment (standby in another AZ)
 - Read replicas (Same AZ or Multi AZ (Availability+) or Cross Region(Availability++))
 - Storage auto scaling (up to a configured limit)
 - Automated backups (restore to point in time)
 - Manual snapshots

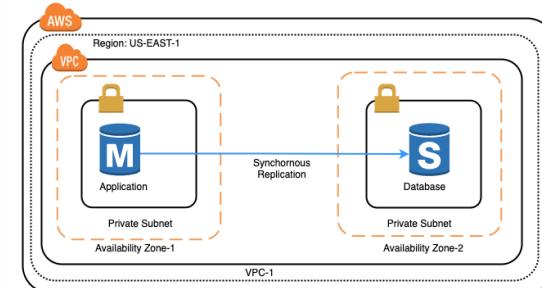


Amazon RDS (Relational Database Service) - Remember

- AWS is responsible for
 - Availability, Scaling (according to your configuration), Durability, Maintenance (patches) and 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)

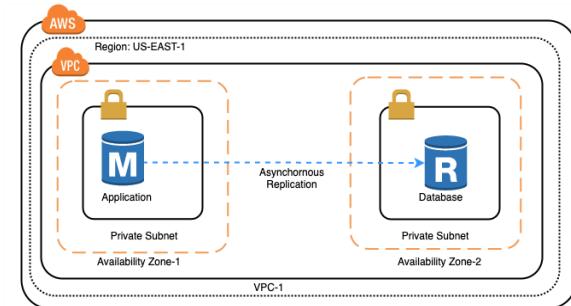
Multi-AZ Deployments

- Create standby in diff. AZ (**Synchronous replication**)
 - Enhances durability, availability and fault tolerance
 - Makes **maintenance** easy
 - Perform maintenance (patches) on standby
 - Promote standby to primary and Perform maintenance on (old) primary
 - **Avoid I/O suspension** when data is backed up
 - Snapshots are taken from standby
 - 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:
 - CNAME record flipped to standby
 - Database performance issues will NOT cause a failover
 - (Good Practice) Use DNS name of database in applications configuration



Read Replicas

- Support **read-heavy database workloads**
 - Reporting, data warehousing, analytics etc.
 - Your apps can connect directly to Read Replicas
- Can be in same or different AZ or different Region
- **Asynchronous replication** (Eventual consistency)
 - For higher consistency, read from master
 - Reduce replication lag - Increase CPU and storage
- Features:
 - Create read replica(s) of a read replica
 - Need to be **explicitly deleted** (Not deleted when DB is deleted)
 - (Mandatory) Enable auto backups before creating read replicas
 - Set Backup Retention period to a value other than 0

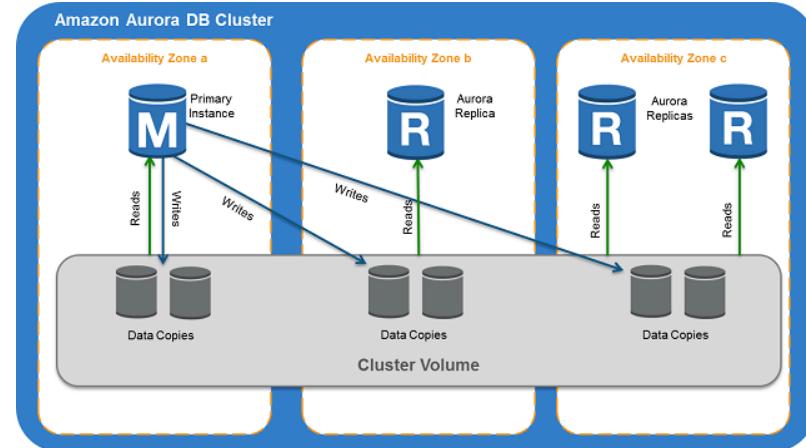


Multi-AZ vs Multi-Region vs Read replicas

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>

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
What are retained when you delete a RDS database instance?	All automatic backups are deleted All manual snapshots are retained (until explicit deletion) (Optional) Take a final snapshot
How do you reduce global latency and improve disaster recovery?	Use multi region read replicas
Efficiently manage database connections	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

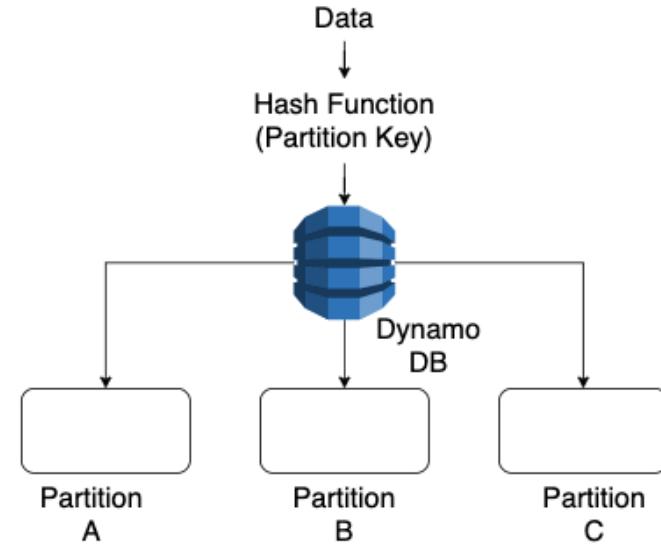
DynamoDB Tables

- 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"  
  }  
}
```

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:** Same partition key as primary key (diff. sort key)
 - Example:
 - Attributes: CustomerId (partition key) + OrderId (sort key), OrderCreationDt, ProductDetails, OrderStatus
 - LSI: [CustomerId (partition key) + OrderCreationDt] OR [CustomerId (partition key) + OrderStatus]
 - Defined at table creation (Modifications NOT allowed)
- **Global secondary index:** Partition and sort key CAN be diff. from primary key
 - Example:
 - Attributes: storeCode (Primary Key), city, state, country, address
 - GSI: state or city or country
 - Can be added, modified & removed later
 - Stored separately (Separate RCU and WCU) from the main table
 - (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

- (**DEFAULT**) **Eventually Consistent Reads** : Might NOT get latest data
 - If tried after few seconds, you will get the latest data
- **Strongly Consistent Reads**: Get the most up-to-date data
 - Reflects updates from all the previous successful write operations
 - Set `ConsistentRead` to true
 - Disadvantages:
 - Returns 500 error in case of network delay
 - May have higher latency
 - Not supported on Global Secondary Indexes
 - Uses more throughput capacity units
- **Supports transactions** (`TransactWriteItems`, `TransactGetItems`)
 - All-or-nothing changes to multiple items both within and across tables
 - Include `PutItem`, `UpdateItem` and `DeleteItem` operations
 - 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)
 - Billed for the provisioned capacity whether its used 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

RCU & WCU - 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
 - $15/4$ is 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
 - $15/4$ is 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

Important DynamoDB API

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

DynamoDB API 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"}}'
```

DynamoDB API - 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

DynamoDB API - Conditional vs Projection vs Filter 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"}}'
aws dynamodb scan --table-name MyTodos --projection-expression "username"
aws dynamodb scan --table-name MyTodos --filter-expression "begins_with(username,:username")
```

- **Conditional Expression:** Update/delete an item only when a condition is true
 - Enables you to check for consistency of update/delete
 - Error message if check fails - ConditionalCheckFailedException - An error occurred with DeleteItem operation
- **Projection Expression:** Retrieve selected attributes in an API Operation
 - Combine with expression-attribute-names to project keywords (Begins with #)
- **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"}}

DynamoDB API - Pagination

```
aws dynamodb scan --table-name MyTodos --max-items 100
```

```
aws dynamodb scan --table-name MyTodos --max-items 100 \
--starting-token eyJFeGNsdXNpdmVTdGFydEtIeSI6IG51bGwsICJib3RvX3RydW5jYXRlx2Ftb3VudCI6Il
```

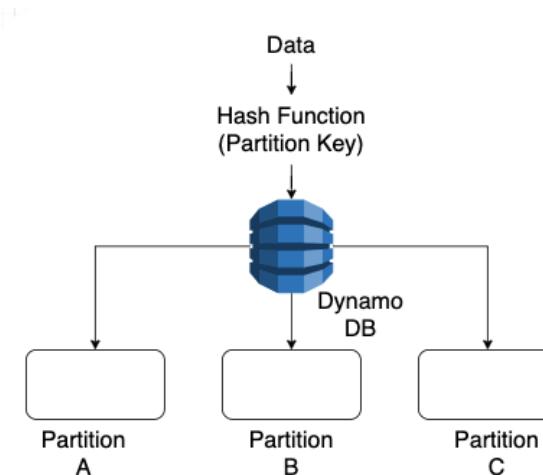
- **--max-items**: total number of items to return in the command's output. If there are more than max items, a token is provided in the output.
- **--starting-token**: A token to specify where to start paginating. This is the NextToken from a previously truncated response.
- **--page-size** : Size of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

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)
 - 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.

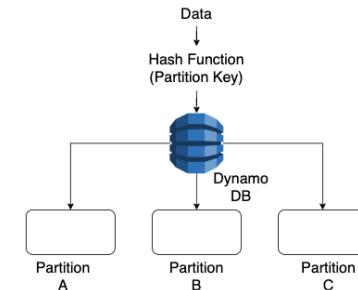
Designing DynamoDB Tables

- Designing Partition Keys:
 - Provisioned capacity evenly divided among partitions
 - Each partition - Max RCU 3000, Max WCU 1000
 - For good performance, distribute items evenly across partitions
 - Prefer attributes that have mostly unique values
 - (NOT RECOMMENDED) Client Type: Only 4 types of clients
 - (NOT RECOMMENDED) Creation date rounded to day, hour, or minute
 - (RECOMMENDED) Product ID which is unique for each Item
 - (RECOMMENDED) Write Sharding with Random or Calculated Suffix
- Scenario: Handling Huge Volumes of Time Series Data
 - Millions of records with date timestamp - recent events are accessed frequently. Events older than two days rarely accessed.
 - Option 1: Create a new table for each period
 - As tables get older, reduce WCU and RCU
 - Option 2: Expire Old Records : Configure TTL Attribute on table
 - Records are automatically deleted on expiry. No WCU consumed.



DynamoDB Best Practices

- Different design approach (vs 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:
 - Reduce page size
 - (IF NEEDED) Isolate scan operations - Duplicate 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

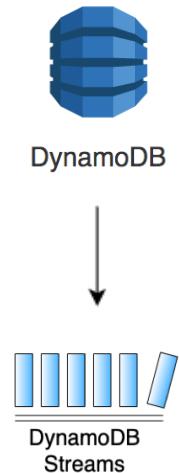


DynamoDB - Remember

- 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 supports Optimistic Locking:
 - Protect your writes by checking the version before updates using conditional expression or use SDK options (@DynamoDBVersionAttribute annotation in Java)

DynamoDB Streams

- Time ordered sequence of item modifications (Stored upto 24 hrs)
 - Create, update or delete of an item => A stream record is written in **near real time**
 - StreamViewType decides the details captured
 - KEYS_ONLY, NEW_IMAGE, OLD_IMAGE, NEW_AND_OLD_IMAGES
- 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 diff. clients for DynamoDB & 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 from RDS or MongoDB - AWS Database Migration Service
- (Feature) Enable point-in-time recovery (max 35 days)
- Use Time to Live (TTL) to automatically expire items
- IAM and Encryption:
 - Server-side encryption in integration with keys from KMS
 - Always enabled - encrypts tables, DynamoDB streams, and backups
 - Client-side encryption with DynamoDB Encryption Client (Integrate with KMS)
 - Use IAM roles to provide EC2 instances or AWS services access to DynamoDB tables
 - Predefined policies available (AmazonDynamoDBReadOnlyAccess, AmazonDynamoDBFullAccess ..)
 - 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 - Scenario Questions

Scenario	Solution
<p>Your Provisioned RCU is not being completely utilized. How can you make scans on large tables (>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

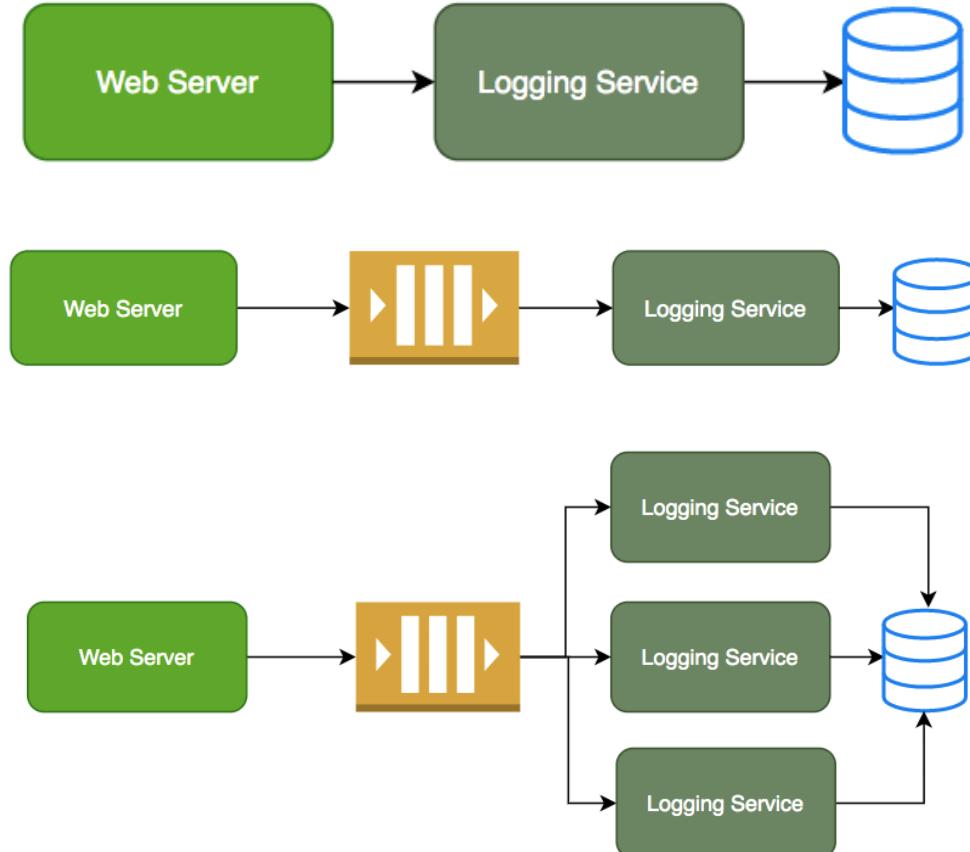
DynamoDB - Scenario Questions - 2

Scenario	Solution
Order Date is being used as partition key causing throttling in high load periods	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!
You want to store large images (0 to 100 MB)	Store Images in S3 Store reference to S3 object (URL, metadata and/or keys) in DynamoDB

Decoupling Applications with SQS and SNS

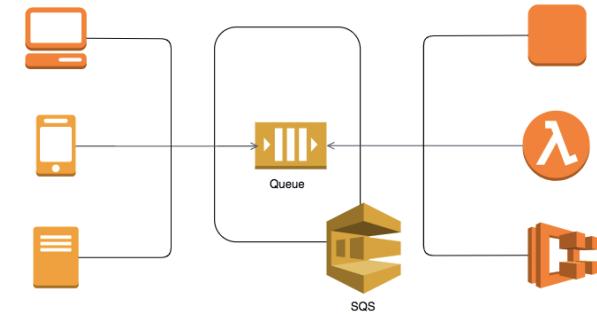
Synchronous vs Asynchronous Communication

- Synchronous Communication:
 - What if your logging service goes down?
 - Will your applications go down too?
 - What if there is high load?
 - Log Service unable to handle and goes down
- Asynchronous Communication:
 - Create a queue or a topic
 - Your applications put the logs on the queue
 - Picked up when the logging service is ready
 - Good example of decoupling!
 - (Possible) Multiple logging service instances reading from the queue!



Asynchronous Communication - Pull Model - SQS

- Producers put messages. Consumers poll on queue.
 - Only one of the consumers will successfully process a message
- **Advantages:**
 - 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
- **Features:**
 - 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

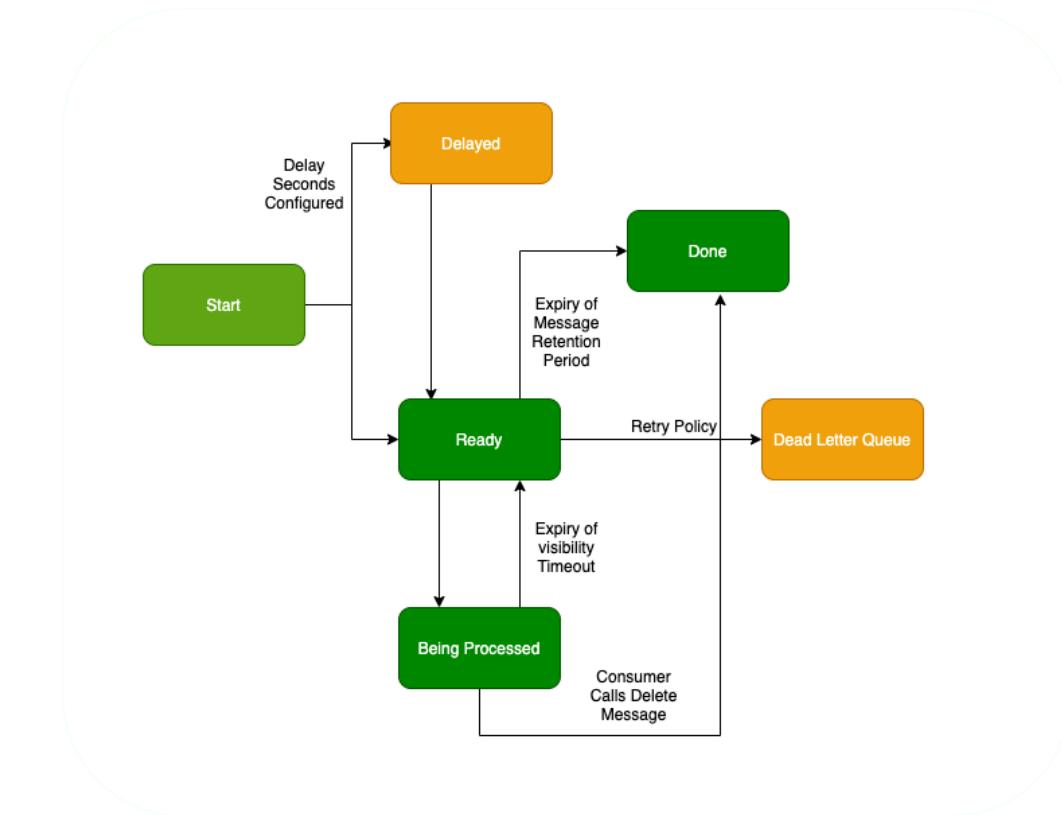
- Standard Queue
 - Unlimited throughput
 - BUT NO guarantee of ordering (Best-Effort Ordering)
 - and NO guarantee of exactly-once processing
 - Guarantees at-least-once delivery (some messages can be processed twice)
- FIFO (first-in-first-out) Queue
 - First-In-First-out Delivery
 - Exactly-Once Processing
 - 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
 - Calls delete message (using receipt handle XYZ)
 - Message is removed from the queue



SQS Queue - Important configuration

Configuration	Description
Visibility timeout	<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>
DelaySeconds	<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>
Message retention period	<p>Maximum period a message can be on the queue</p> <p>Default - 4 days, Min - 60 seconds, Max - 14 days</p>
MaxReceiveCount	Maximum number of failures in processing a message

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

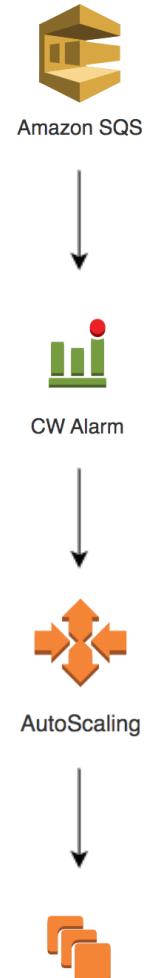
APIs	Description
CreateQueue, DeleteQueue	Create or Delete a Queue
GetQueueAttributes, SetQueueAttributes	Get or Set Attributes on a Queue (DelaySeconds, MessageRetentionPeriod, VisibilityTimeout etc)
SendMessage	Send a message to the Queue
SendMessageBatch	SendMessageBatch delivers up to ten messages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

- **Auto Scaling**
 - Use target tracking scaling policy
 - Use a SQS metric like ApproximateNumberOfMessages
- **Security**
 - 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



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

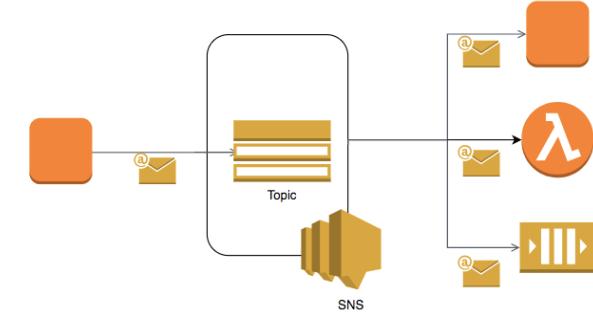
SQS - Scenarios - 2

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

Asynchronous Communication - Push Model - SNS

- How does it work (Publish-Subscribe(pub-sub))?
 - 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
 - (Advantage) Decoupling: Producers don't care about Consumers
 - (Advantage) Availability: Producer up even if subscriber is down
- Use Cases : Monitoring Apps, workflow systems
- Provides mobile and enterprise messaging services
 - Push notifications to Apple, Android, FireOS, Windows devices
 - Send SMS to mobile users and Emails
- REMEMBER : SNS does not need SQS or a Queue



Handling Data Streams

Streaming Data - Simple Solutions

- How to process continuous streaming data from application logs or user actions from social media applications?
 - Characteristics of streaming data: Continuously generated, Small pieces of data which are Sequenced - mostly associated with time
- Option: **S3 Notifications**
 - Send notifications to SNS, SQS, trigger lambdas on
 - creation, deletion or update of an S3 object
 - Setup at bucket level (Use prefix and suffix)
 - Cost efficient for simple use cases (S3 notification -> Lambda)
 - Almost negligible cost (storage for file + invocation)
- Option: **DynamoDB Streams** (DynamoDB > Stream > Lambda)
 - Events from DynamoDB buffered in a stream (real-time sequenced)
 - Can be enabled or disabled
 - Use case - Send email when user registers

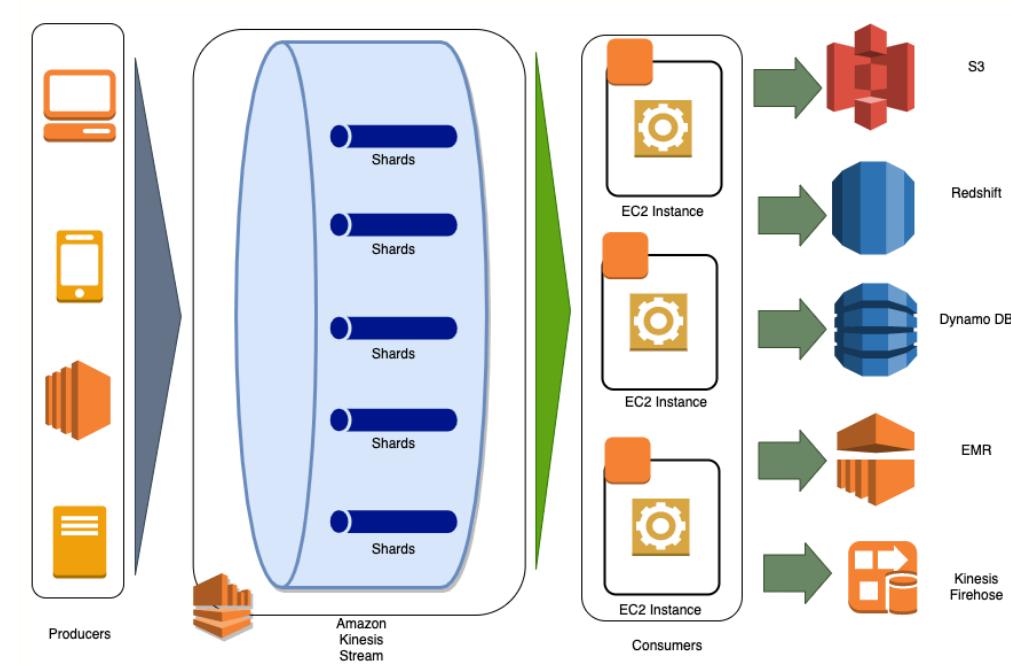


Amazon Kinesis

- Handle streaming data (NOT recommended for ETL Batch Jobs)
- **Amazon Kinesis Data Streams** (Alternative for Kafka)
 - Limitless Real time stream processing with Sub second latency
 - Supports multiple clients(Each client can track their stream position)
 - Retain and replay data (max 7 days & default 1 day)
- **Amazon Kinesis Firehose:** Data ingestion for streaming data
 - Receive > Process (transform - Lambda, compress, encrypt) > Store (S3, Elasticsearch, Redshift and Splunk)
 - Use existing analytics tools based on S3, Redshift and Elasticsearch
- **Amazon Kinesis Analytics:** Continuously analyze streaming data
 - Run SQL queries and write Java apps (find active users in last 5 minutes)
- **Amazon Kinesis Video Streams:** Monitor video streams from web-cams
 - Integrate with machine learning to get intelligence (Examples: traffic lights, shopping malls)

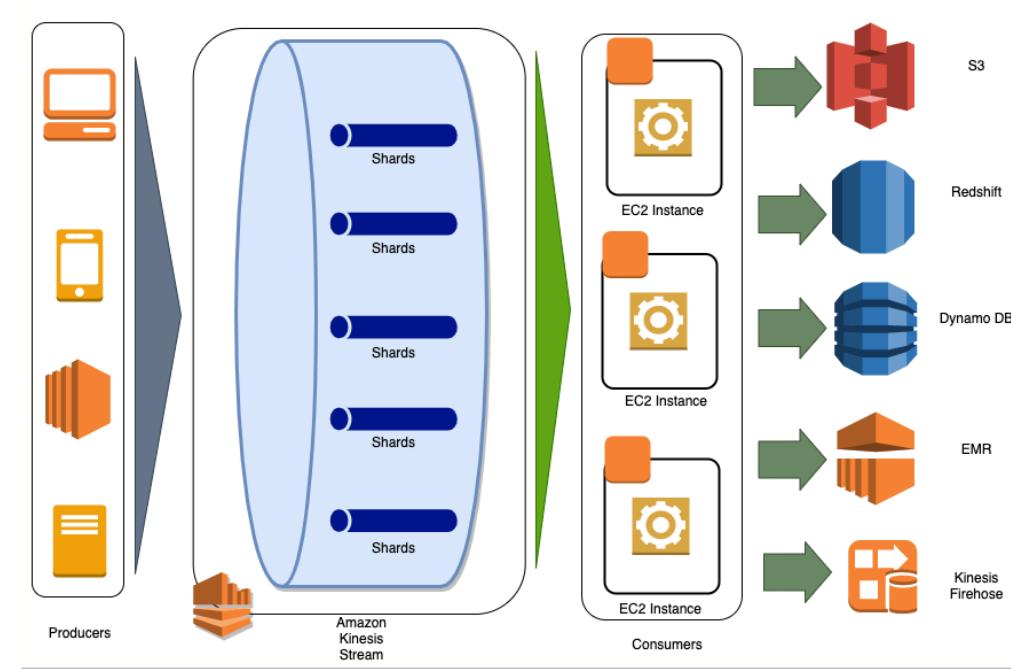
Kinesis Streams - Hierarchy

- **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



Amazon Kinesis Data Streams - Integrations

- Manage Data Streams using Kinesis Data Streams APIs
- Use application integrations to generate streams:
 - Toolkits : AWS SDK, AWS Mobile SDK, Kinesis Agent
 - Service Integrations : AWS IOT, CloudWatch Events and Logs
- Write Kinesis Client Library (KCL) Consumer Applications:
 - Each shard bound to one KCL instance.
 - However: KCL instance can read from many shards
 - (Recommended) Have same number of consumer instances as shards



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.

Kinesis Streams - Scenario Questions

Scenario	Solution
What are the recommended use cases for Kinesis Streams?	<ol style="list-style-type: none">1. A continuous stream of analytics from a web application2. Multiple consumers consuming from a stream of data3. Consume records in the same order a few hours later
Develop Producer Applications for Kinesis Streams	Use Amazon Kinesis Producer Library (KPL) or Amazon Kinesis Agent (pre-built Java application)
Build Consumer Applications for Kinesis Streams	Use Amazon Kinesis Client Library (KCL). Supports Java, Python, Ruby, Node.js and .NET.
ProvisionedThroughputExceededException happens infrequently	Retry with exponential backoff
Increase throughput of an Amazon Kinesis Data Stream	Resharding - Increase the number of shards.

EC2 Fundamentals

EC2(Elastic Compute Cloud)

- **EC2 instances** - Virtual servers in AWS (billed by second)
- **EC2 service** - Provision EC2 instances or virtual servers
- **Features:**
 - 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



EC2 Amazon Machine Image - AMI - Choose OS and Software

In 28
Minutes

- AMI: What OS 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.
- AMIs contain:
 - Root volume block storage (OS and applications)
 - Block device mappings for non-root volumes
- Configure launch permissions on an AMI(Who can use the AMI?)
 - 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 AMI

EC2 Instance Types - Choose Hardware

- Optimized combination of **compute(CPU, GPU), memory, disk (storage) and networking** for specific workloads
 - **m (m4, m5, m6)** - General Purpose. Balance of compute, memory, and networking.
 - **t (t2, t3, t3a)** - **Burstable performance instances** (accumulate CPU credits when inactive). Workloads with spikes : web servers, developer environments and small databases.
 - **c (c4, c5, c5n)** - **Compute optimized..** Batch processing, high performance computing (HPC)
 - **r (r4, r5, r5a, r5n)** - **Memory (RAM) optimized.** Memory caches and in-memory databases.
 - **i (i3, d2)** - **Storage (I/O) optimized.** NoSQL databases and data warehousing.
 - **g (g3, g4)** - **GPU optimized.** FP Calculations, graphics processing, or video compression.
 - **t2.micro:**
 - **t** - Instance Family
 - 2 - generation. Improvements with each generation.
 - **micro** - size. (*nano < micro < small < medium < large < xlarge <*)
- Size increases => compute, memory and networking capabilities increase

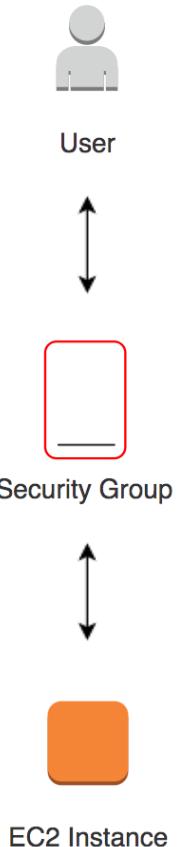
EC2 IP Addresses



- Quick Review: **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
- Public IP address **can be enabled** for EC2 instances in public subnet
- (Remember) When you stop an EC2 instance, public IP address is lost
- **Elastic IP:** Quick & Dirty approach to get a **static public IP address**
 - Can be switched to another instance **in same region**
 - Elastic IP **remains attached** even if you stop the instance(Manually detach)
 - Remember : Elastic IP is billed for when you are NOT using it!

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 are **default deny**. NO RULES => NO ACCESS.
 - 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 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
- 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



EC2 Basics - Userdata, Launch Templates & Customized AMI

In 28
Minutes

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "Welcome to in28minutes" > /var/www/html/index.html
```

- **Bootstrapping:** Install patches or software when an EC2 instance is launched
- Use **userdata** to bootstrap(<http://169.254.169.254/latest/user-data/>)
- **Launch Template:** Avoid specifying EC2 instance details (AMI ID, instance type, user data and network settings) **every time** you launch an instance
 - Use Spot instances and Spot fleets as well
- **Customized AMI:** AMI created by you
 - Faster Boot Up - Avoid installing OS patches and software launch of EC2 instances (**Prefer** using Customized AMI to userdata)
 - **Hardening an Image** - Customize EC2 images to your corporate security standards

EC2 Security - Key Pairs

- EC2 uses **public key cryptography** to protect login credentials
- **Key pair** - public key and a private key
 - Public key is stored in EC2 instance
- 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`



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 Pricing Models Overview

Pricing Model	Description
On Demand	Request when you want it. Flexible and Most Expensive. Immediate (mission critical) workloads (web applications/batch programs).
Spot	Cheapest (upto 90% off). Quote the maximum price. Terminated with 2 minute notice. Cost sensitive, Fault tolerant, Non immediate workloads.
Reserved	Reserve ahead of time. Upto 75% off. 1 or 3 years reservation. Scheduled: Reserve for specific time period in a day. (5% to 10% off) No Upfront or Partial Upfront or All Upfront Payments
Savings Plans	Commit spending \$X per hour on (EC2 or AWS Fargate or Lambda). Upto 66% off. Lot of flexibility. 1 or 3 years reservation. No Upfront or Partial Upfront or All Upfront Payments

EC2 - Spot Instances - Remember

- **Spot Block:** Request Spot instances for **specific duration** (1 or 2 .. or 6 hrs)
- **Spot Fleet:** Request spot instances across a **range of instance types**
 - More instance types => better chances of fulfilling your spot request
- **Linux Spot Instances:** ZERO charge if terminated or stopped by EC2 in the first instance hour. Otherwise, you are charged by second.
 - EC2 terminates in 50 minutes => ZERO cost. You terminate in 50 minutes => Pay for 50 minutes
 - If either EC2 or you terminate spot instance after 70 minutes, you pay for 70 minutes
- Spot instances can be terminated, stopped, or hibernated when interrupted
 - Default - terminated
 - Use **maintain option** while creating spot request for stop and hibernate options
 - Hibernating a Spot instance allows you to save state of EC2 instances and **quickly start up**
- **Safely close** spot request: Cancel Spot Request and Terminate Spot Instances
 - **(Remember)** Canceling a spot request might not terminate active spot instances

EC2 Reserved Instances

- Standard: Commit for a EC2 platform and instance family for 1 year or 3 years. (Up to 75% off)
- Convertible: Standard + **flexibility** to change EC2 platform and instance family. (Up to 54% off)
- Scheduled: Reserve for **specific time period** in a day. (5% to 10% off)
- You can **sell reserved instances** on the AWS Reserved instance marketplace if you do not want to use your reservation

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)

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.

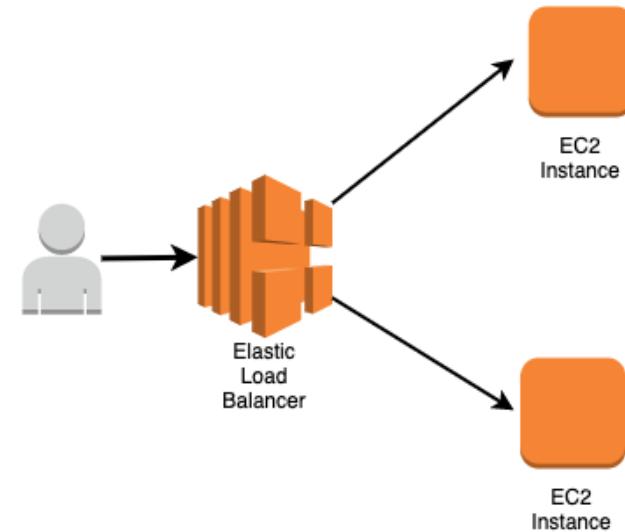
Important EC2 Scenarios - Quick Review

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

Load Balancing

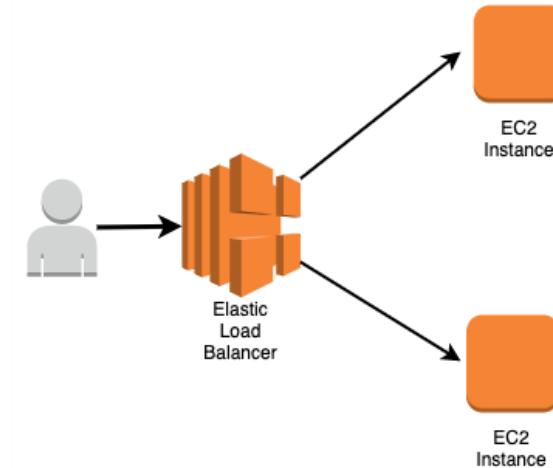
Elastic Load Balancer

- 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**
- **Types:**
 - **Classic Load Balancer (Layer 4 and Layer 7):**
 - Old generation supporting Layer 4(TCP/TLS) and Layer 7(HTTP/HTTPS) protocols (NOT RECOMMENDED)
 - **Application Load Balancer (Layer 7)**
 - New Gen - HTTP/HTTPS and advanced routing approaches.
 - **Network Load Balancer (Layer 4)**
 - New Gen - TCP/TLS and UDP



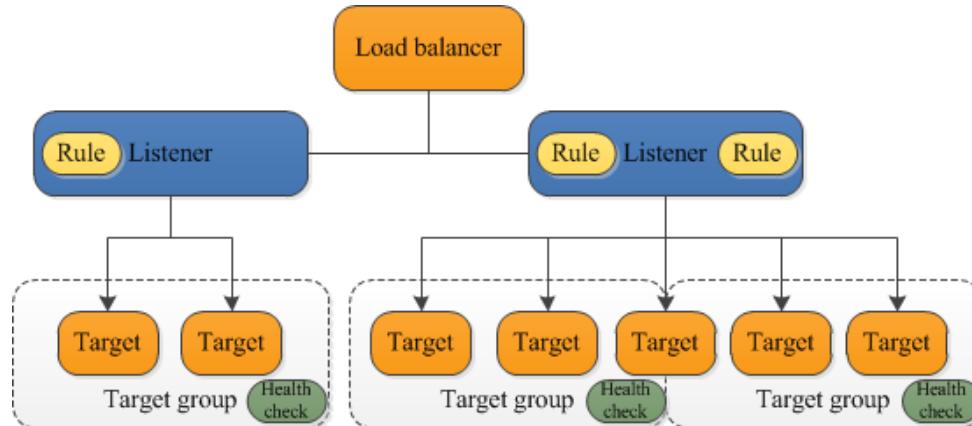
Application Load Balancer

- 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)



How does ALB work?

- Highly decoupled architecture
 - Load balancer can have **multiple listeners**:
 - Each listener has a protocol, a port and a set of rules to route requests to targets
 - 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
 - **Target Group** can be a set of EC2 instances, lambda function or IP Addresses
 - Enable sticky sessions or connection draining at target group level
 - A target can be part of multiple target groups
 - **Listener Rules:** Map request to Target Group
 - Configure multiple listener rules for same listener
 - Rules are executed in the order they are configured.



Reference: AWS Documentation

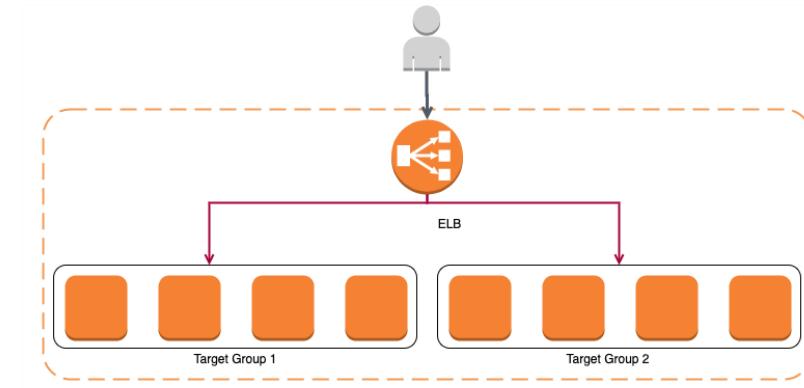
Add listener				
	Listener ID	Security policy	SSL Certificate	Rules
<input type="checkbox"/>	HTTP : 80 arn...6d4fd3790d1b96d9	N/A	N/A	Default: forwarding to awseb-AWSEB-77UXO29Z6IMQ View/edit rules
<input type="checkbox"/>	HTTP : 443 ▲ arn...770a0d2977599957	N/A	N/A	Default: redirecting to HTTP://#{host}:80/#{path}?#{query}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

1 arn...0655b8dbf1d981e6 ▾

IF ✓ Path is /microservice-a	THEN Forward to TARGET_GROUP_A: 1 (100%)
---------------------------------	--

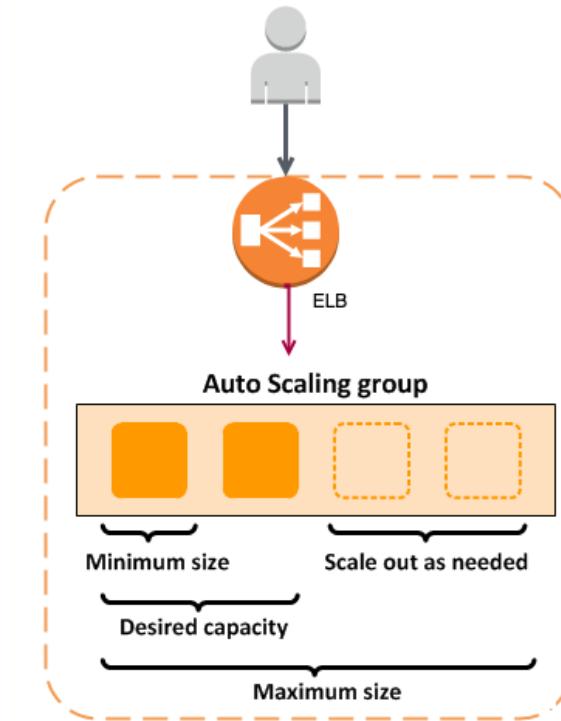
Listener Rules - In Depth

- Possibilities:
 - Based on **path** - `in28minutes.com/a` to target group A and `in28minutes.com/b` to target group B
 - Based on **Host** - `a.in28minutes.com` to target group A and `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`)
 - Based on **IP Address** - all requests from a range of IP address to target group A. Others to target group B.
- Microservice architectures
 - Should we create multiple ALBs?
 - **Nope.** One ALB can support multiple microservices!
 - Create separate target group for each microservices
 - **Classic LB does NOT support multiple target groups**



Auto Scaling Group

- How to scale out and scale in **automatically**?
 - Configure a Auto Scaling Group
- Auto Scaling Group responsibilities:
 - **Maintain** configured number of instances (using 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
 - Can launch On-Demand Instances, Spot Instances, or both
 - **Best Practice:** Use Launch Template
- Auto Scaling Group components:
 - **Launch Configuration/Template** - EC2 instance size and AMI
 - **Auto Scaling Group**
 - 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 Unhealthy instances are 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 Uses CloudWatch alarms CPU utilization >80% => +2 EC2 instances

Dynamic Scaling Policy Types



Scaling Policy	Example(s)	Description
Target tracking scaling	Maintain CPU Utilization at 70%.	Modify current capacity based on a target value for a specific metric.
Simple scaling	+5 if CPU utilization > 80% -3 if CPU utilization < 60%	Waits for cooldown period before triggering additional actions.
Step scaling	+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

Auto Scaling - Scenarios

In 28
Minutes

Scenario	Solution
Change instance type or change size or change AMI of ASG instances	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.
Perform actions before an instance is added or removed	Create a Lifecycle Hook. You can configure CloudWatch to trigger actions based on it.
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

Review

- **Classic Load Balancer**
 - Layer 4(TCP/TLS) and Layer 7(HTTP/HTTPS)
 - Old. Not Recommended by AWS
- **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
- **Network Load Balancer**
 - Layer 4(TCP/TLS and UDP)
 - Very **high performance usecases**
 - Can be assigned a Static IP/Elastic IP

AWS Elastic Beanstalk

AWS Elastic BeanStalk

- **Simplest way to deploy and scale your web application in AWS**
 - Provides end-to-end web application management
 - 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:** Load Balancing, Auto scaling and Managed Platform updates
- **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.
 - Create multiple environments running diff. application versions
 - **Environment Tier:**
 - For batch applications, use **worker tier**
 - For web applications, use **web server tier**

AWS Elastic Beanstalk Environment Tiers

- **Web Server Tier** : Run web applications
 - 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
- **Worker Tier**: Run Batch Applications: 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 and .config files

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

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

- **Source Bundle : ONE AND ONLY ONE ZIP file or WAR file - (MAX SIZE) 512 MB**
 - To deploy multiple WAR files, include them in a ZIP file
 - **DO NOT** have any parent folder (subdirectories are fine)
- **Customize Elastic Beanstalk environment using .config YAML or JSON files:**
 - Placed in a folder **.ebextensions** at root of source bundle with extn **.config**
 - Configure Elastic Load Balancer, Enable X-Ray etc
 - Example: **.ebextensions\xray.config**, **.ebextensions\elb.config**
 - 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?

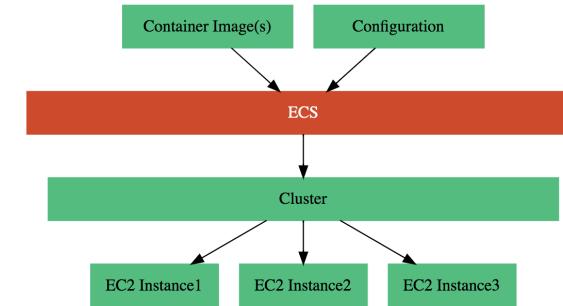
AWS Elastic BeanStalk - Remember

- 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

Containers and Container Orchestration

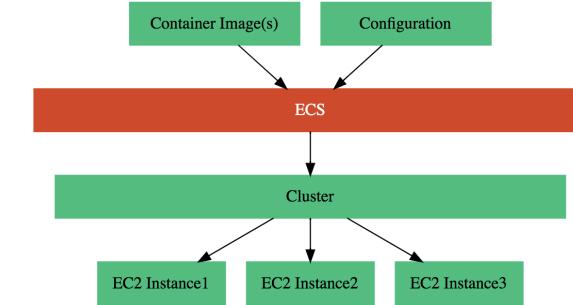
Amazon Elastic Container Service (Amazon ECS)

- Microservices are built in multiple languages (Go, Java, Python, JavaScript, etc)
- Containers simplify deployment of microservices:
 - Step I : Create a self contained Docker image
 - Application Runtime (JDK or Python), Application code and Dependencies
 - Step II : Run it as a container anywhere
 - Local machine OR Corporate data center OR Cloud
- How do you manage 1000s of containers?
 - **Elastic Container Service (ECS)** - Fully managed service for container orchestration
 - Step I : Create a Cluster (Group of one or more EC2 instances)
 - Step II: Deploy your microservice containers
 - **AWS Fargate**: Serverless ECS. DON'T worry about EC2 instances.
 - **Cloud Neutral**: Kubernetes
 - AWS - AWS Elastic Kubernetes Service (EKS)



Amazon Elastic Container Service (Amazon ECS) - Terminology

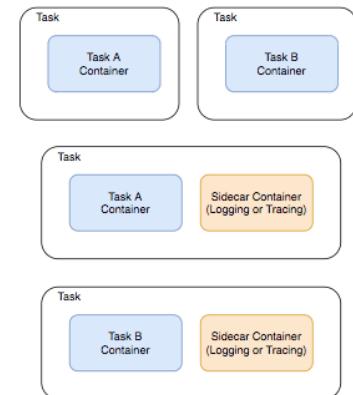
- **Container Instance** = EC2 instance + **container agent**
 - Use ECS ready AMIs
 - Communicates with the ECS cluster
 - Define Cluster Info in `/etc/ecs/ecs.config`
 - Use On-Demand instances or Spot instances



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

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



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

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

- Task Placement Strategies:
 - **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 specified values:
 - Host (instanceId)
 - (OR) Availability Zone(attribute:ecs.availability-zone)
 - (ALSO ALLOWED) Combine strategies and prioritize
- Task Placement Constraints:
 - **distinctInstance** - Place each task on 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-wxyx7890']

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

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

//OR

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

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 and www.app.com/microservice-b)

Amazon ECR (Elastic Container Registry)

- Where do you store docker images for your microservices?
 - Amazon ECR is a Fully-managed Docker container registry provided by AWS
 - (Alternative) Docker Hub
- Docker Commands: Quick Intro To Build and Push Docker Images
 - `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>
 - To pull and push docker images from ECR, you need to login:
 - (DEPRECATED) `aws ecr get-login --region $AWS_DEFAULT_REGION`
 - (REMEMBER) This commands 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`



Routing and Content Delivery

Amazon CloudFront - Content Delivery Network

- Deliver content to your global audience:
 - AWS provides 200+ edge locations around the world
 - Provides high availability and low latency
 - Serve users from nearest edge location (based on user location)
 - If content is not available at the edge location, it is retrieved from the origin server and cached
- Content Source - S3, EC2, ELB or External Websites
- Use Cases: Static web apps, Downloads (media/software)
- Integrates with: AWS Shield (Avoid DDoS attacks)
 - AWS WAF (protect from SQL injection, cross-site scripting)
- Cost Benefits: Zero cost for transfer from S3 to CloudFront
 - Reduce compute workload for your EC2 instances



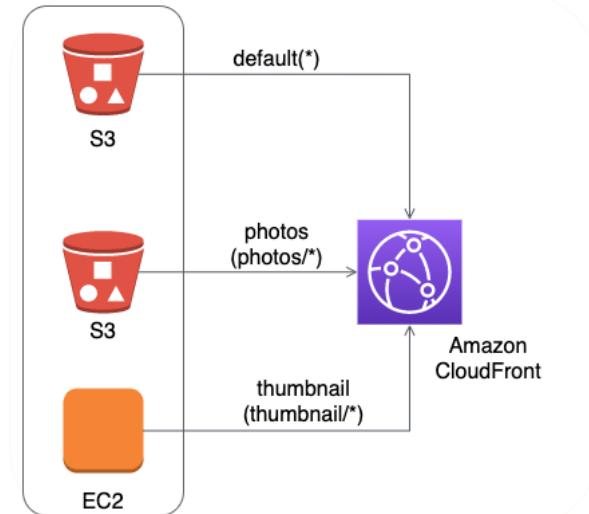
Recommended Architecture - Static Content



- (NOT RECOMMENDED) Serving static content from EC2
- 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)

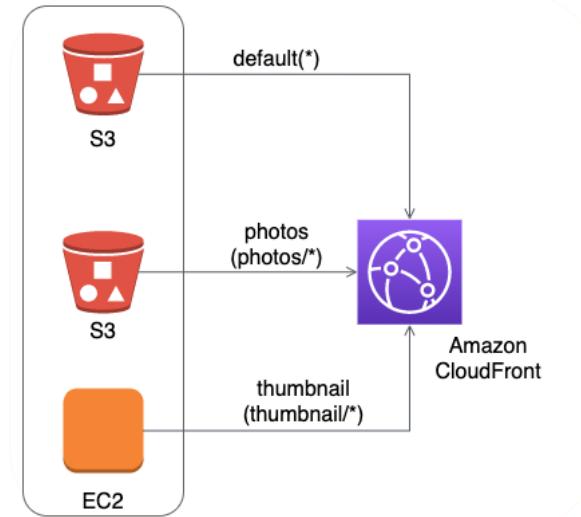
Amazon CloudFront Distribution

- 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



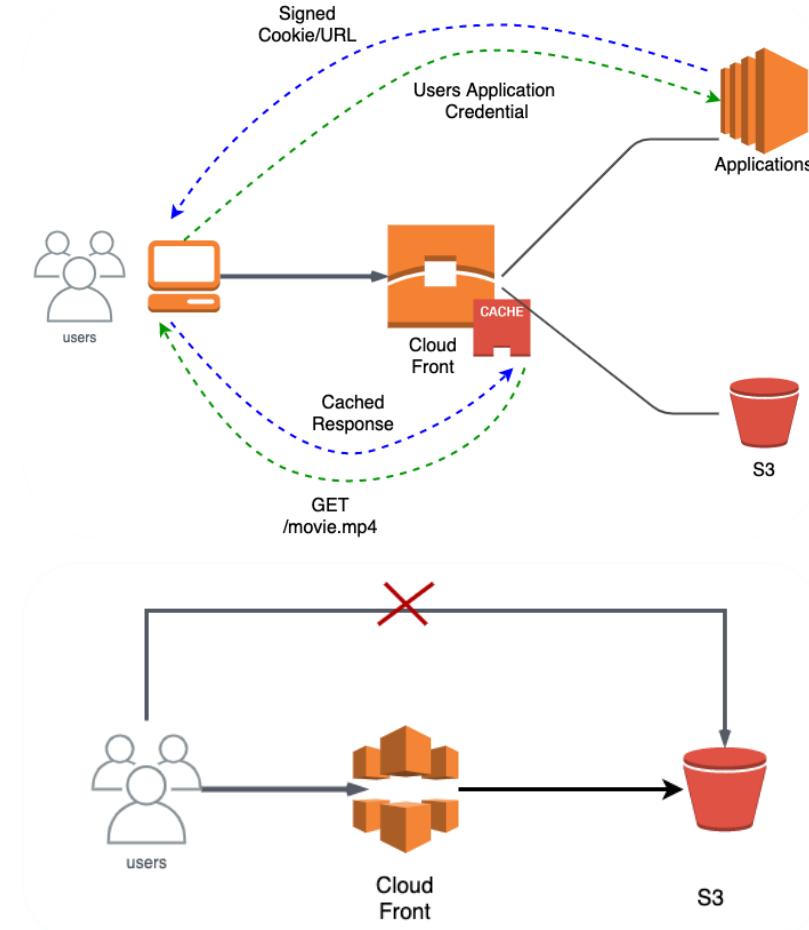
Amazon CloudFront - Cache Behaviors

- 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
 - 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
- Origin Access Identities(OAI)
 - Ensures that only CloudFront can access S3
 - Allow access to S3 only to a special CloudFront user
 - 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

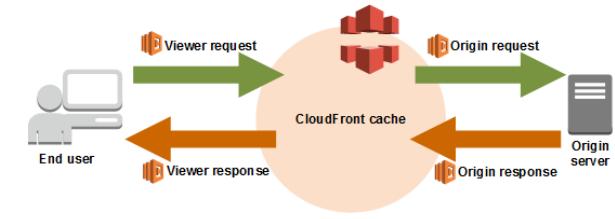


```
{  
    "Version": "2012-10-17",  
    "Id": "PolicyForCloudFrontPrivateContent",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS":  
                    "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity YOUR_IDENTITY_NAME"},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

AWS Lambda@Edge

In 28
Minutes

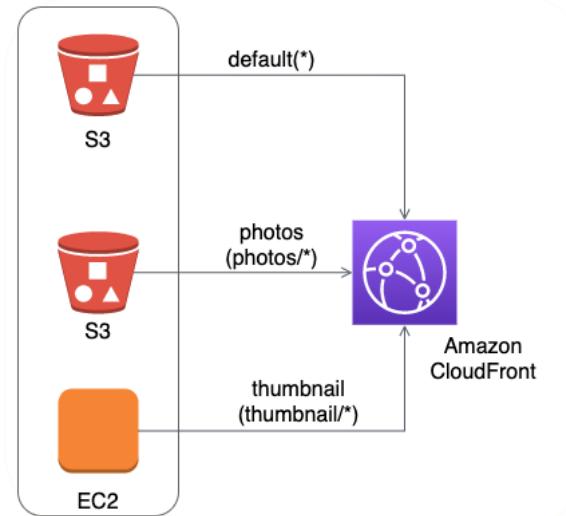
- Run Lambda functions to customize CloudFront content
 - (RESTRICTION) ONLY Python or Node JS supported
- Lambda functions can be run at different points in processing a request in CloudFront:
 - After CF receives a request from a viewer (Viewer Request)
 - Before CF forwards the request to Origin (Origin 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>

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 = Domain Registrar + DNS (Domain Name Server)

- 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
 - Domain Registrar - Buy your domain name
 - DNS - Setup your DNS routing for in28minutes.com
 - Configure Records for routing traffic for in28minutes.com
 - 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?



Route53

Route 53 Concepts

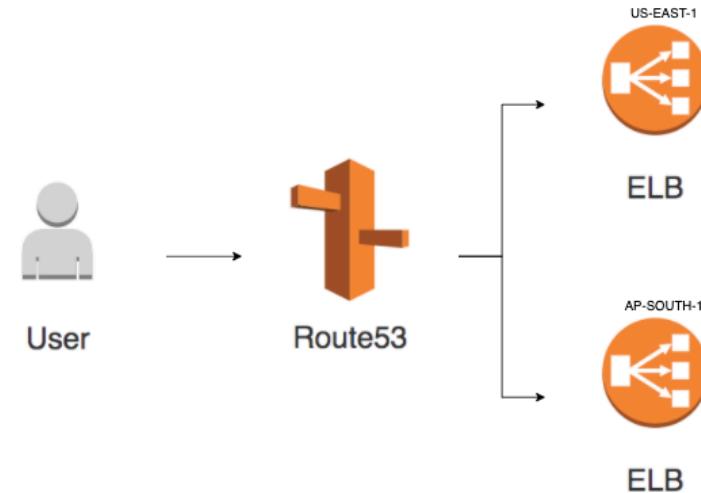
- **Hosted Zone** - Container for DNS records routing traffic for a domain
 - I want to use Route 53 to manage the DNS records for in28minutes.com
 - Create a hosted zone for in28minutes.com in Route 53
 - Hosted zones can be
 - private - routing within VPCs
 - public - routing on internet
 - Standard 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 BUT I can use Route 53 as DNS
 - Create NS records on GoDaddy
 - Redirect to Route 53 Name Servers
 - CNAME - Name1 to Name2
 - Route 53 Specific Extension - Alias records - Route to specific AWS resources (ELB, Elastic Beanstalk, Amazon S3, CloudFront)



Route53

Route 53 - Remember

- Route 53 Specific Extension - Alias records
 - Route to specific AWS resources (ELB, Elastic Beanstalk, Amazon S3, CloudFront)
 - 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)
- 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
Simple	Maps a domain name to (one or more) IP Addresses
Weighted	Maps a single DNS name to multiple weighted resources 10% to A, 30% to B, 60% to C (useful for canary deployments)
Latency	Choose the option with minimum latency Latency between hosts on the internet can change over time
Failover	Active-passive failover. Primary Health check fails (optional cloud Watch alarm) => DR site is used
Geoproximity	Choose the nearest resource (geographic distance) to your user. Configure a bias.
Multivalue answer	Return multiple healthy records (upto 8) at random You can configure an (optional) health check against every record
Geolocation	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



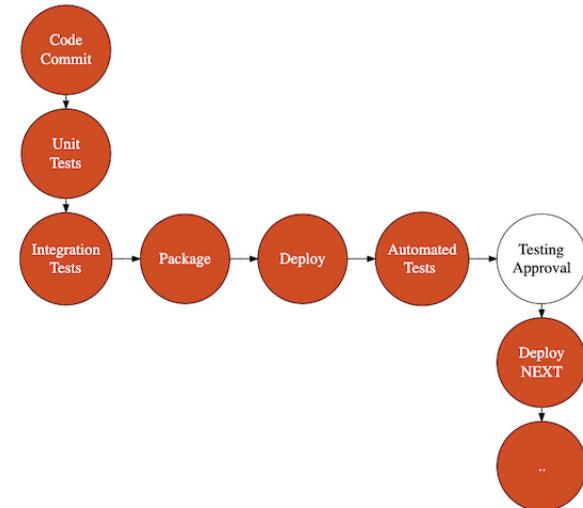
Route53

DevOps

DevOps

DevOps

- Get Better at "**3 Elements of Great Software Teams**"
 - **Communication** - Get teams together
 - **Feedback** - Earlier you find a problem, easier it is to fix
 - **Automation** - Testing, deployment, monitoring etc
- CI/CD Practices:
 - **Continuous Integration**: Continuously run tests & packaging
 - **Continuous Deployment**: Continuously deploy to test env
 - **Continuous Delivery**: Continuously deploy to production
- CI/CD Tools:
 - **AWS CodeCommit** - Private source control (Git)
 - **AWS CodePipeline** - Orchestrate CI/CD pipelines
 - **AWS CodeBuild** - Build and Test Code (packages and containers)
 - **AWS CodeDeploy** - Automate Deployment(ECS, Lambda etc)



DevOps - Practices - IAC (Infrastructure as Code)

In 28
Minutes



- 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

DevOps - IAC (Infrastructure as Code) - AWS



- **Infrastructure Provisioning**
 - Open Source: Terraform
 - AWS:
 - AWS CloudFormation: Provision AWS Resources
 - AWS SAM (Serverless Application Model): Provision Serverless Resources
- **Configuration Management**
 - Open Source Tools: Chef, Puppet, Ansible
 - AWS Service: OpsWorks (Chef, Puppet in AWS)
- **(Remember) Most DevOps Tools AutoScale**
 - CodeCommit, CodePipeline, CodeBuild, CodeDeploy, CloudFormation, OpsWorks

AWS CodeCommit

In 28
Minutes

- 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 CodeDeploy

- Automate application deployments to EC2/On-premise instances, Lambda functions, ECS services etc.
 - Release new features avoiding manual errors (Avoid/minimize downtime)
 - Configure where to release (blue-green/in-place) and how to shift traffic (canary, linear, all-at-once)
 - Supports Autoscaling and Automated rollbacks
 - Integrates with CodePipeline, CI/CD tools (Jenkins, GitHub etc) and IAAC tools (Ansible, Puppet, Chef etc)
- **In-place deployment (ONLY For EC2/On-Premises)**
 - Step 1 : Application is stopped
 - Step 2 : Latest version is installed
 - Step 3 : Application started and validated
- **Blue/green deployment**
 - A new environment is created and traffic shifted to a new environment
 - Works with EC2 (Does NOT work with on-premises) Lambda and ECS



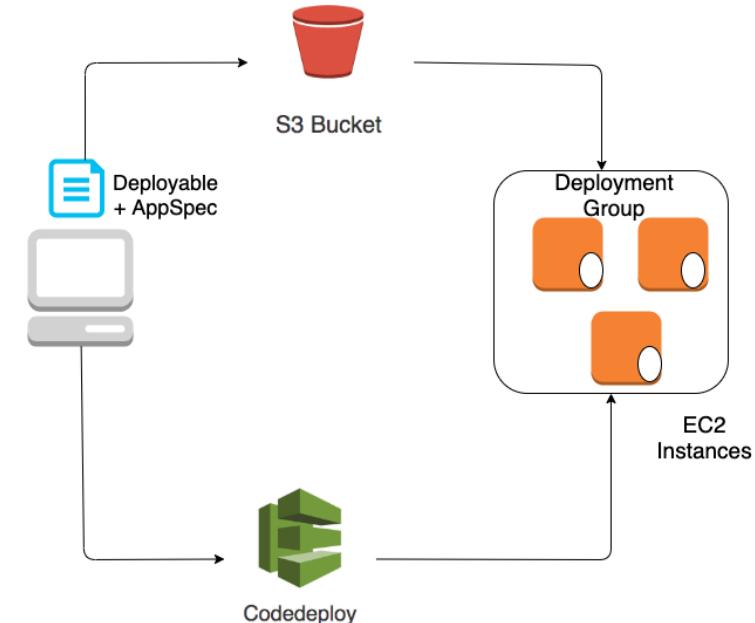
AWS CodeDeploy - Configuration

- **Application** - Unique name for application to be deployed
- **Compute platform** - Where to deploy? (EC2/On-premises / AWS Lambda / Amazon ECS)
- **Deployment configuration** - Deployment rules:
 - **Deployment group** - Which instances should be used?
 - **Target revision** - Which version should be deployed?
 - **Deployment type** - What type of deployment? (In-place/Blue-Green)
 - **Traffic Shifting** - How is traffic routed to new deployment?
 - Canary , Linear , All-at-once
 - **Rollbacks** - "Redeploy previously deployed revision as a new deployment" (technically **new deployments** using old versions)
 - **Automatic rollbacks:** Triggered on failed deployments or on CloudWatch alarms
 - **Manual rollbacks:** Create a new deployment with previously deployed revision



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

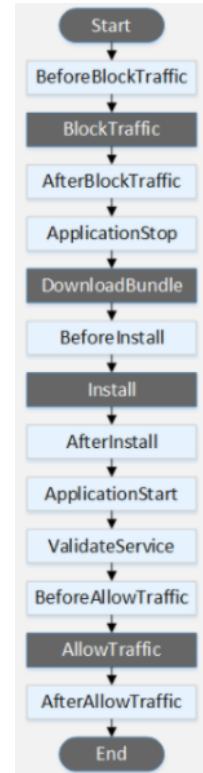
AWS CodeDeploy - Amazon ECS

```
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

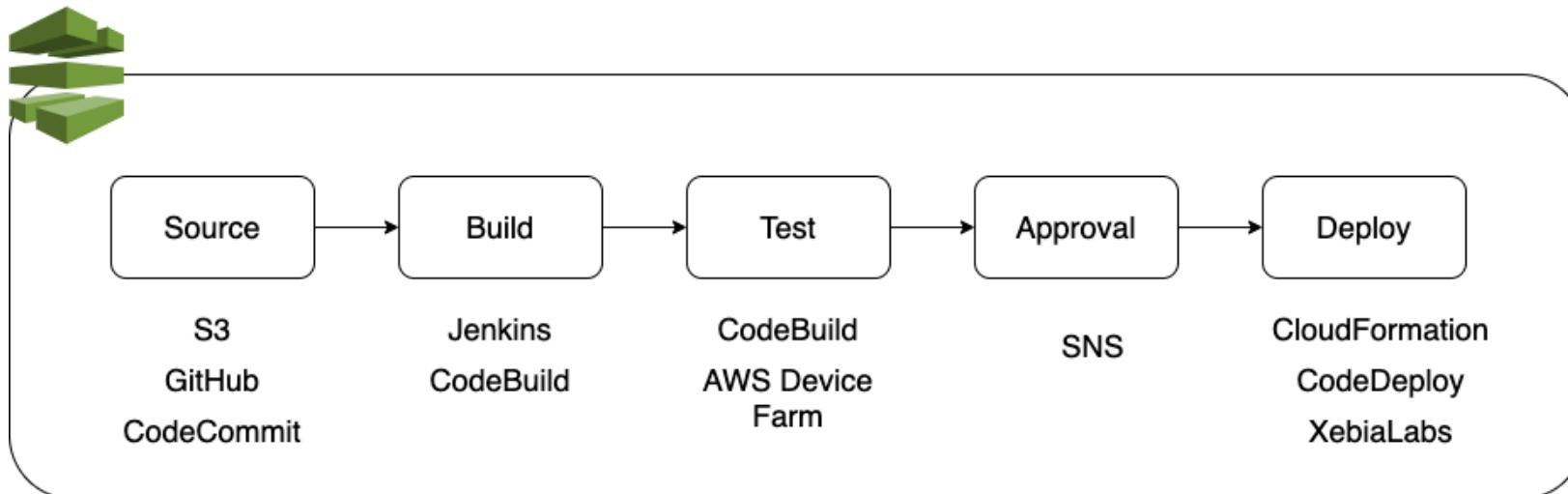
CodeDeploy - Order of hooks

- 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



AWS CodePipeline

In 28
Minutes



- **AWS CodePipeline:** Create Pipelines with multiple stages:
 - Example stages: Build, Test, Release, Deploy are some examples
 - Model, visualize and automate steps involved
 - Integrates with various AWS & Third-party tools to automate end to end release process
- Integrates with a wide variety of tools (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">1. Clone GitHub Repo. Create CodeCommit Repo.2. Create an IAM user for CodeCommit with policy AWSCodeCommitPowerUser.3. Configure Git credentials for CodeCommit (HTTPS, recommended for most users) or an SSH key pair to use when accessing CodeCommit (SSH)4. Push to CodeCommit Repo.

DevOps - Scenario Questions - 2

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`.

How to run integration tests in CodeBuild by connecting to the test database inside a VPC?

Connect CodeBuild to a VPC - Configure Security Groups and a VPC Endpoint

AWS CloudFormation

AWS CloudFormation - 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 CloudFormation

- Free to use - Pay only for the resources provisioned
 - Get an automated estimate for your configuration
- **Template:** A JSON or YAML defining multiple resources
 - 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 template file (track changes over time)
- **Stack:** Group of resources created from CF template
- **Change Sets:**
 - To make changes to stack, update the template
 - Change set shows what would change if you execute (Verify and Execute)



CloudFormation

AWS CloudFormation Templates - Examples

JSON

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

YAML

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

AWS CloudFormation - 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 CloudFormation - 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 CloudFormation - 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
 - Define constraints on parameters - AllowedPattern, AllowedValues, MaxLength etc
- Pseudo parameters - Parameters predefined by AWS CloudFormation (!Ref "AWS::Region")
 - Examples: AWS::AccountId, AWS::Region, AWS::StackId, AWS::StackName

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 CloudFormation - Outputs, Conditions and Transform

In 28
Minutes

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31  
Parameters:  
  EnvType:  
    Type: String  
Conditions:  
  IsProd: !Equals [!Ref EnvType, prod]  
Outputs:  
  HelloWorldFunctionArn:  
    Value: !GetAtt HelloWorldFunction.Arn  
    Condition: IsProd
```

- **Outputs** - Export values from templates for later use
 - CloudFormation does not hide or encrypt output section. Passwords will be visible.
- **Conditions** - Based on the condition: Resource or output is created
 - Do different things in different environments
- **Transform** - Specifies macros to process your template
 - AWS::Include transform - Insert boilerplate template snippets into your templates
 - AWS::Serverless transform - Converts SAM templates to CloudFormation

AWS CloudFormation - 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 CloudFormation - 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 CloudFormation - 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]

Other Intrinsic functions

- **Fn::Join** - Join multiple values. Pass delimiter & array of values.
 - "Fn::Join": ["", ["http://"], {"Fn::GetAtt": ["ElasticLoadBalancer", "DNSName"]}]]
- **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"

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 CloudFormation - 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 CloudFormation scripts
 - Break up a big CloudFormation 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 CloudFormation - 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

Common Resource Attributes

`UpdatePolicy:`

`AutoScalingReplacingUpdate:`

`WillReplace: true`

- **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
- **DeletionPolicy:** Preserve or backup resource when stack is deleted.
 - Possible Values - Retain(Do not delete), Snapshot(take a snapshot) or 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 CloudFormation - Remember

- ElasticBeanstalk - Pre-packaged CloudFormation template with a UI
- 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
- Execute CloudFormation templates from AWS CLI:
 - aws cloudformation create-stack/list-stacks/describe-stacks
- **StackSet**- Create resources in many accounts and regions
 - Establish Trust relationship between administrator account and target accounts
- Python helper scripts simplify deployment on EC2 instances:
 - `cfn-init` : Retrieve resource metadata, install packages etc
 - `cfn-signal`: Synchronize resources(Signal with CreationPolicy or WaitCondition)
- Cross Stack: Resource is referenced. Reuse of Resource.
- Nested stack: Resource is recreated. Allows reuse of templates

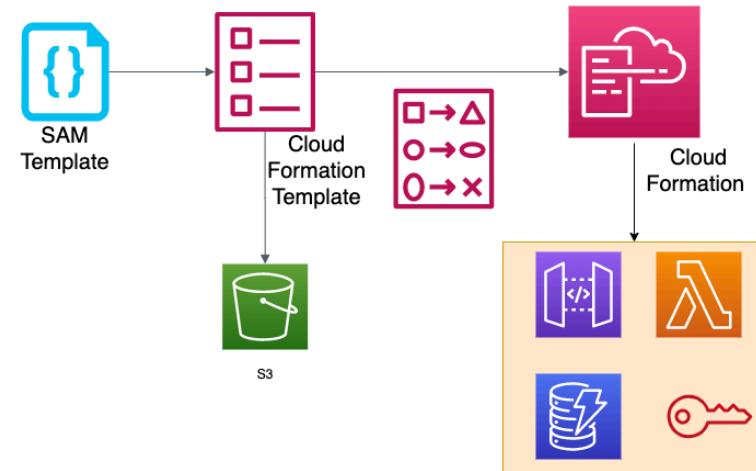


CloudFormation

Serverless Application Model SAM

Serverless Application Model

- **Serverless Application Model** - Open source framework for building serverless applications
 - Infrastructure as Code (IAC) for Serverless Applications
 - Integrate Serverless Best Practices
 - Tracing(X-Ray), CI/CD(CodeBuild,CodeDeploy,CodePipeline) etc
 - Define YAML file with resources
 - Functions, APIs, Databases..
 - (BEHIND THE SCENES) SAM config => CloudFormation
 - Benefits of SAM:
 - Simple 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
         //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

```
//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
```

- Pre-configured policy templates to give access to your Lambda Functions:
 - Examples: SQSPollerPolicy, CloudWatchPutMetricPolicy, DynamoDBCrudPolicy, DynamoDBReadPolicy, DynamoDBWritePolicy

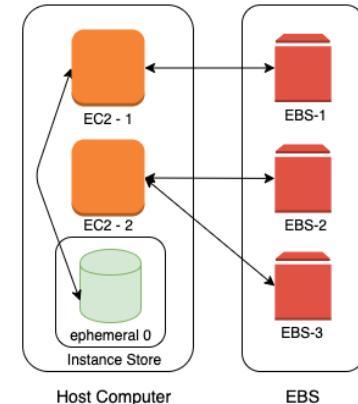
SAM - Scenario Questions

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">1. sam package && sam deploy (OR)2. aws cloudformation package && aws cloudformation deploy
Which sections of the SAM template are required?	Transform and Resources

Storage in AWS - Block Storage and File Storage

Block Storage

- Two popular types of Block Storage:
 - **Instance Store:** Physically attached to EC2 instance
 - Ephemeral storage - Temporary data (Data lost - hardware fails or instance termination)
 - **CANNOT** take a **snapshot** or restore from snapshot
 - **Use case:** cache or scratch files
 - **Elastic Block Store (EBS):** Network Storage
 - More Durable. Very flexible **Provisioned capacity**
 - **Increase size when you need it** - when attached to EC2 instance
 - **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

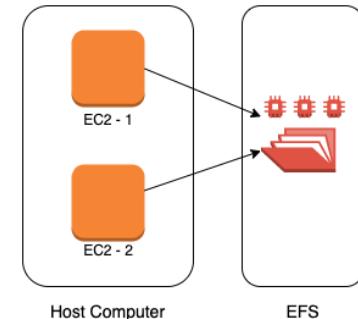
EBS Snapshots and AMIs



- You can create:
 - Snapshot from EBS volume and vice versa
 - AMI from EC2 instance and vice versa
 - AMI from root EBS volume snapshots

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
- (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.

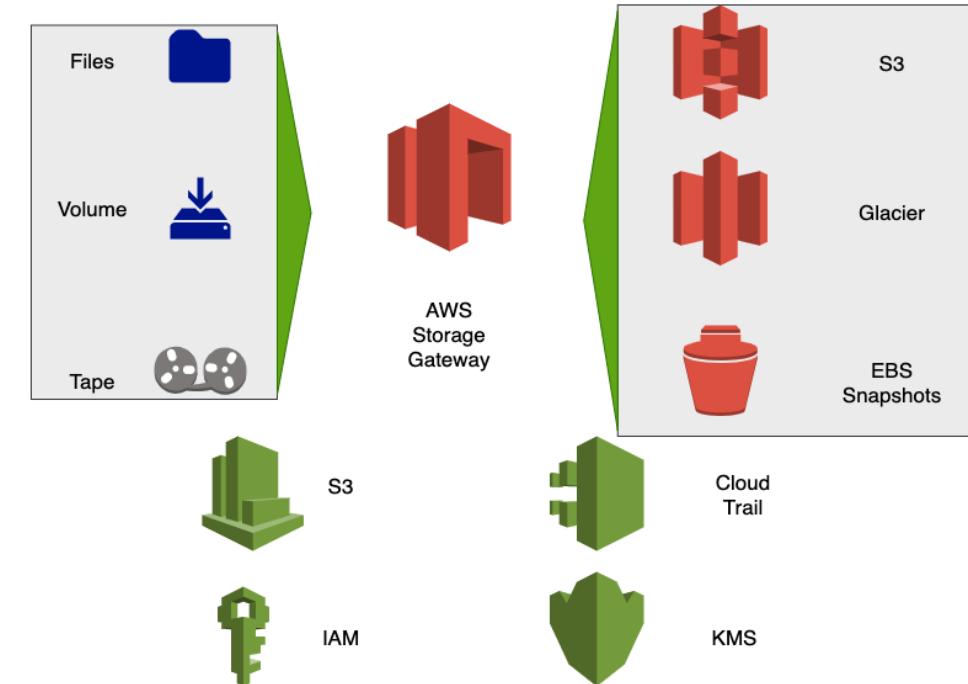


Review of storage options

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

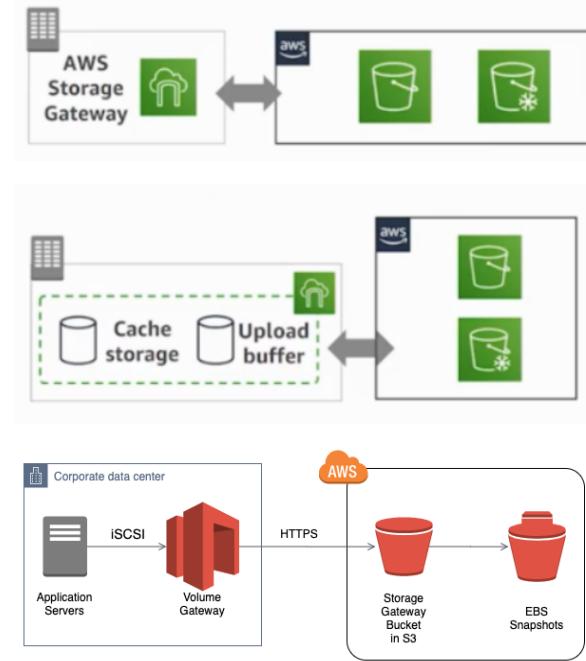
AWS Storage Gateway

- 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



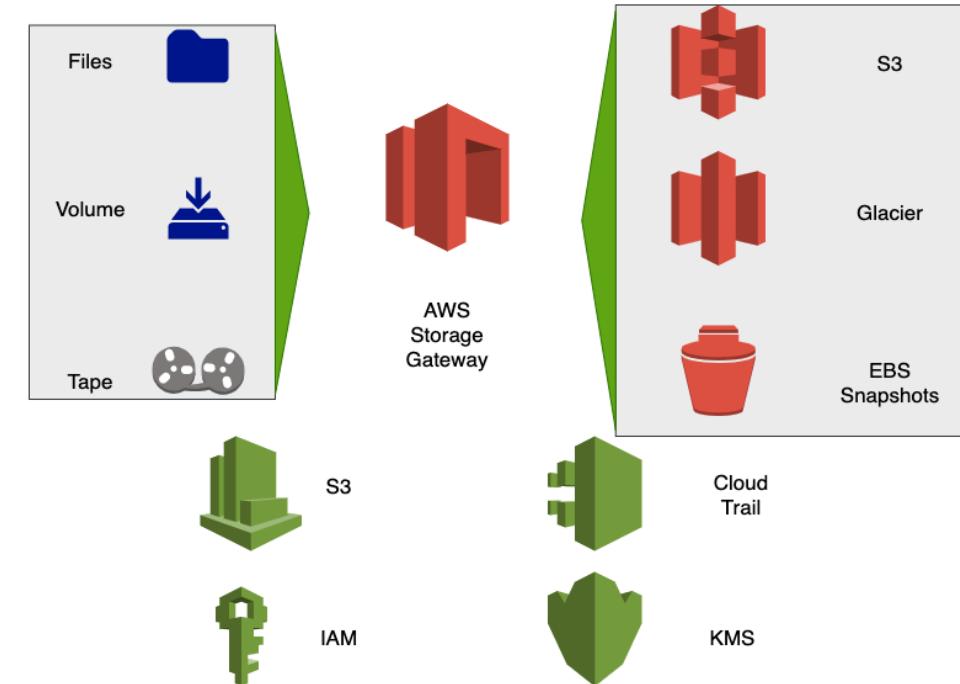
AWS Storage Gateway - Types

- **Storage File Gateway** - Storage for file shares
 - Files stored in Amazon S3 & Glacier
- **Storage Tape Gateway** - Virtual tape backups
 - Tapes stored in Amazon S3 & Glacier
 - Avoid complex physical tape backups (wear and tear)
 - No change needed for tape backup infrastructure
- **Storage Volume Gateway** : Cloud Block Storage
 - Use cases: Backup, Disaster Recovery, Cloud Migration
 - (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 - Review

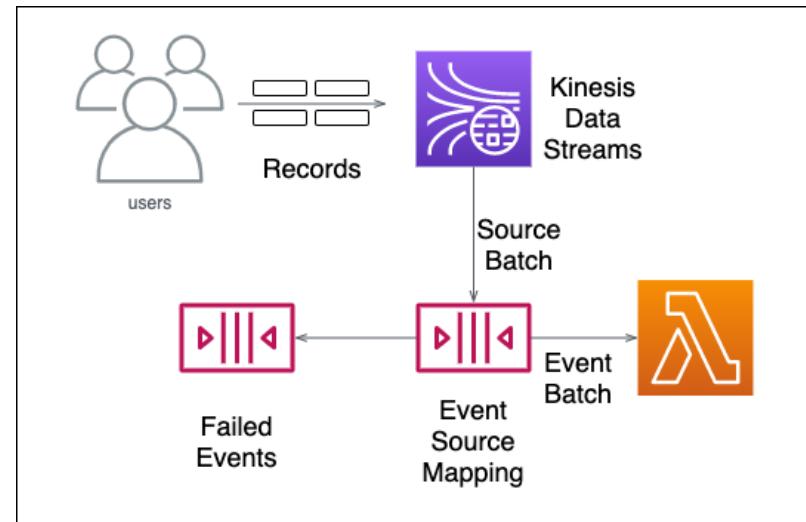
- Key to look for : Hybrid storage (cloud + on premise)
- File share moved to cloud => **AWS Storage File Gateway**
- Tape Backups on cloud => **AWS Storage Tape Gateway**
- Volume Backups on cloud (Block Storage) => **AWS Storage Volume Gateway**
 - High performance => **Stored**
 - Otherwise => **Cached**



More Serverless

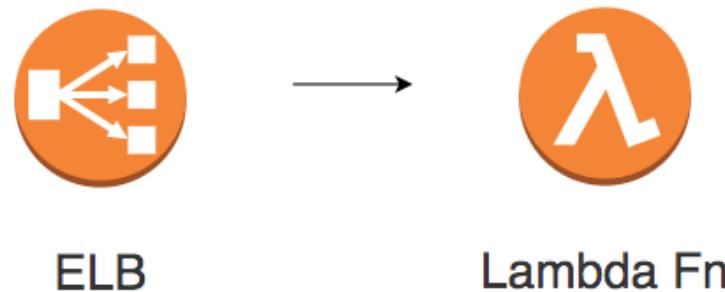
AWS Lambda - Event Source Mapping

- 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 & Application Load Balancers

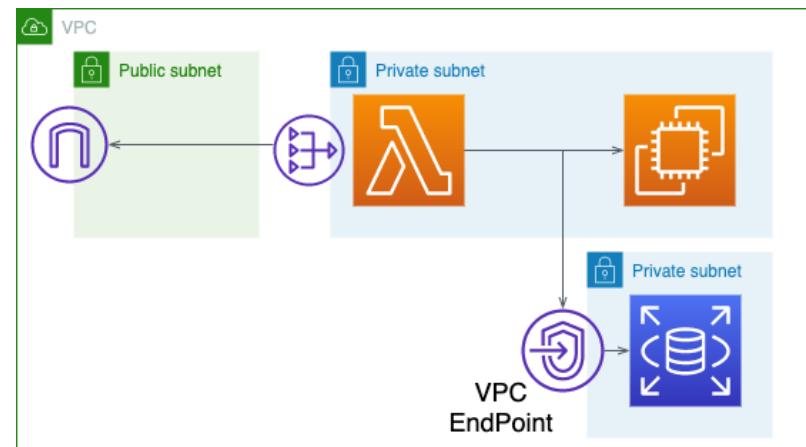
In 28
Minutes



- 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 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



Lambda CloudFormation - Deployment

- Define Lambda functions in CF scripts!
- Deployment package for complex Lambdas:
 - 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 same package as code or use Layers
 - (Option 1) Directly Update Lambda Function
 - *aws lambda update-function-code --function-name func1 --zip-file zippath*
 - (Option 2) Upload to S3 (>50 MB : S3 mandatory)
 - Upload Zip to S3
 - *aws lambda update-function-code --function-name my-function -s3-bucket BUCKET_NAME --s3-key OBJECT_KEY*
 - (OR) Use CloudFormation to deploy Lambda function from S3
 - (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

```
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
```

```
TestLambdaFunction:
  Type: AWS::Lambda::Function
  Properties:
    Handler: HelloWorld.lambda_handler
    Role: !GetAtt LambdaExecutionRole.Arn
    Code:
      S3Bucket: lambda-s3-sample
      S3Key: HelloWorld.zip
      S3ObjectVersion: jUzGu2o2NjsILaVxIoY92
  Runtime: python3.6
```

AWS Lambda quotas

In 28
Minutes

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

SAM - Deployment with CodeDeploy

DeploymentPreference:

Type: Canary10Percent10Minutes

Alarms:

- !Ref AliasErrorMetricGreaterThanZeroAlarm
- !Ref LatestVersionErrorMetricGreaterThanZeroAlarm

Hooks:

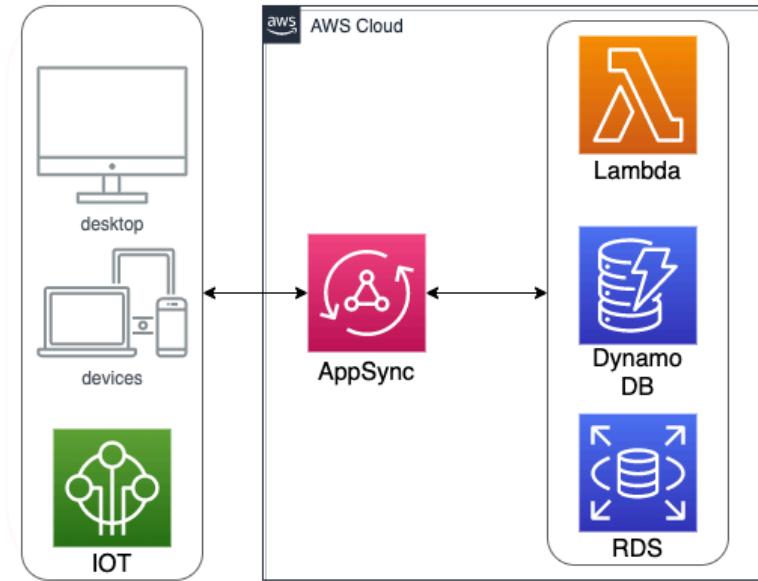
PreTraffic: !Ref PreTrafficLambdaFunction

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 etc
- (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

- 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



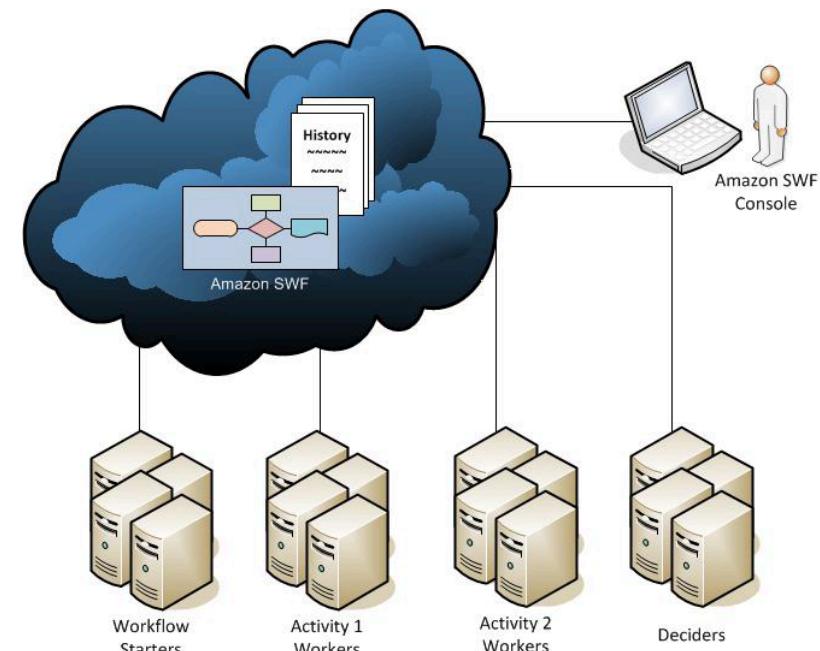
Amazon Simple Workflow Service (SWF)

- 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>

Serverless Options

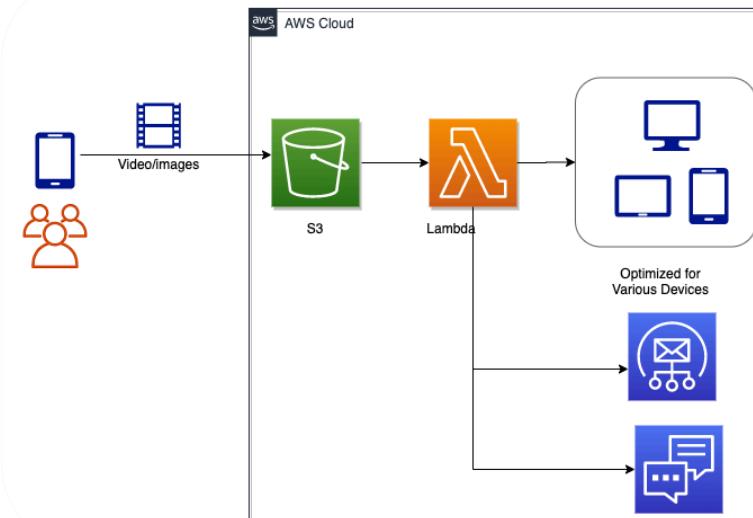
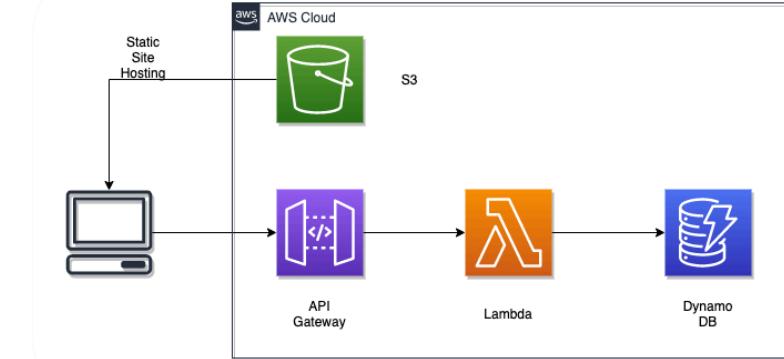
Solution	Description
AWS Lambda	Run code without provisioning servers! FaaS (Function as a Service)
Lambda@Edge	Run lambda functions at AWS Edge Locations (CloudFront)
AWS Fargate	Container Orchestration without worrying about ec2 instances
Amazon S3	Highly scalable Object Storage
Amazon EFS	Elastic file storage for UNIX compatible systems
DynamoDB	Fast, scalable, distributed NoSQL database. RCU/WCU or expensive serverless mode
Aurora Serverless	Run Amazon RDS with Aurora in serverless mode (EARLY STAGE)
RDS Proxy	Manage short lived DB connections from client applications (incl. Lambdas) to RDS
API Gateway	API Management platform - authorization, rate limiting and versioning
AWS Step Functions	Orchestrate workflows (state machines) with Lambda, Fargate, SQS, SNS etc

Serverless Options - Application Integration and Analytics

Solution	Description
Amazon SNS	Pub-sub messaging. Broadcast notifications - SMS, e-mails, push notifications
Amazon SQS	Fully managed queuing service to decouple your apps
Amazon Kinesis	Multiple solutions to process streaming data
Amazon Athena	Query using SQL on data in Amazon S3
Amazon Cognito	Authorization and authentication solutions for web/mobile apps
AWS Serverless Application Model	Open source framework for building serverless applications

Serverless Use case Examples

- Web Application Architecture:
 - Static content stored in S3
 - API Gateway and Lambda are used for the REST API
 - DynamoDB is used to store your data
- Real time event processing:
 - User uploads videos to S3
 - S3 notifications are used to invoke Lambda functions to optimize videos for different devices.



CloudTrail, Config, CloudWatch and X-Ray

AWS CloudTrail and AWS Config

Solution	Description
AWS CloudTrail	<p>Track events, API calls, changes made to your AWS resources.</p> <p>Who (made the request), What (action, parameters, end result) and When?</p> <p>Multi Region Trail - One trail for all AWS regions vs Single Region Trail - Only events from one region</p>
AWS Config	<p>Auditing: Complete inventory of your AWS resources</p> <p>Resource history and change tracking - Find how a resource was configured at any point in time</p> <p>Governance - Customize Config Rules for specific resources or for entire AWS account and Continuously evaluate compliance against desired configuration</p>
AWS Config vs AWS CloudTrail	<p>AWS Config - What did my AWS resource look like?</p> <p>AWS CloudTrail - Who made an API call to modify this resource?</p>

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

- Amazon CloudWatch Metrics: Most AWS services provide free metrics
 - Enable **detailed monitoring** (\$\$\$) if needed
 - **EC2**: CPUUtilization, NetworkIn, NetworkOut
 - (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
 - **ELB**: HTTPCode_Target_2XX_Count, HTTPCode_Target_4XX_Count
 - **DynamoDB**: AccountProvisionedReadCapacityUtilization, ConsumedReadCapacityUnits
 - **Lambda**: Throttles, Errors, ConcurrentExecutions, Duration
 - **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?)
- Metrics exists only in the region in which they are created.
 - Metrics can't be deleted and expire after 15 months

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 Logs

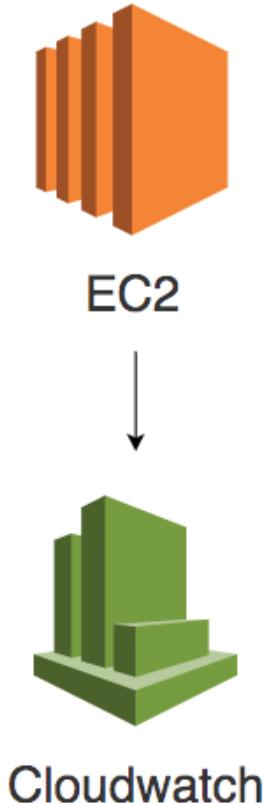
- Monitor and troubleshoot using system, application and custom log files
- **Real time application and system monitoring:**
 - Use **CloudWatch Logs Insights** to write queries and get actionable insights
 - Monitor for patterns in your logs and trigger alerts based on them
 - Example : Errors in a 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



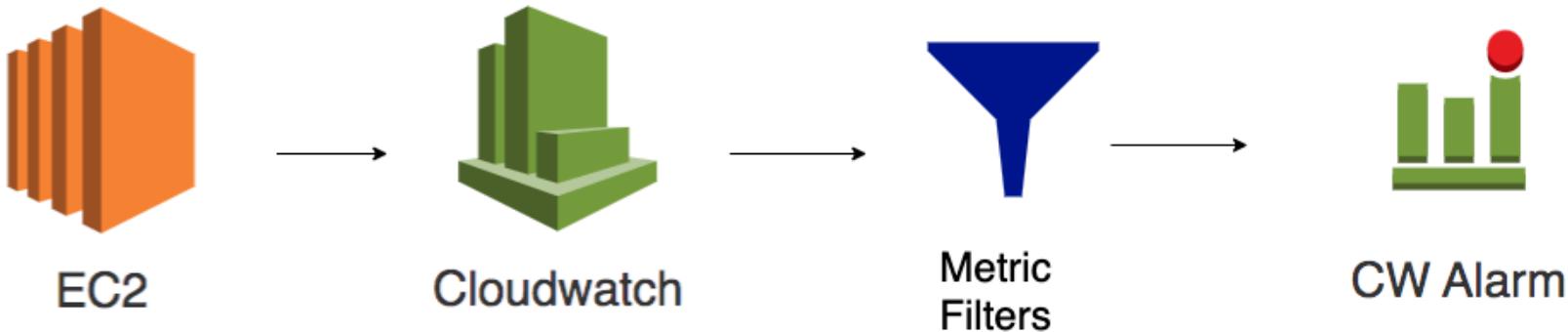
Cloudwatch

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)



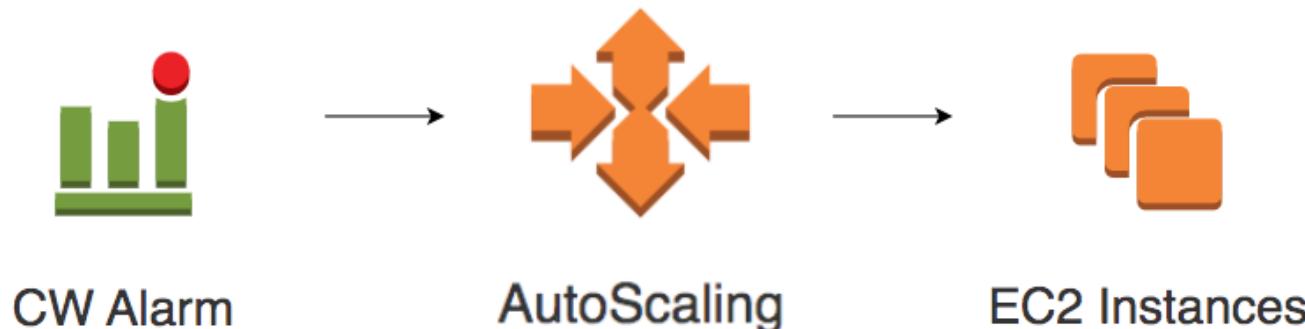
Amazon CloudWatch Logs Metrics - Filters



- 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

N

Amazon CloudWatch Events

- **Take immediate action based on events on AWS resources**
 - Call a AWS Lambda function when an EC2 instance starts
 - 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
- **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)
 - CodeCommit (pullRequestCreated)
 - CodePipeline (Pipeline Execution State Change, Stage Execution State Change)
 - Amazon EC2 State Change Events - stop start terminate

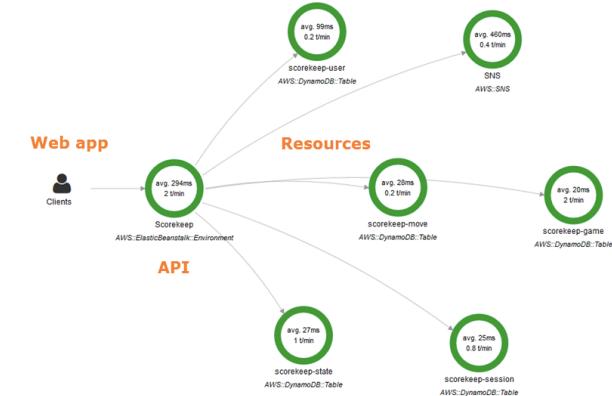


Amazon EventBridge vs CloudWatch Events

- 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
 - **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

X-Ray

- Trace request across microservices/AWS services
 - Analyze, Troubleshoot errors, Solve performance issues
 - Gather tracing information
 - From applications/components/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-bd862e3fe
 - Each service in request chain sends traces to X-Ray with trace ID
 - X-Ray gathers all the information and provides visualization
 - How do you reduce performance impact due to tracing?
 - Sampling - Only a sub set of requests are sampled (Configurable)
 - How can AWS Services and your applications send tracing info?
 - Step I : Update Application Code Using X-Ray SDK
 - Step II: Use X-Ray agents (EASY to use in some services! Ex: 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

```
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
}

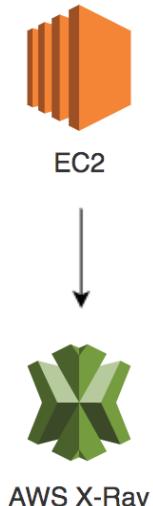
AWSXRay.middleware.setSamplingRules(rules);

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.



AWS CLI

AWS CLI - Getting Started

In 28
Minutes

- Run Commands and Write Scripts - Perform actions with AWS Services
- 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 1000. CLI requests less no of items in each 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

- 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`
 - config file created with region & output format
- DO NOT Use Access Keys on EC2 instances
 - (BEST PRACTICE) Assign Roles to EC2 instances
- How do you handle access to multiple aws accounts?
 - Use profiles
- Use `--profile` argument to call `aws configure` and other aws commands

```
$ 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
```

```
$ 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
```

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)

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

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)

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

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

[my_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 } }

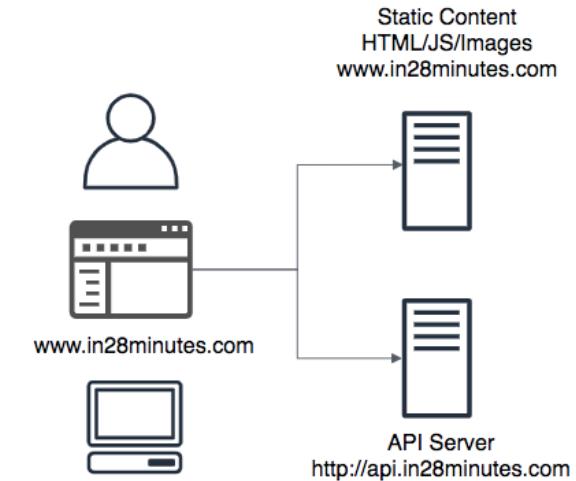
//IAM Policy
"Effect": "Allow",
>Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::XYZ:role/myrole"
```

- 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 (my_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

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

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)



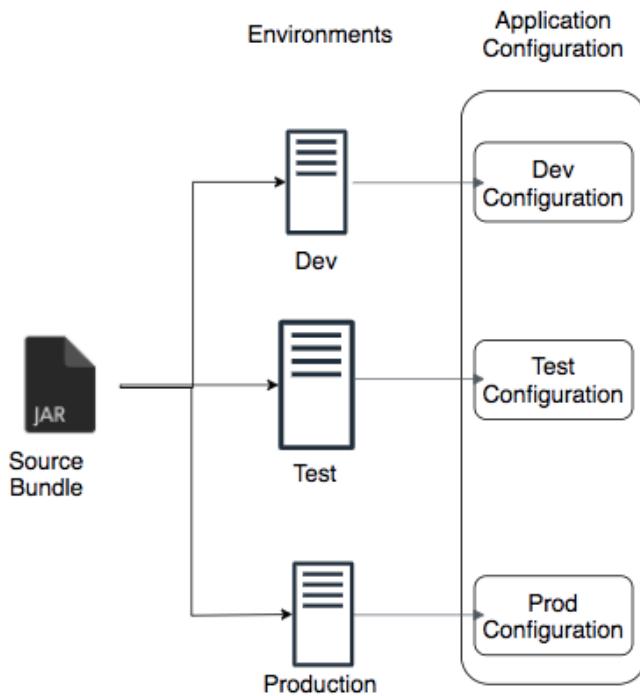
API Gateway

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;
};
```

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 - (Lazy Loading or Cache Aside) vs Write Through

```
\\"LAZY LOADING
get_data(id)
    record = cache.get(id)
    if (record == null)
        record = db.query("YOUR_QUERY", id)
        cache.set(id, record)
    return record

\\WRITE THRUH
get_data(id)
    return cache.get(id)
save_data(id, value)
    record = db.query("YOUR_QUERY", id, value)
    cache.set(id, record)
    return success
```

- Lazy Loading: 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
- Write Through: 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 - Comparisons

- **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 vs DAX**
 - DAX is customized for DynamoDB
 - Very few code changes are needed to cache data from DynamoDB
 - ElastiCache is generic cache

Next slide →

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

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 Pillar

- Avoid/Minimize effort and problems with:
 - Provisioning servers, Deployment, Monitoring and Support
- Recommendations:
 - Use Managed Services: No worry about managing servers, availability 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
- Recommended Approach:
 - 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 (Ex:Use Amazon Elasticsearch to analyze your logs)



AWS Lambda



CloudFormation



Codepipeline



AWS Config



Cloudwatch

Security Pillar

- Principle of least privilege for least time
 - Use temporary credentials when possible (IAM roles, Instance profiles)
 - Enforce MFA and strong password practices
 - Rotate credentials regularly
- Security in Depth - Apply security in all layers
 - VPCs and Private Subnets (Security Groups and Network Access Control List)
 - 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 apps from XSS, SQL injection etc)
 - Use CloudFormation (Automate provisioning infra that adheres to security policies)
- Protect Data at Rest
 - Enable Versioning (when available)
 - Enable encryption - KMS and Cloud HSM (Rotate encryption keys)



AWS IAM



AWS Shield



AWS WAF



AWS KMS



Cloud HSM

Security Pillar - 2

- Protect Data in Transit
 - 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 your data stays in AWS network when possible(VPC Endpoints and AWS PrivateLink)
- Detect Threats: 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



AWS IAM



AWS Shield



AWS WAF



AWS KMS



Cloud HSM

Reliability Pillar

- Reliability: Ability to recover from infra and app issues
 - Adapt to changing demands in load
- Best Practices
 - 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
 - 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



AWS Lambda



Amazon SQS



Amazon SNS



API Gateway



AutoScaling

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 Pillar: Meet needs with min. resources

- Continue being efficient as demand and technology evolves
- Best Practices:
 - Use Managed Services (Avoid Undifferentiated Heavy Lifting)
 - Go Serverless (Lower transactional costs and less operational burden)
 - Experiment (Cloud makes it easy to experiment)
 - Monitor Performance (Trigger CloudWatch alarms - Perform actions with SQS and Lambda)
- 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



API Gateway



Cloudwatch



Amazon SQS

Cost Optimization Pillar: Run systems at lowest cost

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



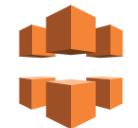
AWS Lambda



Trusted Advisor



Cloudwatch



CloudFront

Get Ready

Certification Resources

Title	Link
Certification - Home Page	https://aws.amazon.com/certification/certified-developer-associate/
AWS Architecture Home Page	https://aws.amazon.com/architecture/
AWS FAQs	https://aws.amazon.com/faqs/ (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?

FASTEST ROADMAPS

in28minutes.com

