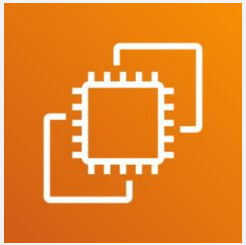

aws
certified
Solutions
Architect
Professional
DOLFINed

AWS COMPUTE OPTIONS

YOU CAN DO IT TOO!



ELASTIC COMPUTE CLOUD (EC2) REFRESHER

AWS Elastic Compute Cloud (EC2)

Review Topic : Elastic Compute Cloud

EC2 – Root/Boot Volume

- EC2 instance root/boot volumes can be EBS or Instance Store volumes

- EBS-Backed EC2 instance
  - It has an EBS root volume

- Instance-store backed EC2 instance
  - It has an Instance-store root volume

Review Topic : Elastic Compute Cloud

EC2 Instance families

- **General Purpose**
  - Balanced memory and CPU
  - Suitable for most applications
  - Ex. M3, M4, T2

- **Compute Optimized**
  - More CPU than memory
  - Compute & HPC intensive use
  - Ex. C2, C4

- **Memory Optimized**
  - More RAM/memory
  - Memory intensive apps, DB, and caching
  - Ex. R3, R4

Review Topic : Elastic Compute Cloud

EC2 Instance Families

- **GPU compute instances**
  - Graphics Optimized
  - High performance and parallel computing
  - Ex. G2

- **Storage Optimized**
  - Very high, low latency, I/O
  - I/O intensive apps, data warehousing, Hadoop
  - Ex. I2, D2

Review Topic : Elastic Compute Cloud

EC2 Status checks

- By default, AWS EC2 service performs automated status checks **every minute**
  - This is done on every running EC2 instance to identify any hardware or software issues
  - Each status check returns either a pass or a fail status
  - If one or more status checks return a “fail”, the overall EC2 instance status is changed to “impaired”

- Status checks are built into the AWS EC2 service,
  - They can not be configured, deleted, disabled, or changed

- You can configure CloudWatch to initiate actions (Reboot or Recovery) on impaired EC2 instance (i.e for failed status checks)

Source: aws.amazon.com

Review Topic : Elastic Compute Cloud

EC2 Monitoring

- EC2 service can send its metric data to AWS Cloudwatch every 5 minutes (enabled by default)
  - This is free of charge
  - It is called basic monitoring

- You can choose to enable detailed monitoring while launching the instance (or later) where the EC2 service will send its metric data to AWS Cloudwatch every 1 minute
  - Chargeable
  - It is called detailed monitoring

- You can set CloudWatch alarm actions on EC2 instance(s) to :
  - Stop, Restart, Terminate, or Recover your EC2 instance
    - You can use Stop or Terminate actions to save cost
    - You can use the reboot and recover to move your EC2 instance to another host

Review Topic : Elastic Compute Cloud

EC2 – Stopping an EC2 instance

- When you stop an **EBS backed instance**, any data in any Instance-store volumes is lost
  - Despite the fact that the instance can be re-started, all instance store data will be gone

- When you stop an EBS-Backed EC2 instance
  - Instance performs a shutdown
  - State changes from running -> Stopping -> Stopped
  - EBS volumes remain attached to the instance
  - Any data cached in RAM or Instance Store volumes is gone
  - Most probably, when restarted again, it will restart on a new physical host
  - Instance retains its private IPv4 address, any IPv6 address
  - Instance releases its public IPv4 address back to AWS pool
  - Instance retains its Elastic IP address
    - You will start to be charged for un-used Elastic IP

Review Topic : Elastic Compute Cloud

EC2 Termination Protection

- This is a feature you can enable such that an EC2 instance is protected against accidental termination through API, Console, or CLI
- This can be enabled for Instance-store backed and EBS-Backed Instances
- CloudWatch can ONLY terminate EC2 instances if they **do not** have the termination protection enabled

Review Topic : Elastic Compute Cloud

EC2 – Instance Meta Data

- Instance Meta Data:
  - This is instance data that you can use to configure or manage the instance
    - Examples are IPv4 address, IPv6 address, DNS Hostnames, AMI-ID, Instance-ID, Instance-Type, Local-hostname, Public Keys, Security groups…
  - Meta data can be only viewed from within the instance itself
    - i.e you have to logon to the instance
  - **Meta data is not protected by encryption** (cryptography), anyone that has access to the instance can view this data
  - To view an EC2 Instance's Meta Data (from the EC2 instance console):

    GET http://169.254.169.254/latest/meta-data/

    OR

    Curl http://169.254.169.254/latest/meta-data/

Review Topic : Elastic Compute Cloud

EC2 – Instance User Data

- Instance user data:
  - Is data supplied by the **user at instance launch** in the form of a script to be executed during the instance boot
  - **User data is limited to 16KB**
  - User data can only be viewed from within the instance itself (logon to it)
  - **You can change user data**
    - To do so, you need to stop the instance first (EBS backed)
      - Instance -> actions -> Instance-settings -> View/Change user data
  - User data is not protected by encryption, do not include passwords or sensitive data in your user data (scripts)
  - You are not charged for requests to read user data or metadata

Review Topic : Elastic Compute Cloud

EC2 – IAM Roles

- In General, for an AWS services to have permission to read or write to another service, an IAM role is required to be attached to the first AWS Service with rights/permissions on the second AWS service

- Drawing on that, for an EC2 instance to have access to other AWS services (example S3) you need to configure an IAM Role, which will have an IAM policy attached, under the EC2 instance.
  - Applications on the EC2 instance will get this role permission from the EC2 instance's metadata

- You can add an IAM to an EC2 instance during or after it is launched

AWS Elastic Compute Cloud (EC2)

EC2 Purchasing Options

Review Topic : Elastic Compute Cloud

EC2 – Instance Purchasing Options

- Reserved Instances
  - 1- or 3-years commitments, high savings, can be zonal (per AZ) or Regional scoped.
- Scheduled instances
  - Upfront purchase instance capacity for a recurring schedule
- Spot Instances
  - Request AWS unused EC2 instances, highest savings, availability not guaranteed when you need it
- Dedicated hosts
  - Pay for a fully dedicated physical host
- Dedicated Instances
  - Pay by the hour for instances that run on single-tenant hardware
- On-Demand
  - Pay by the second for instances that you launch
- Savings plans
  - 1 or 3 years usage commitment for a consistent amount of usage, in USD per hour.

Review Topic : Elastic Compute Cloud

EC2 – Spot instances use cases

- Use it if you are flexible regarding the time you want to run your applications
- Use if your applications can be interrupted (in case AWS terminates the instances)
- Suitable for
  - Data analysis
  - Batch jobs
  - Background processing
  - Optional tasks
- I/O optimized I series Instance(s) are not available as a spot instance

Review Topic : Elastic Compute Cloud

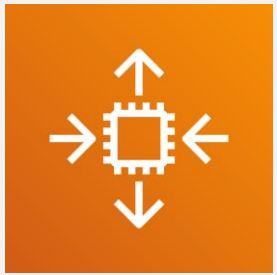
EC2 – Strategies for using Spot Instances

- Use spot instances to augment your reserved and on-demand compute capacity, since spot instances' availability is not guaranteed

- Always guarantee the minimum level of compute using RIs and On-demand and add spot to add compute capacity and save costs

Redundant Array of Independent Disks (RAID)

- RAID array types:
  - RAID 0:
    - Has stripping and no mirroring
    - Provides the best I/O performance among RAID types
    - Resulting I/O is the sum of the individual disks I/Os
    - Failure of one EBS volume means failure of the entire array
  - RAID 1:
    - Redundancy (writing the same data to multiple drives), no Stripping
    - No I/O performance enhancement
  - RAID 10:
    - Redundancy and Stripping (Combines both RAID 0 and RAID 1)
    - Good performance and Redundancy



EC2 AUTO SCALING REFRESHER

AWS Auto Scaling

AWS Auto Scaling Introduction

Review Topic : Auto Scaling

AWS Auto Scaling

- AWS Auto Scaling allows for the configuration of automatic scaling for the AWS resources that are part of your application very quickly.
- Automatic scaling can be configured for individual resources or for whole applications.
- Scaling plans is how Auto scaling is configured and managed
  - Scaling plans uses dynamic scaling and predictive scaling to automatically scale the application resources.
- AWS Auto Scaling is useful for applications that experience daily or weekly variations in traffic flow, including:
  - Cyclical traffic such as high use of resources during regular business hours and low use of resources overnight
  - On and off workload patterns, such as batch processing, testing, or periodic analysis
  - Variable traffic patterns, such as marketing campaigns with periods of spiky growth
- AWS Auto Scaling can be used to scale the following AWS resources:
  - EC2, Spot Instances, DynamoDB, Amazon Aurora, Amazon ECS

Review Topic : Auto Scaling

Application Auto Scaling

- Application Auto Scaling is a web service for developers and system administrators who need a solution for automatically scaling their scalable resources for individual AWS services beyond Amazon EC2.
- Application Auto Scaling allows you to configure automatic scaling for the following resources:
  - Amazon ECS services
  - Spot Fleet requests
  - Amazon EMR clusters
  - AppStream 2.0 fleets
  - DynamoDB tables and global secondary indexes
  - Aurora replicas
  - Amazon SageMaker endpoint variants
  - Custom resources provided by your own applications or services.
  - Amazon Comprehend document classification endpoints
  - Lambda function provisioned concurrency

Review Topic : Auto Scaling

Application Auto Scaling

- You can also use Application Auto Scaling and Amazon EC2 Auto Scaling in combination with AWS Auto Scaling to scale resources across multiple services.

- AWS Auto Scaling can help you maintain optimal availability and performance by combining predictive scaling and dynamic scaling (proactive and reactive approaches, respectively) together to scale your Amazon EC2 capacity faster.

- Application Auto Scaling allows you to automatically scale your scalable resources according to conditions that you define.
  - Target tracking scaling—Scale a resource based on a target value for a specific CloudWatch metric.
  - Step scaling— Scale a resource based on a set of scaling adjustments that vary based on the size of the alarm breach.
  - Scheduled scaling—Scale a resource based on the date and time.

EC2 Auto Scaling

AWS EC2 Auto Scaling Groups

Review Topic : Auto Scaling

EC2 Auto Scaling

- Is an AWS feature that allows your AWS compute needs (EC2 instances fleet) to grow or shrink depending on your workload requirements
- Auto Scaling helps you save cost by cutting down the number of EC2 instances when not needed, and scaling out to add more instances only when it is required
- **AS Components:**
  - **Launch Configuration:**
    - Is the configuration template used to create new EC2 instances for the ASG, defines parameters like:
      - Instance family, instance type, AMI, Key pair, Block devices, and Sec groups are parameters defined in the launch configuration
  - **AS Group:**
    - Is a logical grouping of EC2 instances
  - **Scaling Policy (Plan)**
    - Determines when/if and how the ASG scales or shrinks (On-demand/Dynamic scaling, Cyclic/Scheduled scaling)

Review Topic : Auto Scaling

EC2 Auto Scaling – Launch Templates

- Is similar to a launch configuration
- It specifies AMI ID, Instance type, Security group, tags, key pair among other parameters
- Launch templates allow you to have multiple versions of a template, which you can then reuse to create other templates or template versions.
- AWS recommend creating launch templates
- **Using launch templates, you can provision your capacity using on-demand and spot instances (which can not be done using launch configurations)**
  - This will allow you to achieve the desired scale, performance, and cost targets.
- You can create a template:
  - From scratch
  - Create a new version of an existing one
  - Copy parameters from a launch configuration, running instance, or another template
- Launch templates can not be changed/edited after it is created, but versions can be created

Review Topic : Auto Scaling

Auto Scaling features

- Auto Scaling can span Multi-AZs within the same AWS region. But not across regions.
  - It can be used to create Fault Tolerant designs within a region in AWS
- Cost:
  - There is no additional cost for launching AS Groups. You pay for what you use of EC2 instances
- It works well with AWS ELB, Cloud Watch, and Cloud Trail
- AS is compliant with PCI DSS
- **AZ Rebalance -** Auto Scaling service always tries to distribute EC2 instances evenly across AZs where it is enabled
  - If AS finds that the number of EC2 instances launched by an ASG into subject AZs is not balanced, AS will initiate a Re-Balancing activity
- If Auto Scaling fails to launch instances in an AZ (for AZ failure or capacity unavailability..etc), it will try in the other AZs defined for this AS Group until it succeeds

Review Topic : Auto Scaling

Adding an ELB to the ASG

- You can attach one or more ELBs (Classic, ALB, NLB are all supported) to your existing AS Group
  - The ELB(s) must be in the same region as the AS Group
  - Instances and the ELB(s) must be in the same VPC
- For CLB add the load balancer, for ALB & NLB add the Target group
- Once you do this, any EC2 instance existing or added by the AS Group will be automatically registered with the ASG defined ELB(s)
- Auto Scaling honors connection draining configuration when de-registering an instance manually from an ELB.
- Auto Scaling classifies its EC2 instances health status as either Healthy or Unhealthy
  - By default, AS uses EC2 Status Checks only to determine the health status of an Instance
  - When you have one or more ELBs defined with the AS Group, you can configure Auto Scaling to **use "both" the** EC2 Health Checks and the ELB Health Checks to determine the Instances health status

Review Topic : Auto Scaling

ASG – Scaling Policies

- Maintain a current number of instances all the time
- Manual Scaling
  - Manually change ASG's min/desired/max, attach/detach instances
- Cyclic (schedule based) scaling
  - Predictable load change
- On-demand/Dynamic (Event based) scaling
  - Scaling in response to an event/alarm
- Predictive scaling
  - Combines AWS Auto Scaling with On Demand scaling (Proactive and reactive together)
- Target Tracking Scaling
  - With target tracking scaling policies, you select a scaling metric and set a target value.
  - Amazon EC2 Auto Scaling creates and manages the CloudWatch alarms that trigger the scaling policy and calculates the scaling adjustment based on the metric and the target value.
- Scaling based on Amazon SQS Queue length (number of messages in a Queue)

- An ASG can have multiple policies attached to it at any time

Review Topic : Auto Scaling

Monitoring Auto Scaling Groups

- AWS EC2 service sends EC2 metrics to cloud watch about the ASG instances
  - Basic Monitoring (Every 5 minutes enabled by default – free of charge)
  - You can enable detailed (every 1 minute – chargeable)

EC2 CONTAINER SERVICE (ECS) AND DOCKER

Amazon Elastic Container Service

Introduction

AWS EC2 Container Service

AWS ECS

- Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster.
  - You can host your cluster on a **serverless infrastructure that is managed by Amazon ECS** by launching your services or tasks using the **Fargate launch type.**
  - For more control you can host your tasks on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances **that you manage** by using the EC2 launch type.
- Amazon ECS lets you:
  - Launch and stop container-based applications with simple API calls,
  - Allows you to get the state of your cluster from a centralized service,
  - Gives you access to many familiar Amazon EC2 features.
- You can use Amazon ECS to schedule the placement of containers across your cluster based on your resource needs, isolation policies, and availability requirements.
- Amazon ECS can be used to create a consistent deployment and build experience, manage, and scale batch and Extract-Transform-Load (ETL) workloads, and build sophisticated application architectures on a microservices model.

source: aws.amazon.co

AWS ECS

Features of Amazon ECS

- Amazon ECS is a **regional** service that simplifies running application containers in a highly available manner across multiple Availability Zones within a region.
- You can create Amazon ECS clusters within a new or existing VPC.
- After a cluster is up and running, you can **define task definitions and services** that specify which Docker container images to run across your clusters.
- Container images are stored in and pulled from container registries, which may exist within or outside of your AWS infrastructure.

source: aws.amazon.co

AWS ECS

Containers and Images

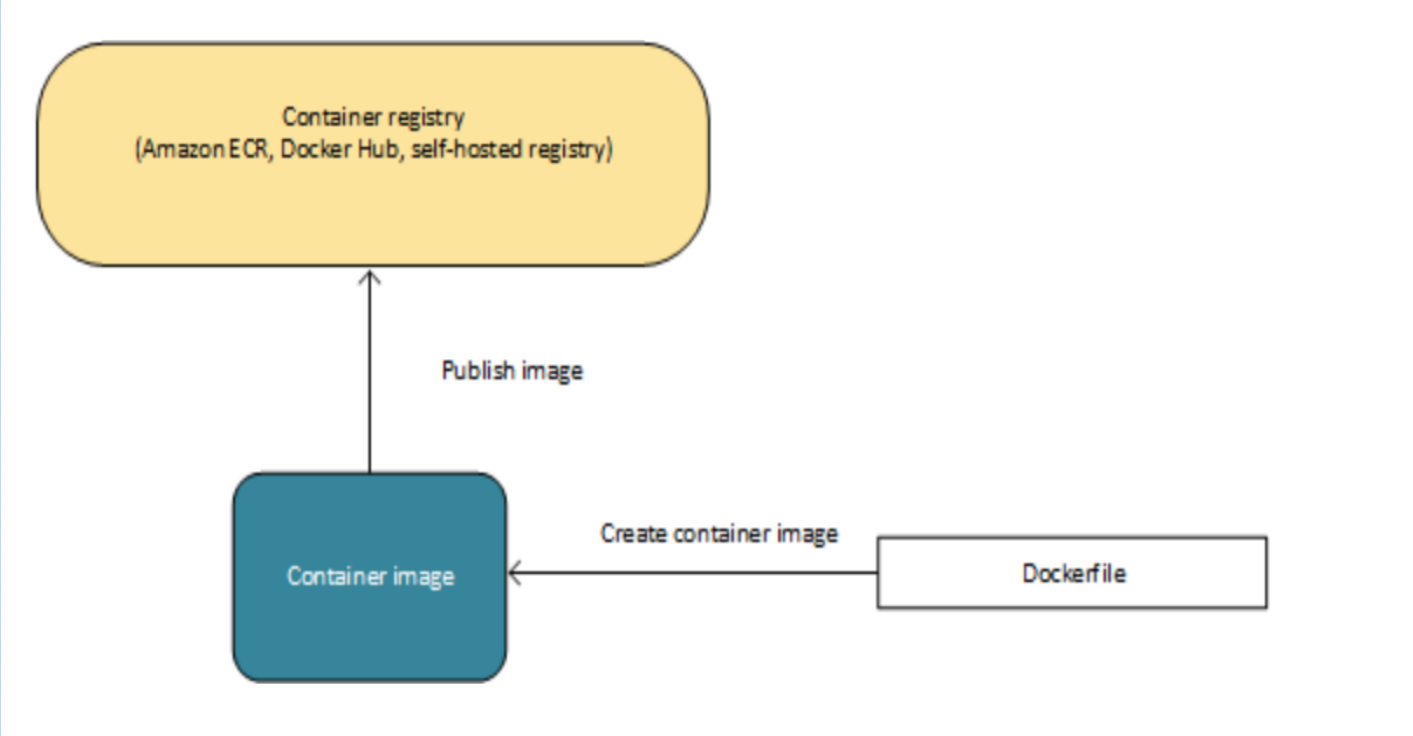
- To deploy applications on Amazon ECS, your application components must be architected to run in *containers*.
- A Docker container is a standardized unit of software development, containing everything that your software application needs to run: code, runtime, system tools, system libraries, etc. Containers are created from a read-only template called an *image*.
- Images are typically built from a Dockerfile, a plain text file that specifies all of the components that are included in the container. These images are then stored in a *registry* from which they can be downloaded and run on your cluster.

source: aws.amazon.co

AWS EC2 Container Service

AWS ECS

- Docker Image creation workflow:



source: aws.amazon.co

AWS ECS Launch Types

- **Fargate Launch Type**

  The Fargate launch type allows you to run your containerized applications without the need to provision and manage the backend infrastructure.

  Just register your task definition and Fargate launches the container for you.
- You Fargate cluster is hosted on a **serverless infrastructure that is managed by Amazon ECS.**
- The Fargate launch type only supports using container images hosted in Amazon ECR or publicly on Docker Hub.
- If you use the Fargate launch type with tasks within your cluster, Amazon ECS manages your cluster resources.

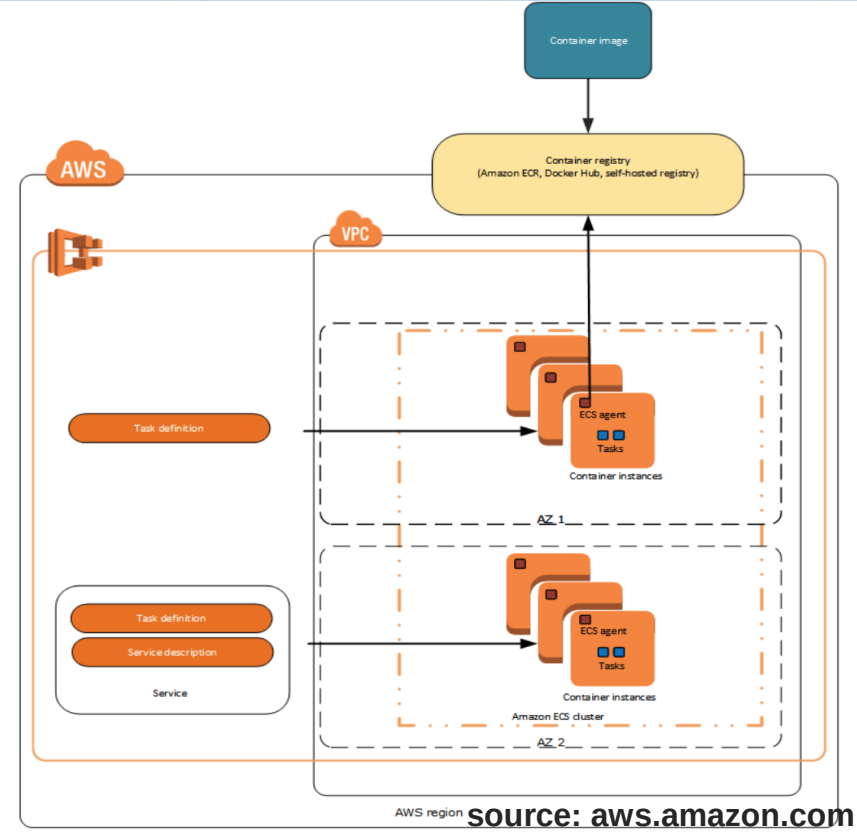
source: aws.amazon.co

AWS EC2 Container Service

AWS ECS

Amazon ECS launch types:

- **EC2 Launch Type**
- The EC2 launch type allows you to run your containerized applications on a cluster of Amazon EC2 instances that you manage.
  - Provides for more control
  - You need to manage the EC2 fleet (cluster)
- Private repositories are currently only supported using the EC2 launch type.



source: aws.amazon.com

AWS ECS

Amazon ECS Container Instances

- An Amazon ECS container instance is an Amazon EC2 instance that is running the Amazon ECS container agent and has been registered into a cluster.
- When you run tasks with Amazon ECS, your tasks using the EC2 launch type are placed on your active container instances.
- When you run tasks using Amazon ECS, you place them on a *cluster*, which is a logical grouping of resources.
- If you use the EC2 launch type, then your clusters will be a group of container instances you manage.
- Amazon ECS downloads your container images from a registry that you specify and runs those images within your cluster.

source: aws.amazon.co

Amazon Elastic Container Service

ECS Task Definition & Tasks

AWS ECS – Task Definition

Task Definitions

- To prepare your application to run on Amazon ECS, you create a *task definition*.
- The task definition is a text file, in JSON format, that describes one or more containers, up to a **maximum of ten**, that form your application. It can be thought of as a blueprint for your application.
- Amazon ECS task definitions use Docker images to launch containers on the container instances in your clusters.
- **An ECS Task:**
  - **Is the instantiation of a task definition within an ECS cluster.**
    - After you have created a task definition for your application within Amazon ECS, you can specify the number of tasks that will run on your cluster.

An ECS Services is:

- A service allows you to run and maintain a specified number (the "desired count") of simultaneous instances of a task definition in an ECS cluster.
  - Underneath which we define the desired number of Tasks to be run
  - Task Definisition -> Service executes the task a number of times -> end up with ECS containers equal to that number of times the service ran the task

source: aws.amazon.com

AWS ECS Container Registry (AWS ECR)

Amazon Elastic Container Registry

- Amazon ECR is a managed AWS Docker registry service. Customers can use the familiar Docker CLI to push, pull, and manage images.

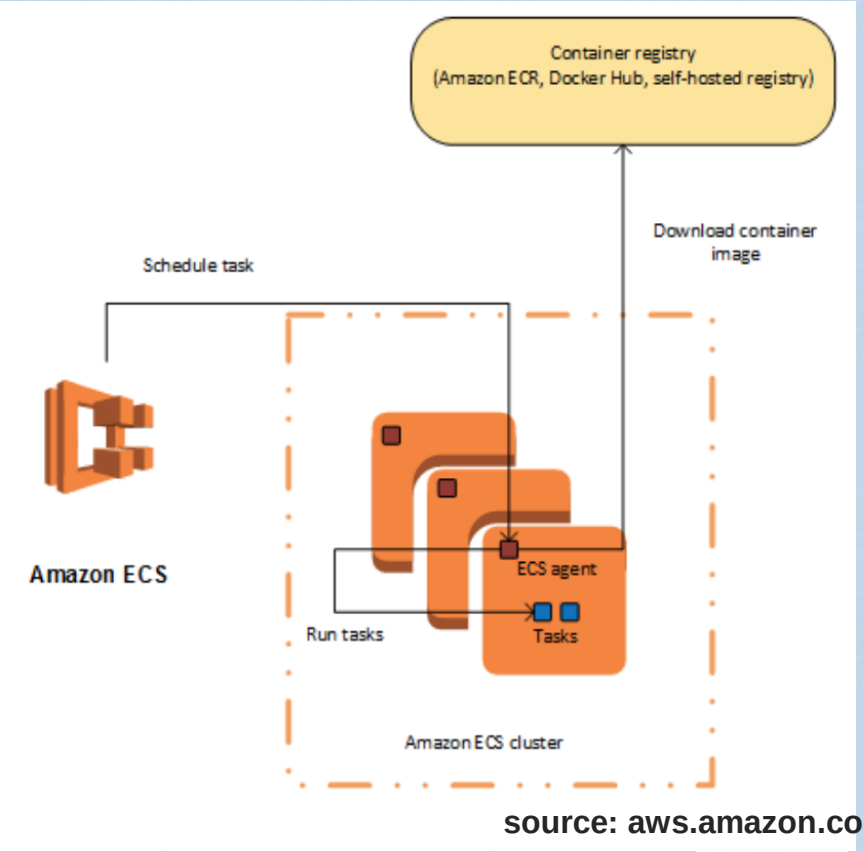
source: aws.amazon.co

AWS EC2 Container Service

AWS ECS – Container Agent

- **The *container agent*** runs on each infrastructure resource within an Amazon ECS cluster.

- It sends information about the resource's current running tasks and resource utilization to Amazon ECS, and starts and stops tasks whenever it receives a request from Amazon ECS.



source: aws.amazon.co

AWS ECS – IAM Roles and Task Roles

Amazon ECS IAM Policies, Roles, and Permissions

- Amazon ECS container instances make calls to the Amazon ECS and Amazon EC2 APIs on your behalf, so they need to authenticate with your credentials.
  - This authentication is accomplished by creating an IAM role for your container instances and associating that role with your container instances when you launch them.
- Basically, in Amazon ECS, IAM can be used to control access at the container instance level using IAM roles
  - This is required if you are to control using ECS Launch type

source: aws.amazon.co

AWS ECS – IAM Roles for Tasks

- You must create an IAM policy for your tasks to use that specifies the permissions that you would like the containers in your tasks to have.
  - You must also create a role for your tasks to use before you can specify it in your task definitions.
- You can create the role using the **Amazon EC2 Container Service Task Role** service role in the IAM console.

Benefits of Using IAM Roles for Tasks

- **Credential Isolation:** A container can only retrieve credentials for the IAM role that is defined in the task definition to which it belongs; a container never has access to credentials that are intended for another container that belongs to another task.
- **Authorization:** Unauthorized containers cannot access IAM role credentials defined for other tasks.
- **Auditability:** Access and event logging is available through CloudTrail to ensure retrospective auditing.
- Task credentials have a context of taskArn that is attached to the session, so CloudTrail logs show which task is using which role.

source: aws.amazon.co

Amazon Elastic Container Service

ECS Task Definitions and ALB

AWS Application Load Balancer

Load balancing to ECS hosted Micro-Services

- You can use a micro-services architecture to structure your application as services that you can develop and deploy independently.
- You can install one or more of these services on each EC2 instance, with each service accepting connections on a different port.
- You can use a single Application Load Balancer to route requests to all the services for your application.
- When you register an EC2 instance with a target group, you can register it multiple times;
  - For each service, register the instance using the port for the service.

AWS Application Load Balancer

Load balancing to ECS hosted Micro-Services

Service Load Balancing

- Your Amazon ECS service can optionally be configured to use Elastic Load Balancing to distribute traffic evenly across the tasks in your service.
- Application Load Balancers offer several features that make them particularly attractive for use with Amazon ECS services:
  - Application Load Balancers allow containers to use dynamic host port mapping
    - Such that multiple tasks from the same service are allowed per container instance

AWS Application Load Balancer

AWS ALB

The Application Load Balancer integrates with EC2 Container Service (ECS) using Service Load Balancing.

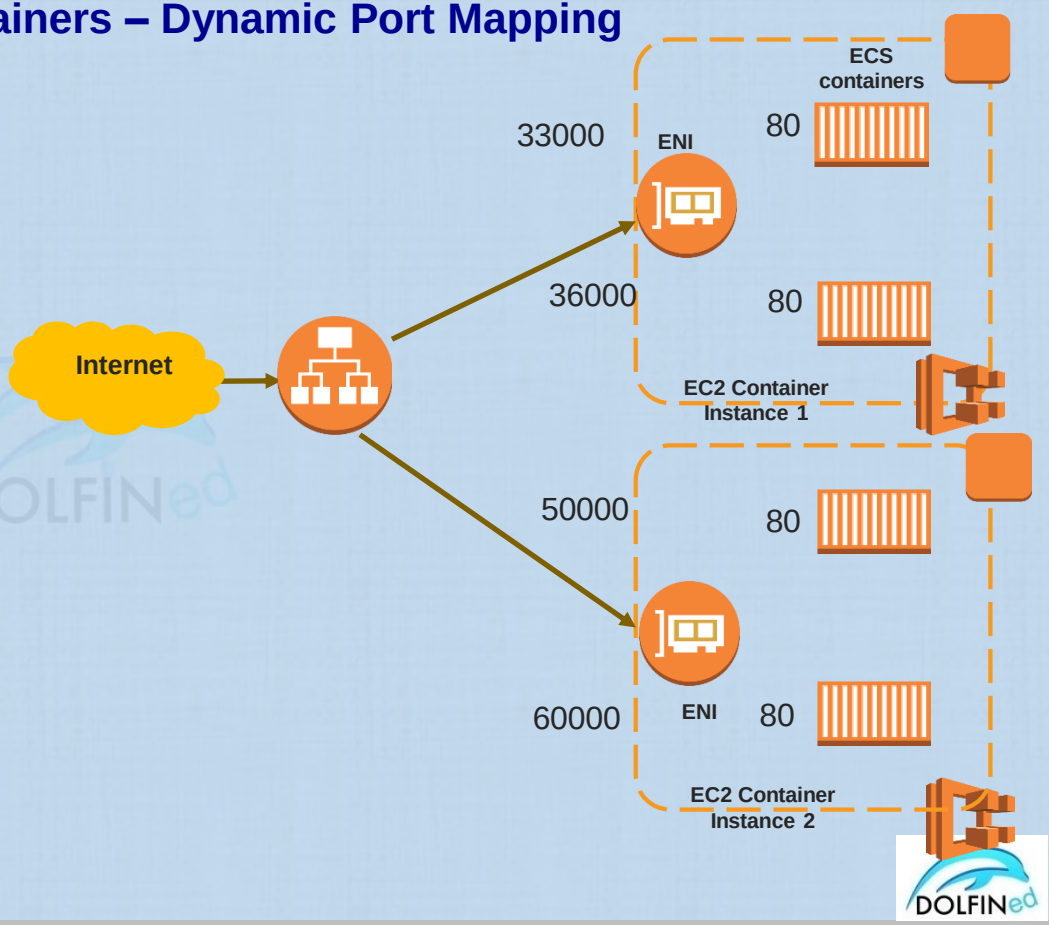
- ECS Instances can be registered with the ALB using multiple ports
  - This allows for requests to be routed to multiple containers on a single container instance
- Application Load Balancers *allow containers to use dynamic host port mapping*
- Amazon ECS will automatically register tasks with the ALB using a dynamic container-to-host port mapping
  - Such that multiple tasks (using the same port) from the same service are allowed per container instance
  - You can use dynamic port mapping to support multiple tasks from a single service on the same container instance.
  - This allows for dynamic mapping of services to ports as specified in the ECS task definition.
- The ECS task scheduler will automatically add these tasks to the ALB.
- In dynamic port mapping, Amazon ECS manages updates to your services by automatically registering and deregistering containers with the ALB using the instance ID and port for each container.

AWS Application Load Balancer

AWS ALB with Containers – Dynamic Port Mapping

- An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and **can route requests to one or more ports on each container instance in your cluster.**

- **Application Load Balancers support dynamic host port mapping**.
- If your task's container definition specifies port 80 for a container port, and port 0 for the host port, then the host port is dynamically chosen from the ephemeral port range of the container instance (such as 32768 to 61000 on the latest Amazon ECS-optimized AMI).



AWS LAMBDA

AWS Lambda

Introduction

AWS Lambda – How it works

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- With AWS Lambda, you can run code for virtually any type of application or backend service - **all with zero administration.**
- AWS Lambda manages all the administration
- AWS Lambda runs your code **on a high-availability compute infrastructure**
- AWS Lambda executes your code only when needed and **scales automatically,** from a few requests per day to thousands per second.
- You **pay only for the compute time you consume** – No charge when your code is not running.
- All you need to do is supply your code in the form of one or more Lambda functions to AWS Lambda, in one of the languages that AWS Lambda supports (currently Node.js, Ruby, Java, C#, GO, PowerShell, and Python), and the service can run the code on your behalf.
- AWS Lambda takes care of provisioning and managing the servers to run the code upon invocation.
- This is in exchange for flexibility, which means you cannot log in to compute instances, or customize the operating system or language runtime.
  - If you do want to manage your own compute, you can use EC2 or Elastic Beanstalk

source: aws.amazon.com

AWS Lambda – Triggers

- You can use AWS Lambda to run your code in response to:
  - Events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table,
  - To run your code in response to HTTP requests using Amazon API Gateway, or
  - Invoke your code using API calls made using AWS SDKs.

- With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB, process streaming data stored in Kinesis, or create your own back end that operates at AWS scale, performance, and security.

source: aws.amazon.co

When to use & Serverless Applications

- AWS Lambda is an ideal compute platform for many application scenarios, if you can :
  - Write your application code (You are responsible for your own code) in languages supported by AWS Lambda (that is, Node.js, Java, C# and Python),
  - and run within the AWS Lambda standard runtime environment and resources provided by Lambda.

- A typical serverless application consists of one or more Lambda functions triggered by events such as object uploads to Amazon S3, Amazon SNS notifications, and API actions.
  - Those functions can standalone or leverage other resources such as DynamoDB tables or Amazon S3 buckets.

- The most basic serverless application is simply a function.

source: aws.amazon.co

AWS Lambda – Building Blocks of a Lambda-Based Apps

- **Lambda function:**

The foundation, it is comprised of your custom code and any dependent libraries.

- **Event source:**

An AWS service, such as Amazon SNS, or a custom service, that triggers your function and executes its logic.

- **Downstream resources:**

An AWS service, such DynamoDB tables or Amazon S3 buckets, that your Lambda function calls once it is triggered.

- **Log streams:**

While Lambda automatically monitors your function invocations and reports metrics to CloudWatch, you can annotate your function code with custom logging statements that allow you to analyze the execution flow and performance of your Lambda function to ensure it's working properly.

- **AWS Serverless Application Model (AWS SAM)**

A model to define serverless applications. AWS SAM is natively supported by AWS CloudFormation and defines simplified syntax for expressing serverless resources.

source: aws.amazon.co

AWS Lambda

- Configuration
- Invocation

AWS Lambda Functions Configuration

Lambda function configuration information includes the following key elements:

- **Compute resources that you need**
  - You only specify the amount of memory you want to allocate for your Lambda function (128MB to 3018MB).
    - AWS Lambda allocates CPU power proportional to the memory

- **Maximum execution time (timeout)**
  - You pay for the AWS resources that are used to run your Lambda function.
  - To prevent your Lambda function from running indefinitely, you specify a timeout.
  - When the specified timeout is reached, AWS Lambda terminates your Lambda function.
  - Default is 3 seconds; maximum is 900 seconds (15 minutes)

- **IAM role (execution role)**
  - This is the role that AWS Lambda assumes when it executes the Lambda function on your behalf.

source: aws.amazon.co

AWS Lambda Function – Services it can access

- Lambda functions can access :
  - AWS Services or non-AWS services
  - AWS Services running in AWS VPCs (Ex. Redshift , ElastiCache, RDS instances)
  - Non-AWS Services running on EC2 instances in an AWS VPC
    - Additional configuration will be required for VPC access (Security group and subnet IDs)

- AWS Lambda runs your function code securely within an AWS internal VPC by default.
  - This VPC has connectivity to AWS services and the internet.

- To enable your Lambda function to access resources inside your VPC:
  - Provide additional VPC-specific configuration information that includes VPC subnet IDs and security group IDs
  - Lambda will then create an ENI for each subnet and security group attached to the Lambda function
  - Running Lambda functions this way is slower.

source: aws.amazon.com

AWS Lambda Functions – Invocation

- When building applications on AWS Lambda, including serverless applications, the core components are Lambda functions and event sources.
- An event source is the AWS service or custom application that publishes events,

The use cases for AWS Lambda can be grouped into the following categories:

- **Using AWS Lambda with AWS services as event sources**
  - Event sources publish events that cause the Lambda function to be invoked.
- **On-demand Lambda function invocation over HTTPS** (Amazon API Gateway)
  - You can also invoke your Lambda function over HTTPS.
  - You can do this by defining a custom REST API and endpoint using API Gateway.
- **Invocation types:**
  - **On-demand Lambda function invocation** (build your own event sources using custom apps)
  - **Scheduled events**
    - You can also set up AWS Lambda to invoke your code on a regular, scheduled basis using the AWS Lambda console.

source: aws.amazon.com

AWS Services

AWS Lambda – Supported AWS event sources for Lambda Functions

- **Amazon S3**
- **Amazon DynamoDB**
- **Amazon Kinesis Streams**
- **Amazon Simple Notification Service**
- **Amazon Simple Email Service**
- Amazon Cognito
- **AWS CloudFormation**
- **Amazon CloudWatch Logs**
- **Amazon CloudWatch Events**
- AWS CodeCommit
- **Scheduled Events (powered by Amazon CloudWatch Events)**
- AWS Config
- Amazon Alexa
- Amazon Lex
- **Amazon API Gateway**
- AWS IoT Button
- **Amazon CloudFront**
- **Amazon Kinesis Firehose**
- **Amazon SQS**
- Other Event Sources: Invoking a Lambda Function On Demand

source: aws.amazon.co

AWS Lambda – Invoking Lambda Functions On-demand

Invoking a Lambda Function On Demand

- In addition to invoking Lambda functions using event sources, you can also invoke your Lambda function on demand.
- You don't need to preconfigure any event source mapping in this case.
- The custom application must have the necessary permissions to invoke your Lambda function.
  - For example, user applications can also generate events (build your own custom event sources)
  - User applications such as client, mobile, or web applications can publish events and invoke Lambda functions using the AWS SDKs or AWS Mobile SDKs such as the AWS Mobile SDK for Android

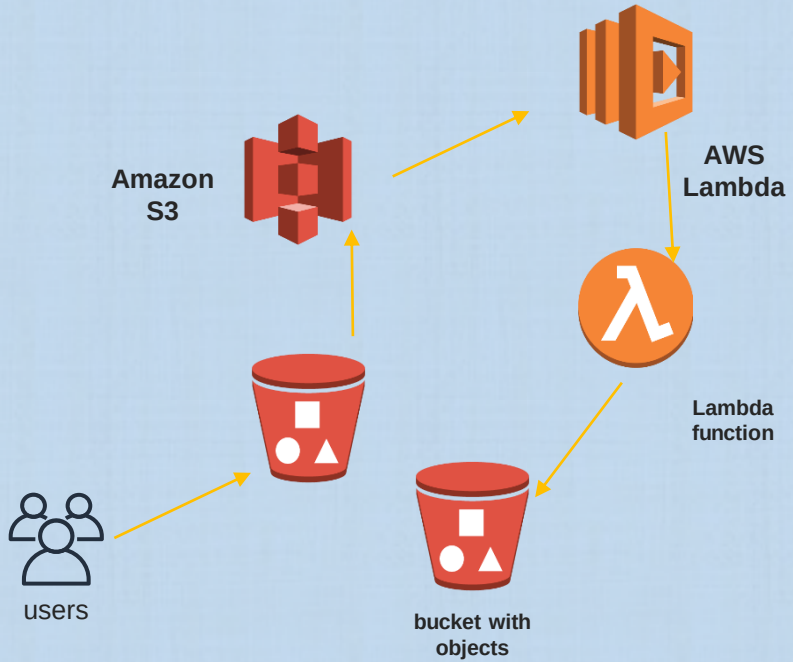
source: aws.amazon.co

AWS Lambda

Use Cases and Examples

AWS Lambda – File processing use case

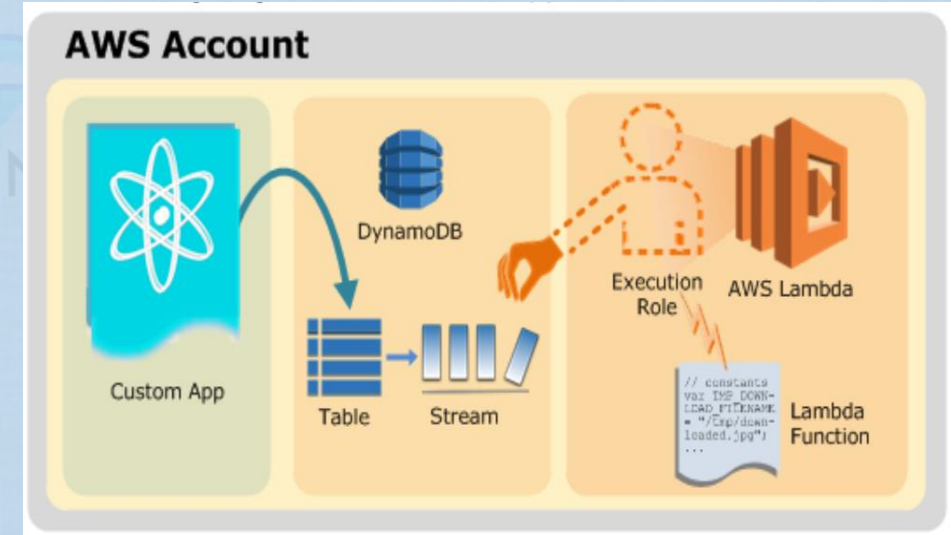
- You have a photo sharing application. People use your application to upload photos, and the application stores these user photos in an Amazon S3 bucket.
- Then, your application creates a thumbnail version of each user's photos and displays them on the user's profile page. A Lambda function is created which will create a thumbnail automatically.
- Amazon S3 is one of the supported AWS event sources that can publish object-created events and invoke your Lambda function.
- Your Lambda function code can read the photo object from the S3 bucket, create a thumbnail version, and then save it in another S3 bucket.



source: aws.amazon.co

AWS Lambda – Data and Analytics use case

- You are building an analytics application and Storing raw data in a DynamoDB table.
- Custom app updates the DynamoDB table.
- Amazon DynamoDB publishes item updates to the stream.
- AWS Lambda polls the stream and invokes your Lambda function when it detects new records in the stream.
- AWS Lambda executes the Lambda function by assuming the execution role you specified at the time you created the Lambda function.

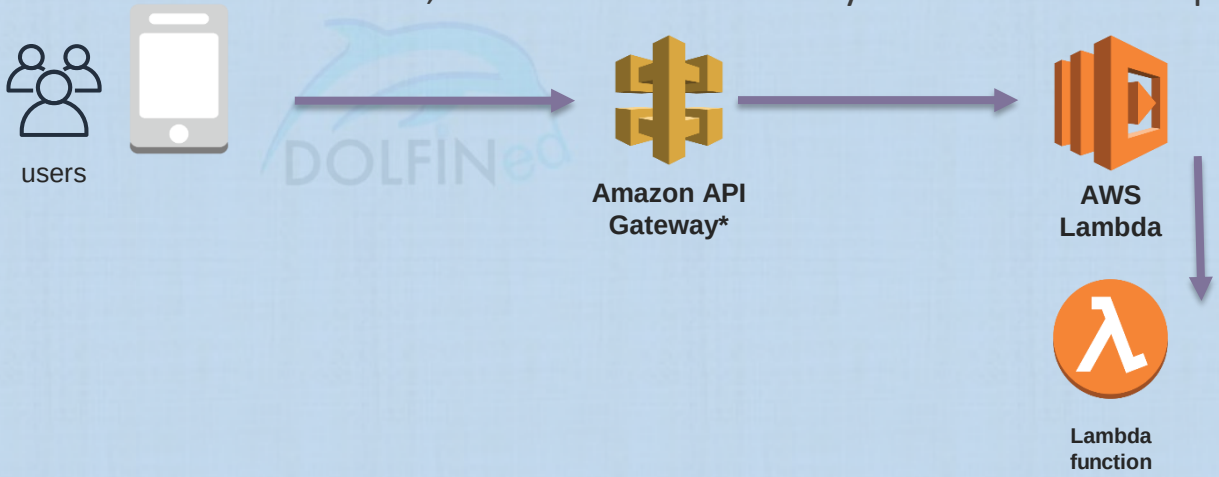


source: aws.amazon.co

AWS Lambda – Website use case

Websites

- Suppose you are creating a website and you want to host the backend logic on Lambda.
- You can invoke your Lambda function over HTTP using Amazon API Gateway as the HTTP endpoint.
- Now, your web client can invoke the API, and then API Gateway can route the request to Lambda.



source: aws.amazon.co

AWS Lambda

Scaling & Limits & Monitoring

AWS Lambda - Scaling

- AWS Lambda will dynamically scale capacity in response to increased traffic, subject to your account's Account Level Concurrent Execution Limit

- To handle any burst in traffic, Lambda will immediately increase your concurrently executing functions by a predetermined amount, dependent on which region it's executed

- Lambda depends on Amazon EC2 to provide Elastic Network Interfaces for VPC-enabled Lambda functions, these functions are also subject to Amazon EC2's rate limits as they scale.

source: aws.amazon.co

AWS Lambda – Understanding Scaling Behavior

- Concurrent executions refers to the number of executions of your function code that are happening at any given time.

- **Event sources that aren't stream-based**
  - If you create a Lambda function to process events from event sources that aren't stream-based
    - Each published event is a unit of work, in parallel, up to your account limits.
    - **This means one Lambda function invocation per event**
    - Therefore, the number of events (or requests) these event sources publish influences the concurrency.

- **Stream-based event sources**
  - For Lambda functions that process Kinesis or DynamoDB streams the number of shards is the unit of concurrency.
  - If your stream has 100 active shards, there will be at most 100 Lambda function invocations running concurrently.

source: aws.amazon.co

AWS Lambda – Resource Limits per Invocation

Resource	Limits
Memory allocation range	Minimum = 128 MB / Maximum = 3008 MB (with 64 MB increments). If the maximum memory use is exceeded, function invocation will be terminated.
Ephemeral disk capacity ("/tmp" space)	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds

Also there is a 1000 concurrent executions limit per account (soft limit)

source: aws.amazon.co

AWS Lambda Monitoring

Using Amazon CloudWatch

- AWS Lambda automatically monitors Lambda functions on your behalf, reporting metrics through Amazon CloudWatch.

Tracing Lambda-Based Applications with AWS X-Ray

- AWS X-Ray is an AWS service that allows you to detect, analyze, and optimize performance issues with your AWS Lambda applications.

Logging AWS Lambda API Calls By Using AWS CloudTrail

- AWS Lambda is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of AWS Lambda in your AWS account and delivers the log files to an Amazon S3 bucket that you specify.

source: aws.amazon.co

LAMBDA@EDGE

AWS Lambda @ Edge

Introduction

Lambda@Edge

- It is a compute service that enables the execution of Lambda functions which facilitates the customization of the content delivered by CloudFront.
- This is achievable by authoring Node.js functions in one Region, US-East-1 (N. Virginia), and then executing them in AWS locations globally, that are closer to the viewer, without provisioning or managing servers.
- Lambda@Edge scales automatically, from a few requests per day to thousands per second.
- Processing requests at AWS locations closer to the viewer instead of on origin servers significantly reduces latency and improves the user experience.
- When a CloudFront distribution is associated with a Lambda@Edge function, CloudFront intercepts requests and responses at CloudFront edge locations.

Lambda@Edge – When can it be invoked?

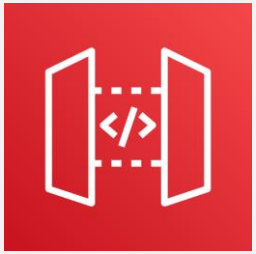
- A Lambda@Edge trigger is one combination of CloudFront distribution, cache behavior, and event that causes a function to execute.
- CloudFront events that can trigger a Lambda@Edge function
  - Lambda functions can be executed when the following CloudFront events occur:
    - Viewer Requests: When CloudFront receives a request from a viewer
    - Origin Request: Before CloudFront forwards a request to the origin
    - Origin Response: When CloudFront receives a response from the origin
    - Viewer Response: Before CloudFront returns the response to the viewer
- For each CloudFront cache behavior, you can add up to 4 triggers that will invoke the lambda function defined
- Each Lambda@Edge function must contain the callback parameter to successfully process a request or return a response.

Lambda@Edge – Use cases

- Use in A/B testing:
  - A Lambda@Edge function can inspect cookies and rewrite URLs so that users see different versions of a site (when testing two versions of your website).
    - This avoids the need to create redirects or change the URL.
    - This can be done by:
      - Having the Lambda@Edge function set cookies when CloudFront receives a request,
      - The Lambda@edge function will assign the user to version A or B randomly, and then
      - It returns the corresponding version to the viewer.
- A Lambda@Edge function can change the value of a response header based on the value of another header
- A Lambda@Edge function can make network calls to external resources to confirm user credentials, or fetch additional content to customize a response.
- A Lambda@Edge function can inspect headers or authorization tokens, and insert a header to control access to the content before CloudFront forwards the request to the origin.

Lambda@Edge – Use cases (cont.)

- A Lambda@Edge function can check cookies for other criteria.
  - Ex. : On an online website that sells apparel, if you use cookies to indicate which color a user chose for a dress, a Lambda function can change the request so that CloudFront returns the image of a dress in the selected color.
- A Lambda@Edge function can generate HTTP responses when CloudFront viewer request or origin request events occur.
  - It can also replace the HTTP response in origin and viewer response events.
  - Generating an HTTP Redirect
- By inspecting the request headers, Lambda functions can be used to return a response based on the client's device or a redirect to a country specific URL
- Normalizing Query String Parameters to Improve the Cache Hit Ratio
  - For example, use a Lambda@Edge function to change query strings before CloudFront forwards requests to the origin



AMAZON API GATEWAY

Amazon API Gateway

Introduction

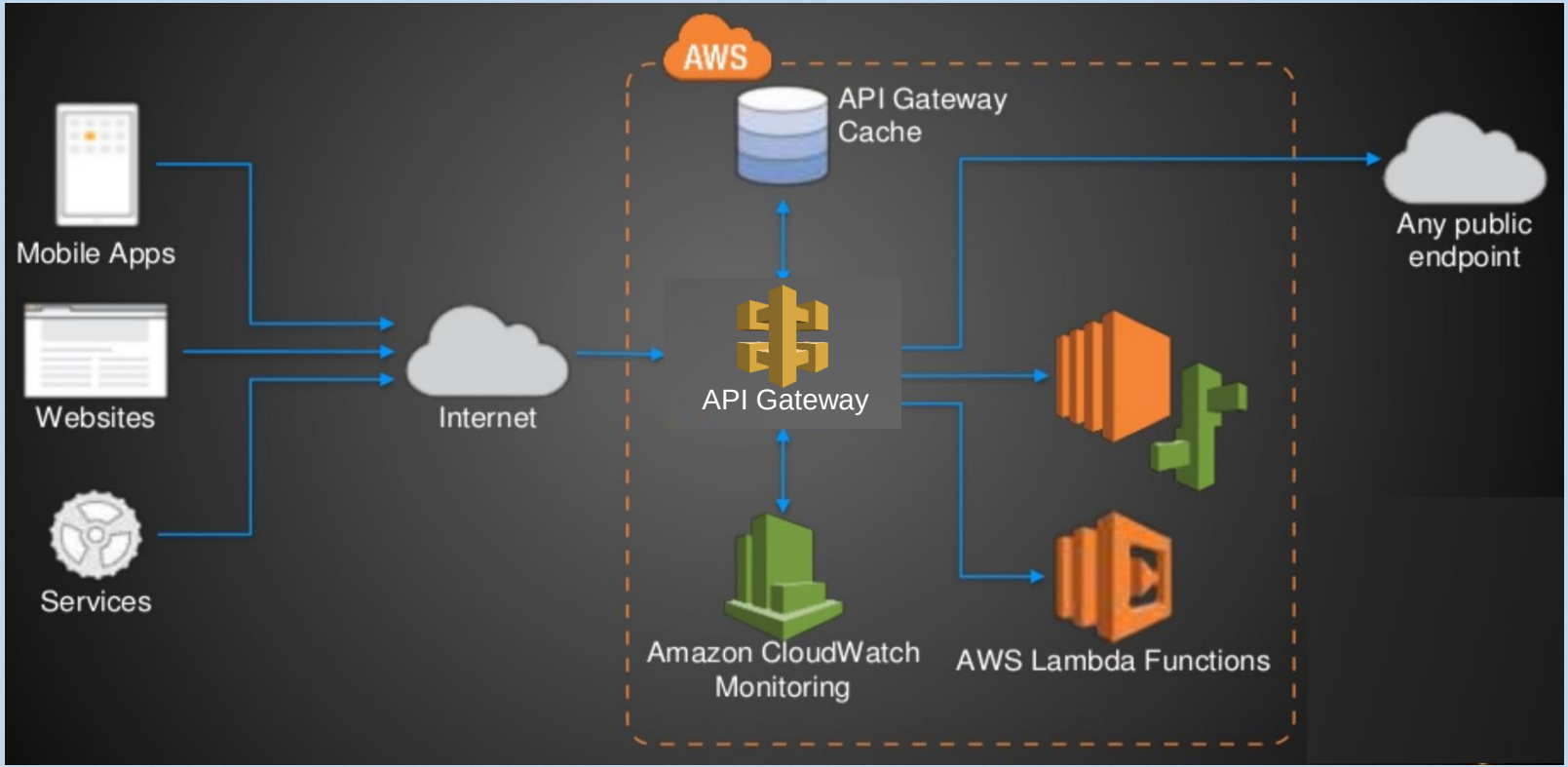
API Gateway

- Amazon API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs **at any scale.**
  - Together with AWS Lambda, API Gateway forms the app-facing part of the AWS serverless infrastructure.
- Amazon API Gateway handles all of the tasks involved in accepting and processing **up to hundreds of thousands of concurrent API calls,** including traffic management, authorization and access control, monitoring, and API version management.
- An API gateway API is a collection of resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. The collection can be deployed in one or more stages.
- Permissions to invoke a method are granted using IAM roles and policies or API Gateway custom authorizers. An API can present a certificate to be authenticated by the backend.
- Typically, API resources are organized in a resource tree according to the application logic.
  - Each API resource can expose one or more API methods that must have unique HTTP verbs supported by API Gateway.

source: aws.amazon.com

AWS Services : API Gateway

API Gateway



source: aws.amazon.co

API Gateway

- With Amazon API Gateway, you can provide your clients with a consistent and scalable programming interface to access three types of endpoints in the backend:
  - Invoking AWS Lambda functions,
  - Calling other AWS services, and
  - Accessing an HTTP website or webpage.
- To do this, you create an API Gateway API to integrate each API method with a backend endpoint.
  - Each backend endpoint is associated with an API Gateway integration type.

- Gateway Methods:
  - Each resource within a REST API can support one or more of the standard HTTP methods.
  - You define which verbs should be supported for each resource (GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS) and their implementation.

source: aws.amazon.co

API Gateway Methods

HTTPS endpoints

- All of the APIs created with Amazon API Gateway expose (to the Clients) HTTPS endpoints only.
  - Amazon API Gateway does not support unencrypted (HTTP) endpoints with the clients.
- By default, Amazon API Gateway assigns an internal domain to the API that automatically uses the Amazon API Gateway certificate.
  - When configuring your APIs to run under a custom domain name, you can provide your own certificate for the domain.

source: aws.amazon.co

API Gateway – API endpoints (Hostnames of Deployed APIs)

API Gateway supports the following types of API endpoints,

Edge-optimized API endpoint:

- Relies on Amazon **Cloudfront** distributions**, is the default endpoint types**
- API requests are routed to the nearest CloudFront Point of Presence (POP) which typically improves connection time for geographically diverse clients.

Regional API endpoint:

- It is intended to serve clients, such as EC2 instances, in the same AWS region where the API is deployed.
- Together with Route53 latency-based routing, regional endpoints enable an API developer to deploy an API to multiple regions using the same regional API endpoint configuration

Private API endpoint:

- Runs inside a VPC. Example backend services are EC2, EC2, ELB services and Lambda.

source: aws.amazon.co

API Gateway

Backend Services

- Amazon API Gateway can execute AWS Lambda functions in your account,
- It can start AWS Step Functions state machines, or
- Call HTTP endpoints hosted on AWS Elastic Beanstalk, Amazon EC2, and non-AWS hosted HTTP based operations that are accessible via the public Internet.

- API Gateway also allows you to specify a mapping template to generate static content to be returned, helping you mock your APIs before the backend is ready (response from within the API itself – Mock Integration)

- You can also integrate API Gateway with other AWS services directly –
  - For example, you could expose an API method in API Gateway that sends data directly to Amazon Kinesis.

source: aws.amazon.com

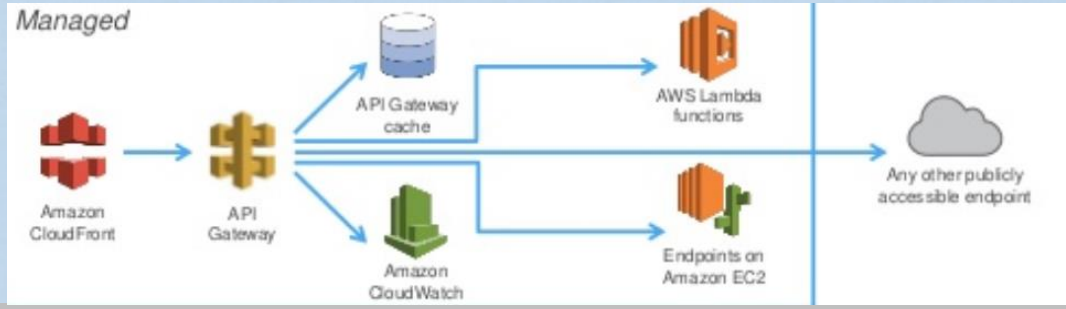
Amazon API Gateway

Features and Benefits

AWS Services – API Gateway Features

API Gateway Benefits/Features

- Robust, secure, and scalable access to backend APIs and Hosts **multiple versions** and **release stages** of your APIs
- Create and distribute **API Keys** to developers
- Use of **AWS Sig-v4** to authorize access to APIs
- **Throttle** and Monitor requests to protect your backend
- Integrates with X-Ray, WAF, CloudWatch and CloudTrail for Protection, Troubleshooting, monitoring and logging
- Manage **Cache** to store API responses
- **SDK Generation** for iOS, Android, and JavaScript
- Reduced Latency and Distributed Denial of Service protection by using CloudFront
- Request/Response data transformation and API mocking
- Swagger support
- Open APIs, API Keys, Usage Plans for 3rd party API developers



API Gateway Features – Throttling/Caching/Scaling

Resiliency

- **Through Throttling rules,** Amazon API Gateway helps you manage traffic to your back-end systems via throttling rules that are based on the number of requests per second, for each HTTP method (GET, PUT..) in your APIs.

Caching:

- You can set up a cache with customizable keys and **time-to-live (TTL)** in seconds for your API data to avoid hitting your back-end services for each request.
  - Enhanced response times and reduces load on backend services

Scaling:

- Amazon API Gateway handles any level of traffic received by an API, so you are free to focus on your business logic and services rather than maintaining infrastructure.

source: aws.amazon.co

API Gateway Features – API Versions

API Lifecycle Management -

Multiple Versions of the same REST API

- Amazon API Gateway lets you run multiple versions of the same API simultaneously so that applications can continue to call previous API versions even after the latest versions are published.
- Amazon API Gateway gives you the **ability to clone an existing API to create a new version**
- You can determine which version of the API is being accessed/used.

API Lifecycle Management

Multiple Release Stages:

- Amazon API Gateway also helps you manage multiple release stages for each API version, such as alpha, beta, and production.
  - Each API stage can be configured to interact with different backend endpoints based on your API setup.
  - Specific stages and versions of an API can be associated with a custom domain name and managed through Amazon API Gateway.

source: aws.amazon.co

Amazon API Gateway

Monitoring & X-Ray Integration

API Gateway – API Operations and Monitoring

Monitoring through API Gateway dashboard:

- After an API is deployed and in use, Amazon API Gateway provides you with a REST API dashboard to visually monitor calls to the services.

- API Gateway also meters utilization by third-party developers, the data is available in the API Gateway console and through the APIs.

Monitoring through CloudWatch:

- The Amazon API Gateway logs (near real time) back-end performance metrics such as API calls, Latency, and error rates to AWS CloudWatch in your account
  - This allow for the set up of custom CloudWatch alarms on Amazon API Gateway APIs

source: aws.amazon.co

Using Amazon X-RAY to debug/monitor API Gateway

- X-Ray helps in tracing API Gateway API Execution
- AWS X-Ray can be used to trace and analyze user requests as they travel through your Amazon API Gateway APIs to the underlying services.
- API Gateway supports AWS X-Ray tracing for all API Gateway endpoint types: regional, edge-optimized, and private.
- You can use AWS X-Ray with Amazon API Gateway in all regions where X-Ray is available.
- By obtaining the end-to-end view of an entire request by X-Ray, analyzing latencies in the respective APIs and their backend services can be achieved.
- You can use an X-Ray service map to view the latency of an entire request and that of the downstream services that are integrated with X-Ray.
- You can enable X-Ray for an API stage by using the API Gateway console, or by using the API Gateway API or CLI.

Amazon API Gateway

Permissions and access control to APIs

Controlling Access to a REST API in API Gateway

API Gateway supports multiple mechanisms for controlling access to your API:

- Resource policies
- Standard AWS IAM roles and policies
- Cross-origin resource sharing (CORS)
- Lambda authorizers
- Amazon Cognito user pools
- Client-side SSL certificates
  - Can be used to verify that HTTP requests to your backend system are from API Gateway.
- Usage plans
  - Let you provide API keys to your customers — and then track and limit usage of your API stages and methods for each API key.

API Gateway – Cross Account Access to APIs

When AWS identity and access management is enabled on a specific resource,

- IAM users from different AWS accounts cannot access that resource unless the caller is allowed to assume the resource owner's role,
  - i.e API Gateway does not currently support cross-account authentication.

source: aws.amazon.co

Control Access to an API with Amazon API Gateway Resource Policies

- Amazon API Gateway resource policies, attached to an API, can control whether a specified principal are authorized to invoke the API.
- API Gateway resource policies can be used to control whether invoking the API is allowed by:
  - IAM Users, IAM Roles, or Service roles from a specified AWS account, same account or cross-account
  - Specified source IP address ranges or CIDR blocks
  - Specified virtual private clouds (VPCs) or VPC endpoints (in any account)
- Resource policies can be used with private, edge-optimized, and regional API endpoints.
- Resource policies can be attached to an API using AWS console, AWS CLI, or AWS SDKs.
- API Gateway resource policies can be used together with IAM policies.

API Gateway – AWS Authorization

With Amazon API Gateway, you can optionally set your API methods to require authorization.

- IAM:
  - To authorize and verify API requests to AWS services, API Gateway can leverage signature version 4
  - Using signature version 4 authentication, you can use Identity and Access Management (IAM) and access policies to authorize access to your APIs and all your other AWS resources.

- Lambda Authorizers:
  - You can also use AWS Lambda functions to verify and authorize bearer tokens such as JWT tokens or SAML assertions.

- Amazon Cognito User Pools (No authorization – only authentication)
  - You can retrieve temporary credentials associated with a role in your AWS account using Amazon Cognito.

source: aws.amazon.co

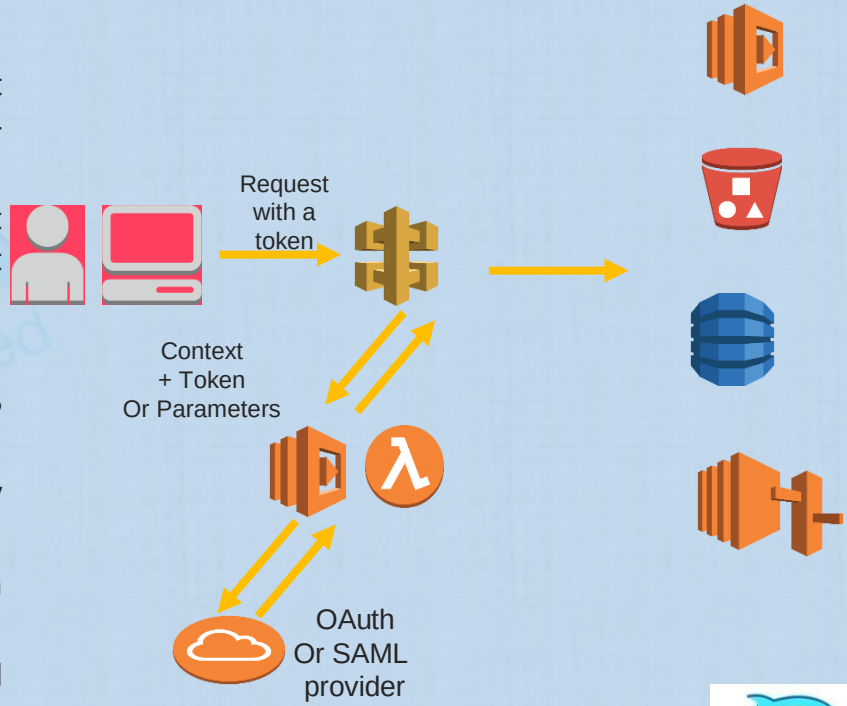
API Gateway Lambda Authorizers

When a client calls your API, API Gateway verifies whether a Lambda authorizer is configured for the API method.

- If so, API Gateway calls the Lambda function.
- In this call, API Gateway supplies the authorization token that is extracted from a specified request header for the token-based authorizer,
- or it passes in the incoming request parameters as the input (for example, the event parameter) to the request parameters-based authorizer function.

Configure Cross-Account Lambda Authorizer

- You can use an AWS Lambda function from a different AWS account as your API authorizer.
- Each account can be in any region where Amazon API Gateway is available.
- The Lambda authorizer function can use bearer token authentication strategies such as OAuth or SAML.
- This makes it easy to centrally manage and share a central Lambda authorizer function across multiple API Gateway APIs.



API Gateway Lambda Authorizers

- Various authorization strategies can be implemented, such as:
  - JSON Web Token (JWT) verification and
  - OAuth provider callout.

- A custom scheme, can also be implemented, based on incoming request parameter values, to return IAM policies that authorize the request.
- If the returned policy is invalid or the permissions are denied, the API call fails.
- API Gateway caches the returned policy (if valid), associated with the incoming token or identity source request parameters.
  - You can set the TTL period to zero seconds to disable the policy caching.
  - The default TTL value is 300 seconds.
  - Currently, the maximum TTL value of 3600 seconds cannot be increased.

Control Access to APIs Using Amazon Cognito User Pools as Authorizer

- Let you create customizable authentication and authorization solutions for your REST APIs.
- As an alternative to using IAM roles and policies or Lambda authorizers, you can use an Amazon Cognito user pool to control who can access your API in Amazon API Gateway.
- To use an Amazon Cognito user pool with your API, you must first create an authorizer of the COGNITO_USER_POOLS type and then configure an API method to use that authorizer.
- After the API is deployed, the client must first sign the user in to the user pool, obtain an identity or access token for the user, and then call the API method with one of the tokens, which are typically set to the request's Authorization header.
- The API call succeeds only if the required token is supplied and the supplied token is valid, otherwise, the client isn't authorized to make the call because the client did not have credentials that could be authorized.

SERVERLESS APPLICATION MODEL (SAM)

AWS SAM

Introduction, components & Benefits

What is it?

- A serverless application
  - It is a combination of Lambda functions, event sources, and **other resources** that work together to perform tasks.
  - A serverless application is more than just a Lambda function
    - It can include additional resources such as APIs, API Gateway, DataBases, S3 Buckets, DynamoDB tables, and event source mappings.

- AWS Serverless Application Model (AWS SAM) is an open-source framework that can be used to simplify building serverless applications on AWS.

- AWS SAM can be used to define (Package, Deploy and Troubleshoot errors) serverless applications on AWS.

- Is an extension of AWS Cloudfromation

AWS SAM Components

- AWS SAM template specification.
  - The template specification is in YAML or JSON
  - It is used to define the serverless application it represents.
    - It provides a simple and clean syntax to describe the functions, APIs, permissions, configurations, and events that make up a serverless application.
    - An AWS SAM template file is used to operate on a single, deployable, versioned entity that's your serverless application.

- AWS SAM command line interface (AWS SAM CLI).
  - It is a tool that can be used to build serverless applications that are defined by AWS SAM templates.
    - The CLI provides commands that validates that AWS SAM template files are written according to the specification, It can be used, among other things, to:
      - Define, package, test, and deploy serverless applications
      - Invoke Lambda functions locally (for local testing),
      - Test applications locally before deploying to AWS
      - Step-through debug Lambda functions,

Benefits of Using AWS SAM

- AWS SAM integrates with other AWS services; this provides the following benefits when creating serverless applications with AWS SAM:
  - **Single-deployment configuration**
    - AWS SAM makes it easy to organize the serverless application's components, resources and operate on a single stack.
    - AWS SAM can be used to share configuration (such as memory and timeouts) between resources, and deploy all related resources together as a single, versioned entity.
  - **AWS SAM is an extension of AWS Cloudformation:**
    - AWS SAM uses the reliable deployment capabilities of AWS CloudFormation.
    - The full suite of Cloudformation resources, intrinsic functions, and other template features are accessible to AWS SAM templates.
- **Local debugging and testing**
  - The AWS SAM CLI allows for locally building, testing, and debugging serverless applications that are defined by AWS SAM templates.

Benefits of Using AWS SAM

- **Built-in best practices:**
  - AWS SAM to define and deploy the serverless application's infrastructure as config.
    - This facilitates the enforcement of best practices, such as code reviews.
    - With a few lines of configuration, safe deployments can be enabled through CodeDeploy,
    - X-Ray can be used to enable tracing.
- **Deep integration with AWS Development tools:**
  - AWS SAM can be used with a suite of AWS tools to build serverless applications.
    - New applications in the AWS Serverless Application Repository can be discovered
    - For authoring, testing, and debugging AWS SAM–based serverless applications, you can use the AWS Cloud9.
    - To build a deployment pipeline for serverless applications, use CodeBuild, CodeDeploy, and CodePipeline.
    - AWS CodeStar can be used to get started with a project structure, code repository, and a CI/CD pipeline that's automatically configured.

AWS SAM

- Using AWS SAM
- AWS SAM Template Concepts
- AWS Resources that can be declared in an AWS SAM Template
- Controlling access to API Gateway APIs in AWS SAM

Using AWS SAM

- **Initialize the application**
  - Sample template and app code or create your own.
- **Test Locally.**
  - Test the application locally using sam local invoke and/or sam local start-api.
- **Package.**
  - When satisfied with the application components (Lambda, API , DynamoDB…etc)
    - Bundle the components (Lambda functions for ex,) , AWS SAM template, and any dependencies into an AWS CloudFormation deployment package using sam package.
    - The **deployment package** can be used to deploy the application to AWS
  - An S3 bucket will be required at this step (new or existing), it will be used to save the packaged code.
- **Deploy.**
  - Deploy the application (using the deployment package created) to AWS using sam deploy.
    - Testing the application in the AWS Cloud can be carried out at this stage

AWS Resources that can be declared in an AWS SAM Template

- AWS SAM defines the following resources that are specifically designed for serverless applications:
  - AWS::Serverless::Api for **API Gateway resource**
  - AWS::Serverless::Application
    - This can be used to embed a serverless **application from the AWS Serverless Application Repository or from an Amazon S3 bucket as a nested application.**
  - AWS::Serverless::Function **for Lambda function**
  - AWS::Serverless::LayerVersion
    - This resource type creates **a Lambda layer version (LayerVersion**) that contains library or runtime code that's needed by a Lambda function
  - AWS::Serverless::SimpleTable **for DynamoDB tables**
    - This resource type provides simple syntax for describing how to create DynamoDB tables

Controlling Access to the API Gateway APIs in AWS SAM

- AWS SAM can be used to control who can access the API Gateway APIs by enabling authorization within the AWS SAM template.
  - AWS SAM supports a few mechanisms for controlling access to your API Gateway APIs:
    - Lambda authorizers.
    - Amazon Cognito user pools.
      - Amazon Cognito user pools are user directories in Amazon Cognito.
      - A client that would like to access the API must first sign a user in to the user pool and obtain an identity or access token for the user.

Automating Deployments

- AWS SAM can be used with several other AWS services to automate the deployment process of serverless application.
  - CodeBuild:
    - Use CodeBuild to build, locally test, and package serverless application.
  - CodeDeploy (Comes built in with AWS SAM to ensure safe Lambda function deployment):
    - Use CodeDeploy to gradually deploy updates to the serverless applications.
    - If enabled, gradual deployments of the application, new versions/aliases of Lambda functions are automated
  - CodePipeline:
    - Use CodePipeline to model, visualize, and automate the steps that are required to release serverless applications

AWS Serverless Application Repository

- Publishing Serverless Applications The AWS Serverless Application Repository is a service that hosts serverless applications that are built using AWS SAM.
- If you want to share serverless applications with others, you can publish them in the AWS Serverless Application Repository.
- You can also search, browse, and deploy serverless applications that have been published by others.

AWS SAM Use Case – Process Amazon S3 Events

- The application consists of a Lambda function that's invoked by an Amazon S3 object upload event source.

- This sample serverless application will:
  - Process object-creation events in Amazon S3.
  - For each image that's uploaded to a bucket, Amazon S3 detects the object-created event and invokes a Lambda function.
  - The Lambda function invokes Amazon Rekognition to detect text that's in the image.
  - It then stores the results returned by Amazon Rekognition in a DynamoDB table.

- This use case requires that AWS resources are created and IAM permissions are configured before you can test the Lambda function locally.
  - AWS CloudFormation will be leveraged to create the resources and configure the permissions, so they need not be done manually before testing the Lambda function locally.

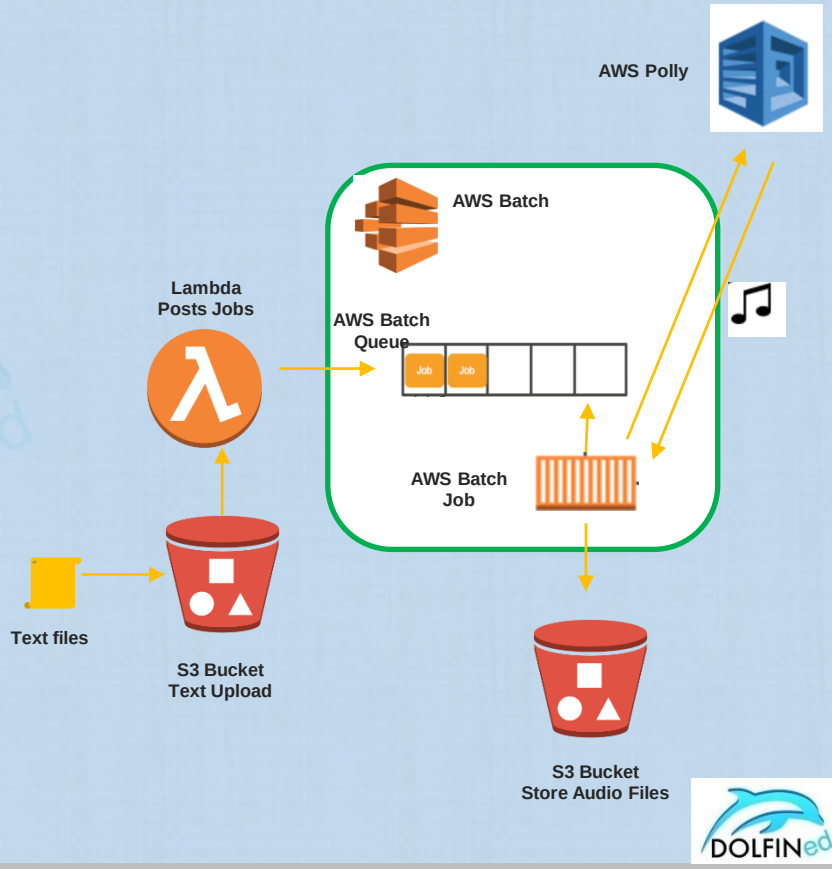
AWS BATCH

AWS Batch

- What it is
- AWS Batch Multi Node
- Components
- Integration with AWS CloudWatch Events
- Integration with AWS CloudTrail

AWS Batch - What is it?

- AWS Batch enables you to run batch computing workloads on the AWS Cloud regardless of the nature of the job.
  - AWS Batch creates and manages the compute resources in your AWS account, giving you full control and visibility.
  - AWS Batch removes the undifferentiated heavy lifting of configuring and managing the required infrastructure, as in traditional batch computing software.
- As a fully managed service, AWS Batch enables you to run batch computing workloads of any scale.
- AWS Batch supports any job that can be executed as a Docker container.
- Jobs specify their memory requirements and number of vCPUs.



AWS Batch - What is it?

- AWS Batch is a regional service that simplifies running batch jobs across multiple Availability Zones within a region.
  - AWS Batch compute environments can be created within a new or existing VPC.
- It uses on-demand and spot instances.
- This service can efficiently provision resources in response to jobs submitted in order to eliminate capacity constraints, reduce compute costs, and deliver results quickly.
- AWS Batch automatically provisions compute resources and optimizes the workload distribution based on the quantity and scale of the workloads.

- **Multi-node parallel jobs** allows for running single jobs that span multiple Amazon EC2 instances.
  - With AWS Batch multi-node parallel jobs, large-scale, tightly coupled, high performance computing applications and distributed GPU model training can be run on AWS without the need to launch, configure, and manage Amazon EC2 resources directly.

AWS Batch Components

- **Jobs**
  - A unit of work that you submit to AWS Batch.
  - Ex., a shell script, a Linux executable, or a Docker container image.
- **Job Definitions**
  - A job definition specifies how jobs are to be run; think of it as a blueprint for the resources in the job.
- **Job Queues**
  - An AWS Batch job is submitted to a particular queue where it stays until it gets scheduled on a compute environment
  - One or more compute environments can be associated with a job queue
  - Priorities can be assigned to job queues and compute environments
- **Scheduler**
  - Owned by AWS evaluated when and where to run Jobs that have been submitted to the Job Queues.
- **Compute Environment**
  - AWS Batch uses Amazon ECS container instances in its compute environments.
  - A compute environment is a set of managed or unmanaged compute resources that are used to run jobs.

AWS Batch – Permissions and IAM Roles

- Create IAM Roles for your Compute Environments and Container Instances
- Your AWS Batch compute environments and container instances require AWS account credentials to make calls to other AWS APIs on your behalf.
- You must create an IAM role that provides these credentials to your compute environments and container instances, then associate that role with your compute environments.

CloudWatch Events & CloudTrail

- AWS Batch Event Stream for CloudWatch Events
  - AWS Batch event stream for CloudWatch Events can be used to receive near real-time notifications regarding the current state of jobs that have been submitted to the job queues.
  - With AWS Batch and CloudWatch Events, scheduling and monitoring code that continuously polls AWS Batch for job status changes can be eliminated.
  - Instead, AWS Batch job state changes can be handled asynchronously using any CloudWatch Events target, such as AWS Lambda, Amazon SQS, Amazon SNS, or Kinesis Data Streams.
  - AWS Batch jobs are available as CloudWatch Events targets.

- AWS Batch is integrated with **AWS CloudTrail,** a service that provides a record of actions taken by a user, role, or an AWS service in AWS Batch.