Week - 2 Cloud Services & AWS Fundamentals

- Overview of AWS Global Infrastructure
- Detailed Overview of Elastic Compute Cloud (EC2)
- Setting Up Your First EC2 Instance
- In-Depth Guide to EC2 Instance Configuration
- Exploring EC2 Options in Detail
- Connecting to Cloud Instances
- Security Group Handling and Management
- Introduction to Amazon S3
- Auto Scaling and Load Balancing
- Understanding Cloud Formation and CloudWatch
- Exploring Simple Notification Service (SNS) and Simple Queue Service (SQS)
- Overview of Relational Database Service (RDS) and Identity and Access Management (IAM)
- Project-Based Learning: ECS and ECR
- Serverless Architecture
- Utilizing CloudWatch for Monitoring and Setting Billing Alarms

Hands-On Experience with AWS Services:

- AWS VPC
- AWS Lambda
- Amazon API Gateway
- Amazon SNS
- Amazon CloudFront
- AWS CloudFormation

Overview of AWS Global Infrastructure

AWS - Amazon Web Services is one of the most extensively used cloud service providers. AWS is one of the leading cloud service providers globally, offering a wide range of services, including computing power, storage, databases, machine learning, analytics, networking, content delivery, and more. These services allow customers to build and deploy applications quickly and securely without the need to invest in physical infrastructure. AWS's global infrastructure spans multiple geographic regions, each containing multiple availability zones, ensuring high availability, fault tolerance, and scalability for customers' applications and workloads.

Region: A region is a separate geographic area, typically consisting of multiple data centers. AWS operates in over 25 regions worldwide. Geographic regions contain multiple Availability Zones (AZs).

Availability Zones(AZ): Each region consists of multiple isolated locations called Availability Zones. These AZs are physically separate from each other but are interconnected through high-speed, low-latency networking. They provide redundancy and resilience against failures. AWS provides us the feasibility to launch our servers in multiple availability zones to achieve high availability/reliability

Edge Locations: AWS has a network of Edge Locations that serve content via its Content Delivery Network (CDN) service, Amazon CloudFront. These Edge Locations are spread globally to deliver content to users with lower latency.

Data Centers: AWS data centers are state-of-the-art facilities equipped with redundant power, networking, and cooling systems. They are designed to ensure high availability and security of customer data.

Amazon Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud (VPC) is a service that enables you to launch AWS resources in a logically isolated virtual network that you define. This isolation allows you to create a network environment that closely resembles a traditional network you might operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Key Components of VPC

- Subnets: Sub-divisions within a VPC that allow you to group resources based on security and operational needs. Subnets can be public (with access to the internet) or private (without direct internet access).

- Route Tables: These determine how network traffic is directed. Each subnet in your VPC must be associated with a route table, which contains rules (routes) that determine where network traffic from your subnet is directed.

- Internet Gateway: A horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the internet. It is used to allow public access to your instances.

- NAT Gateway: Network Address Translation (NAT) gateways enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating connections with those instances.

- Elastic IP Addresses: Static IP addresses designed for dynamic cloud computing. They can be associated with instances in your VPC for consistent communication with external systems.

- Endpoints: Enable you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.

- Security Groups: Act as virtual firewalls for your instances to control inbound and outbound traffic. Each instance in a VPC is associated with one or more security groups.

- Network ACLs (Access Control Lists): Provide an additional layer of security at the subnet level. They control inbound and outbound traffic to and from subnets.

- DHCP Options Sets: Define DHCP options for your VPC, such as domain name servers, domain name, and the NTP servers to be used by your instances.

- VPC Peering: Allows you to route traffic between VPCs using private IP addresses, enabling you to interconnect different VPCs, even across different AWS accounts and regions.

- VPN Connections: Virtual Private Network (VPN) connections allow you to securely connect your VPC to your on-premises network or other VPCs.

How VPC Works

- Create a VPC: Define the IP address range (CIDR block) for your VPC.
- Add Subnets: Create subnets within the VPC, dividing the IP address range into smaller segments.
- Configure Route Tables: Set up route tables to control the routing of traffic between subnets and to/from the internet.
- Set Up Gateways and Endpoints: Attach an internet gateway, NAT gateway, or endpoints as needed.
- Security Configuration: Define security groups and network ACLs to control access to instances and subnets.
- NAT Gateway: Allows the application servers to download updates and patches from the internet without exposing them directly.

Detailed Overview of Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Amazon Elastic Compute Cloud (EC2) is one of the core services provided by Amazon Web Services (AWS) and serves as a foundational building block for deploying virtual servers in the cloud. AWS is the first major cloud provider that supports Intel, AMD, and Arm processors, the only cloud with on-demand EC2 Mac instances, and the only cloud with 400 Gbps Ethernet networking.

Instances: EC2 instances are virtual servers running in the AWS cloud. Users can launch instances with different configurations, including CPU, memory, storage, and network capacity, according to the application requirements.

Instance Types: AWS offers a wide range of instance types optimized for various workloads. These include general-purpose instances, compute-optimized instances, memory-optimized instances, storage-optimized instances, and GPU instances, among others.

Amazon Machine Images(AMIs): AMIs are Preconfigured templates used to launch EC2 instances. Users can choose from a selection of pre-configured AMIs provided by AWS or create their own custom AMIs tailored to their specific needs.

Key Pairs: It gives the secured login information for the instances. A key Pair consists of a public key and a private key that is a set of security credentials used to prove your identity when connecting to an Amazon EC2 instance.

Security Groups: Security groups act as virtual firewalls for EC2 instances, controlling inbound and outbound traffic. Users can define rules to allow or deny specific types of traffic based on protocols, ports, and IP addresses.

Tags: Tags are the Metadata that can be created and assigned to the Amazon EC2 resources.

Amazon EBS volumes: EBS provides block-level storage volumes that users can attach to their EC2 instances. These volumes are highly available and durable, allowing users to store data persistently even after the instance is terminated.

Elastic IP addresses: EC2 instances can be associated with elastic IP addresses, which are static IP addresses that users can allocate and re-assign to instances as needed. This enables users to maintain a consistent IP address for their applications.

Virtual private clouds (VPCs): Amazon Virtual Private Cloud (Amazon VPC) is a virtual network environment that allows users to provision a logically isolated section of the AWS Cloud where they can launch AWS resources in a virtual network that they define. It is like we have created a private cloud on AWS.

Setting Up First EC2 Instance:

- **Sign in to AWS Console:** Log in to your AWS Management Console using your AWS account credentials at https://console.aws.amazon.com/ec2/.
- **Navigate to EC2 Dashboard:** Once logged in, navigate to the EC2 service by either typing "EC2" in the search bar or locating it under the "Compute" section in the AWS services menu.
- **Launch Instance:** Click on the "Launch Instance" button to begin the process.
- **Choose an Amazon Machine Image (AMI):** Select an AMI that suits your requirements. AWS offers a variety of preconfigured images, including different operating systems and software configurations.
- **Choose instance type:** Select the instance type based on your workload. Each instance type offers different compute, memory, and storage capacities.
- **Configure Instance details:** Configure additional settings such as the number of instances, networking options, and security groups.
- **Add Storage:** Specify the size and type of storage for your instance. You can choose between Amazon EBS volumes and instance store volumes.
- **Add Tags (Optional):** Add tags to your instance for easier management and organization.
- **Configure Security Group:** Set up security groups to control inbound and outbound traffic to your instance. Define rules to allow specific types of traffic.
- **Review and Launch:** Review your instance configuration and click "Launch" to start the instance.

Security Group Handling and Management

Security groups act as virtual firewalls for your EC2 instances, controlling inbound and outbound traffic. Each security group contains a set of rules that define which

traffic is allowed or denied. A security group controls the traffic that is allowed to reach and leave the resources that it is associated with. For example, after you associate a security group with an EC2 instance, it controls the inbound and outbound traffic for the instance.

Creating Security Groups: To create a security group, navigate to the EC2 dashboard in the AWS Management Console. Under the "Network & Security" section, select "Security Groups" and click "Create Security Group." Specify a name, description, and VPC for the security group.

Configuring inbound Rules: Define inbound rules to control incoming traffic to your instances. You can specify rules based on protocols (e.g., TCP, UDP), ports, IP ranges, or security group references. For example, you might allow SSH (port 22) traffic only from your IP address.

Configuring Outbound Rules: After setting up inbound rules, configure outbound rules to regulate outgoing traffic from your instances. Follow similar steps as configuring inbound rules, ensuring you define the appropriate protocols, port ranges, and destinations for outbound traffic.

Managing Security Groups:

Regular Review: Periodically review your security group configurations to ensure alignment with your organization's security policies. Remove outdated or unnecessary rules to minimize security risks.

Monitoring: Monitor traffic flow through your security groups to identify unauthorized access attempts or abnormal activity. Utilize AWS CloudTrail and VPC Flow Logs for capturing and analyzing network traffic.

Automation: Implement automation tools and scripts to streamline security group management tasks, such as rule updates and configuration changes, ensuring consistency across EC2 instance.

Compliance: Ensure security group configurations adhere to industry regulations and compliance standards relevant to your organization. Regularly audit and validate security configurations to maintain compliance.

Introduction to Amazon S3

Amazon Simple Storage Service (Amazon S3) is a scalable object storage service offered by Amazon Web Services (AWS). It provides developers and businesses with a secure, durable, and highly available storage solution for storing and retrieving

any amount of data over the internet. Amazon S3 is designed to deliver 99.999999999% (11 nines) durability and offers various features for data management, access control, and integration with other AWS services.

Key Features of Amazon S3:

1. **Scalability:** Amazon S3 scales automatically to accommodate any amount of data, from gigabytes to petabytes, without requiring capacity planning or upfront provisioning.
2. **Durability and Availability:** Amazon S3 stores data redundantly across multiple facilities and provides built-in fault tolerance, ensuring high durability and availability of stored objects.
3. **Data Protection:** Amazon S3 offers features such as server-side encryption, encryption in transit (SSL/TLS), and versioning to protect data against unauthorized access, data breaches, and accidental deletions.
4. **Flexible Data Storage Class:** Amazon S3 offers multiple storage classes optimized for different use cases, including Standard, Intelligent-Tiering, Standard-IA (Infrequent Access), One Zone-IA, Glacier, and Glacier Deep Archive, allowing users to optimize costs based on their data access patterns and retention requirements.
5. **Lifecycle Policies:** Users can define lifecycle policies to automatically transition objects between different storage classes or delete them after a specified period, helping to optimize storage costs.

Practical Steps for Using Amazon S3:

- **Sign Up for AWS Console:** If you haven't already, sign up for an AWS account at https://aws.amazon.com/ and log in to the AWS Management Console.

- **Create an S3 Bucket:** Navigate to the Amazon S3 console and click on "Create bucket." Specify a globally unique bucket name, choose a region, and configure additional settings such as versioning, logging, and encryption.

- **Upload Objects in Bucket:** Once the bucket is created, you can upload objects (files) to it using the AWS Management Console, AWS CLI (Command Line Interface), or SDKs (Software Development Kits) for various programming languages.

- **Set Permissions and Access Control:** Configure access control settings for your S3 bucket to control who can access the objects stored within it. You can define bucket policies, access control lists (ACLs), and use AWS Identity and Access Management (IAM) roles to manage permissions.

- **Manage Object Life Cycle:** Define lifecycle policies to automatically transition objects to different storage classes or delete them based on predefined criteria such as age or object tags.

- **Monitor and Analyze Usage:** Utilize Amazon S3 metrics and CloudWatch alarms to monitor the usage and performance of your S3 buckets. You can also enable logging to capture detailed access logs for auditing and analysis purposes.

Bucket Policy: This is a bucket policy where you are allowing the Get object action to any item inside the *mybucketgfg* bucket. This policy will make sure that everyone has access to the object.

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

Auto Scaling and Load Balancing:

Autoscaling:

Autoscaling is a feature provided by AWS that automatically adjusts the number of EC2 instances in a fleet based on demand. It ensures that your application always has the right amount of compute capacity to handle varying levels of traffic or workload **without over-provisioning** or **under-provisioning resources**.

When configuring Auto Scaling, you set up policies that define the conditions under which new instances should be launched or existing instances terminated. These policies can be based on metrics such as CPU utilization, network traffic, or custom application metrics. For example, you can define a policy to add more instances when CPU utilization exceeds a certain threshold and remove instances when it falls below another threshold.

By using Auto Scaling, you can achieve the following benefits:

1) **Improved Performance:** Auto Scaling ensures that your application can handle increases in traffic or workload without experiencing performance degradation or downtime.
2) **Cost Optimization:** Auto Scaling helps optimize costs by automatically scaling the number of instances up or down based on demand, allowing you to pay only for the resources you need.
3) **High Availability:** Auto Scaling enhances the availability of your application by automatically replacing unhealthy instances and distributing traffic across healthy instances.

Load Balancing:

Load Balancing is a mechanism used to distribute incoming traffic across multiple instances or servers to ensure optimal performance, availability, and reliability of applications. AWS provides Elastic Load Balancing (ELB) services, which automatically distribute incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses.

There are three types of Elastic Load Balancers provided by AWS:

- **Application Load Balancer(ALB):** ALB operates at the application layer (Layer 7) of the OSI model and can route traffic based on advanced criteria, such as HTTP headers, URL paths, and hostnames. It is ideal for modern applications with multiple microservices or containers.
- **Network Load Balancer(NLB):** NLB operates at the transport layer (Layer 4) and is designed to handle high volumes of traffic with low latency. It is suitable for applications that require extreme performance and scalability, such as gaming or real-time messaging.

- **Classic Load Balancer(CLB):** CLB provides basic load balancing across multiple EC2 instances and operates at both the application and transport layers. It is primarily used for applications that do not require advanced routing features.

By using Load Balancers, you can achieve the following benefits:

High Availability: Load Balancers distribute incoming traffic across multiple instances, ensuring that no single instance becomes overwhelmed and improving the availability of your application.

Scalability: Load Balancers can automatically scale to handle increases in traffic, allowing your application to scale horizontally by adding more instances as needed.

Fault Tolerance: Load Balancers can detect unhealthy instances and redirect traffic to healthy instances, improving the fault tolerance and resilience of your application.

Understanding Cloud Formation and CloudWatch

AWS Cloud Formation:

AWS Service for defining and provisioning infrastructure as a code.
Allows you to model AWS Resources in a template file. In the real world, we don't use the WebUI to create the infrastructure, In the real world we create the template file with all the configurations, variables, and settings, Once this file is ready we can give it to AWS Cloudformation service, It will create the infrastructure for us.

Checkout VPC Cloud Formation Template - https://github.com/sudhanshuvlog/GFG-Devops18/blob/main/AWS/vpc-cloudformation-template.yml

Exploring Simple Notification Service (SNS) and Simple Queue Service (SQS):

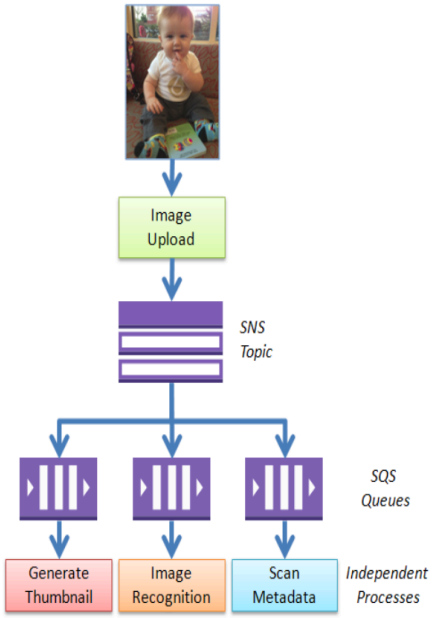
SNS is a fully managed messaging service that enables you to send notifications and messages to distributed systems, mobile devices, and other endpoints. It follows a publish-subscribe (pub/sub) model, where publishers (called topics) send messages to subscribers (called subscribers) who have expressed interest in receiving those messages. SNS supports a variety of protocols, including HTTP/HTTPS, email, SMS, and mobile push notifications (iOS, Android). Messages published to an SNS topic can be delivered to multiple subscribers simultaneously, allowing for broadcast-style communication. It provides features such as message filtering, message attributes, and message delivery retries for enhanced flexibility and reliability. It integrates with other AWS services, enabling you to trigger actions or workflows based on SNS notifications.

Amazon Simple Queue Service:

SQS is a fully managed message queuing service that decouples the components of a distributed application by enabling asynchronous communication between them. It offers two types of queues: standard queues and FIFO (First-In-First-Out) queues, each designed for different use cases.
SQS standard queues provide best-effort ordering and at-least-once message delivery, making them suitable for high-throughput applications with flexible ordering requirements. Messages sent to an SQS queue are stored temporarily until they are

processed by a consumer application, enabling asynchronous communication and workload decoupling.It integrates seamlessly with other AWS services, including AWS Lambda, Amazon EC2, Amazon S3, and AWS CloudWatch, allowing you to build scalable and resilient applications.

Amazon SNS and Amazon SQS are essential messaging services offered by AWS for building scalable, decoupled, and reliable applications in the cloud. SNS facilitates pub/sub-style messaging for broadcasting notifications, while SQS enables asynchronous communication and workload decoupling through message queuing. By leveraging these services, you can design distributed systems that are resilient, scalable, and responsive to changes in workload and demand.



Overview of Relational Database Service (RDS) and Identity and Access Management (IAM)

Identity and Access Management:

IAM is a web service provided by AWS for securely managing access to AWS resources. It allows you to create and manage users, groups, and roles to control who can access your AWS infrastructure and what actions they can perform.

IAM follows the principle of least privilege, enabling you to grant only the necessary permissions required for users and applications to perform their tasks. With IAM, you can create and enforce security policies, define access control rules, and audit user activity to maintain compliance and governance. It provides fine-grained access controls, allowing you to specify permissions at the resource level, down to individual API actions and operations. IAM integrates with AWS services like Amazon S3,

Amazon EC2, AWS Lambda, and Amazon RDS, enabling you to secure and manage access to your cloud resources effectively.

IAM Users:

IAM (Identity and Access Management) users are entities created to represent individual people or applications needing access to AWS resources. Each user has a unique set of credentials and can be assigned specific permissions for accessing and managing AWS services.

IAM Roles:

IAM roles are entities that define a set of permissions but are not associated with a specific user. Instead, roles can be assumed by trusted entities, such as IAM users, applications, or AWS services, allowing temporary access to resources without sharing long-term credentials.

IAM Policies:
IAM policies are JSON documents that define permissions. They specify what actions are allowed or denied on which resources. Policies can be attached to users, groups, or roles to control their access. Policies provide fine-grained control over AWS resources.

IAM Groups:

IAM groups are collections of IAM users. You can attach policies to groups, and all users within the group inherit the permissions specified in the group's policies. This simplifies the management of permissions, especially when dealing with multiple users requiring similar access levels.

Sample IAM Policy:

- Effect: "Allow" indicates that the actions specified in the statement are permitted.
- Action: Lists the S3 actions that are allowed.
    - s3:ListBucket: Permission to list the objects in the specified S3 bucket.
    - s3:GetObject: Permission to retrieve objects from the bucket.
    - s3:PutObject: Permission to upload objects to the bucket.

- Resource: Specify the S3 resources these actions apply to.
    - arn:aws:s3:::your-s3-bucket-name: The bucket itself.
    - arn:aws:s3:::your-s3-bucket-name/*: All objects within the bucket.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket",
            "s3:GetObject",
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::your-s3-bucket-name",
            "arn:aws:s3:::your-s3-bucket-name/*"
         ]
      },
   ]
}
```

Amazon Relational Database Service:

Amazon RDS is a fully managed relational database service provided by AWS, offering scalable, high-performance, and cost-effective databases in the cloud. It supports several popular relational database engines, including MySQL, PostgreSQL, Oracle, SQL Server, and Amazon Aurora.

RDS manages routine database tasks such as provisioning, patching, backups, and monitoring, allowing you to focus on building applications rather than managing infrastructure. With RDS, you can easily launch, scale, and manage database instances in a matter of minutes, without the need for manual intervention or expertise in database administration. It provides features such as automated

backups, point-in-time recovery, read replicas, and Multi-AZ (Availability Zone) deployments for enhanced reliability, durability, and fault tolerance. It integrates seamlessly with other AWS services, including AWS Lambda, Amazon S3, Amazon Redshift, and AWS CloudWatch, enabling you to build scalable and resilient applications with ease.

Serverless Architecture

Serverless architecture, often associated with AWS Lambda, is a cloud computing model where cloud providers dynamically manage the allocation of machine resources.

Overview of Serverless Architecture:

- Serverless architecture abstracts server management, allowing developers to focus solely on writing code for their applications.
- In a serverless environment, developers deploy functions or applications without worrying about provisioning or managing servers.
- Serverless computing scales automatically based on demand, ensuring optimal performance and resource utilization.
- Serverless applications are event-driven, meaning they respond to triggers or events such as HTTP requests, database changes, file uploads, etc.
- AWS Lambda is a leading serverless computing service offered by AWS, allowing developers to run code in response to events without provisioning or managing servers.

AWS Lambda:

AWS Lambda is a serverless compute service that runs code in response to events and automatically scales to handle varying workloads. With Lambda, you can upload your code (written in languages such as Node.js, Python, Java, or C#) and AWS handles the rest, including provisioning, scaling, and monitoring. Lambda functions are stateless and short-lived, typically executing within milliseconds and billed per execution and the duration of code execution. Functions can be triggered by various AWS services such as API Gateway, S3, DynamoDB, SNS, and CloudWatch Events. AWS Lambda integrates seamlessly with other AWS services, allowing you to build serverless applications that react to events in real time. Lambda functions can be used for various use cases, including data processing, real-time stream processing, backend services, automation, and more.

Lambda Function -

The below lambda function will be triggered when any object is pushed to the S3 bucket, This lambda function will get the object via AWS Python SDK, then it will pass the object to the AWS Rekognisation service which will detect the image and will reply to us with the image labels. All the function logs can be found in the Cloudwatch Service.

https://github.com/sudhanshuvlog/GFG-Devops18/blob/main/AWS/Lambda-Function-aws-s3-rekognition.py

Amazon API Gateway

API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. API Gateway acts as a "front door" for applications to access data, business logic, or functionality from your backend services, such as applications running on Amazon EC2, AWS Lambda, or any web application.

Key Features of Amazon API Gateway

- API Creation and Management: Simplifies the process of creating and deploying APIs. You can define RESTful APIs, WebSocket APIs, and HTTP APIs.

- Security and Authorization: Supports various authentication and authorization mechanisms, including AWS IAM roles, Lambda authorizers, and Amazon Cognito.

- Monitoring and Logging: Integrates with Amazon CloudWatch to provide detailed monitoring and logging of API calls.

- Scaling and Performance: Automatically scales to handle the number of requests your API receives.

Connecting API Gateway with AWS Lambda

Connecting API Gateway with AWS Lambda allows you to create powerful and scalable serverless applications. Here's how you can do it:

Create an API in API Gateway:

- Go to the API Gateway service in the AWS Management Console.
- Click "Create API" and choose the API type (REST API, HTTP API, or WebSocket API).

- For a REST API, choose "Build" under "REST API", then select "New API" and give it a name and description.
- Create a Resource and Method:

- Under your new API, click "Resources" in the left-hand menu.
- Click "Create Resource" and provide a resource name and path.
- With the new resource selected, click "Create Method" and choose the HTTP method (e.g., GET, POST).
- Select "Lambda Function" as the integration type.
  In the integration settings, choose the region where your Lambda function is located. Enter the name of your Lambda function and click "Save". API Gateway will prompt you to grant it permission to invoke your Lambda function; click "OK".
- Deploy the API: Click "Actions" and select "Deploy API".
  Create a new stage (e.g., "prod") and provide a stage name.
- Invoke the API: You can use this URL to make HTTP requests to your API, which will trigger your Lambda function.

Benefits of having serverless architecture:

Reduced Operational Overhead: Eliminates the need for server provisioning, maintenance, and scaling, allowing developers to focus on building applications.
Cost-effective: With serverless computing, you only pay for the compute time consumed by your functions, leading to cost savings compared to traditional server-based architectures.
Scalability and elasticity: Serverless platforms automatically scale up or down based on workload demand, ensuring optimal performance and resource utilization.
Increased Agility: Serverless architecture enables rapid development, deployment, and iteration of applications, facilitating faster time-to-market and innovation.

Serverless architecture, exemplified by AWS Lambda, offers a scalable, cost-effective, and agile approach to building and deploying applications in the cloud. By Using serverless computing, developers can focus on writing code and delivering value to customers without the complexities of managing infrastructure.

Utilizing CloudWatch for Monitoring and Setting Billing Alarms

CloudWatch for Monitoring:

Amazon CloudWatch is a monitoring and observability service provided by AWS. It collects and tracks metrics, logs, and events from various AWS services and resources in real-time.

CloudWatch provides insights into the operational health, performance, and resource utilization of your AWS environment. We can monitor metrics such as CPU utilization, memory usage, disk I/O, network traffic, and more. CloudWatch allows you to create custom dashboards to visualize key metrics and monitor the health of your infrastructure. It also supports alarms and notifications to alert you when metrics exceed predefined thresholds or when specific events occur. CloudWatch Logs centralizes log data from AWS services and applications, making it easier to troubleshoot issues and analyze system behavior.

Setting Billing Alarms:

CloudWatch can also be used to monitor AWS billing and set up billing alarms to manage costs effectively. We can track your AWS usage and spending using CloudWatch Billing Metrics, which provide insights into your monthly bill. By setting up billing alarms, you can receive notifications when your AWS costs exceed specified thresholds. This helps to proactively monitor and control AWS spending, preventing unexpected charges and budget overruns.
To set up billing alarms, navigate to the CloudWatch console, select Billing, and then click on Create Alarm. Specify the billing metric to monitor (e.g., EstimatedCharges), set the threshold and conditions for the alarm, and configure notification actions (e.g., SNS topic, email). Once configured, CloudWatch will monitor your AWS spending and notify you when the specified thresholds are exceeded, allowing you to take appropriate actions to optimize costs.

Amazon CloudFront

Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency and high transfer speeds. CloudFront is integrated with AWS, including physical locations directly connected to the AWS global infrastructure, as well as other AWS services.

Key Features of CloudFront

- Global Network of Edge Locations: CloudFront has a vast network of data centers (edge locations) around the world to cache content closer to end-users.
- Low Latency and High Transfer Speeds: By serving content from edge locations closer to the user, CloudFront reduces latency and improves download speeds.

- Security: CloudFront integrates with AWS Shield for DDoS protection, AWS Web Application Firewall (WAF) for web application security, and AWS Certificate Manager (ACM) for SSL/TLS certificates.
- Content Caching: Automatically caches content at edge locations, reducing the load on the origin server and improving the performance for users.
- Customizable Content Delivery: Supports dynamic and static content delivery, with capabilities for customizing the caching behavior and controlling access to content.

Create a CloudFront Distribution:

- In the AWS Management Console, navigate to the CloudFront service. Click "Create Distribution".
- Choose between a "Web" distribution for HTTP/HTTPS content or an "RTMP" distribution for media streaming (RTMP is deprecated and not commonly used anymore).
- Configure Origin Settings: Specify the origin domain name, which is the source of the content that CloudFront will deliver. This can be an S3 bucket, an HTTP server, an Elastic Load Balancer, or another web server.
- Set Up Cache Behavior: Define how CloudFront will cache and deliver your content. You can specify different behaviors for different paths or file types.
  Configure settings like allowed HTTP methods, cache TTL (Time-to-Live) settings, and query string forwarding.
- Configure Distribution Settings: Set up distribution-specific settings such as SSL certificates, logging, and price class (which controls the number of edge locations your content is served from).
  Enable additional features like Geo Restriction, which allows you to restrict access to your content based on the geographic location of your viewers.
- Deploy the Distribution: Review the settings and click "Create Distribution".
  It may take some time for the distribution to be deployed and propagated across all edge locations.
- Use the CloudFront Domain Name: Once the distribution is deployed, CloudFront will provide a domain name (e.g., d123456abcdef8.cloudfront.net). Use this domain name to access your content. You can also configure your own custom domain names using Route 53 or other DNS services.

Example Use Cases

- Static Content Delivery: Serve static assets like HTML, CSS, JavaScript, and images from edge locations to improve load times for websites and applications.
- Video Streaming: Deliver high-quality video content to a global audience with minimal buffering and latency.

- API Acceleration: Distribute APIs globally, reducing the latency for API calls and improving the performance of applications.
  Performance: Improves the performance of your content delivery by caching content at edge locations closer to your users.
- Scalability: Automatically scales to handle varying traffic patterns without the need for manual intervention.
- Security: Enhances security with DDoS protection, data encryption, and custom security rules.
- Cost Efficiency: Reduces the load on your origin servers, potentially lowering your infrastructure costs.

Amazon Elastic Container Service (ECS)

Amazon Elastic Container Service (ECS) is a fully managed container orchestration service that allows you to run and manage Docker containers on a cluster of Amazon EC2 instances. ECS simplifies the deployment and management of containerized applications.

Key Concepts of ECS

- ECS Cluster: A logical grouping of EC2 instances or AWS Fargate tasks where you run your containerized applications.

- Task Definition: A blueprint that describes how a Docker container should run. It includes details like container images, CPU and memory requirements, networking, and IAM roles.

- Service: Defines how many instances of a task definition to run and maintains the desired count, replacing tasks if they fail.

- Task: An instance of a task definition that is running on an ECS cluster.

Task Definition

A task definition is a text file in JSON format that describes one or more containers that form your application. Here are some key components of a task definition:

- Container Definitions: Details about the containers that make up the task, including Docker image, CPU, memory, ports, and environment variables.
- Volumes: Data volumes used by the containers.
- Network Mode: Specifies the Docker networking mode to use for the containers in the task (e.g., bridge, host, awsvpc).
- IAM Roles: Roles that the task can assume to access AWS services.

Creating an ECS Cluster

- Navigate to the ECS service in the AWS Management Console.
  Click "Create Cluster".
- Choose between networking-only cluster (for Fargate tasks), EC2 Linux + Networking, or EC2 Windows + Networking, depending on your use case.
- Configure the cluster settings, including the VPC, subnets, and instance type if using EC2 instances.
- Click "Create".

AWS Fargate

AWS Fargate is a serverless compute engine for containers that works with both Amazon ECS and Amazon EKS (Elastic Kubernetes Service). With Fargate, you don't need to provision or manage servers. Fargate handles the infrastructure, allowing you to focus on defining and running your containers.

Benefits of AWS Fargate

- No Server Management: No need to provision, configure, or scale clusters of virtual machines to run containers.
- Right-Sizing: Specify and pay for the exact amount of vCPU and memory resources required for your containerized applications.
- Isolation: Improved security with application isolation by design.

ECS Service

An Amazon ECS Service is a construct that allows you to run and maintain a specified number of instances of a task definition simultaneously in an Amazon ECS cluster. It ensures that the desired number of tasks are running and reschedules tasks if they fail or stop.

Key Features of ECS Service

- Desired Task Count: Ensures that a specified number of tasks are running at all times. If a task fails or stops, the service scheduler launches another instance to replace it.
- Load Balancing: Integrates with Elastic Load Balancing (ELB) to distribute traffic evenly across the tasks in your service.
- Service Discovery: Enables service discovery to connect services together within a VPC using AWS Cloud Map or DNS names.

- Auto Scaling: Automatically adjusts the number of tasks in your service based on specified criteria, such as CPU or memory usage.
- Task Placement Strategies: Controls how tasks are placed on instances in your cluster, based on binpack, random, or spread strategies.
- Deployment Types: Supports rolling updates, blue/green deployments (using AWS CodeDeploy), and more to ensure smooth updates with minimal downtime.