AWS Solution Architect Associate

Version : C03

Domain 2

Task 1

Design scalable and loosely coupled architectures

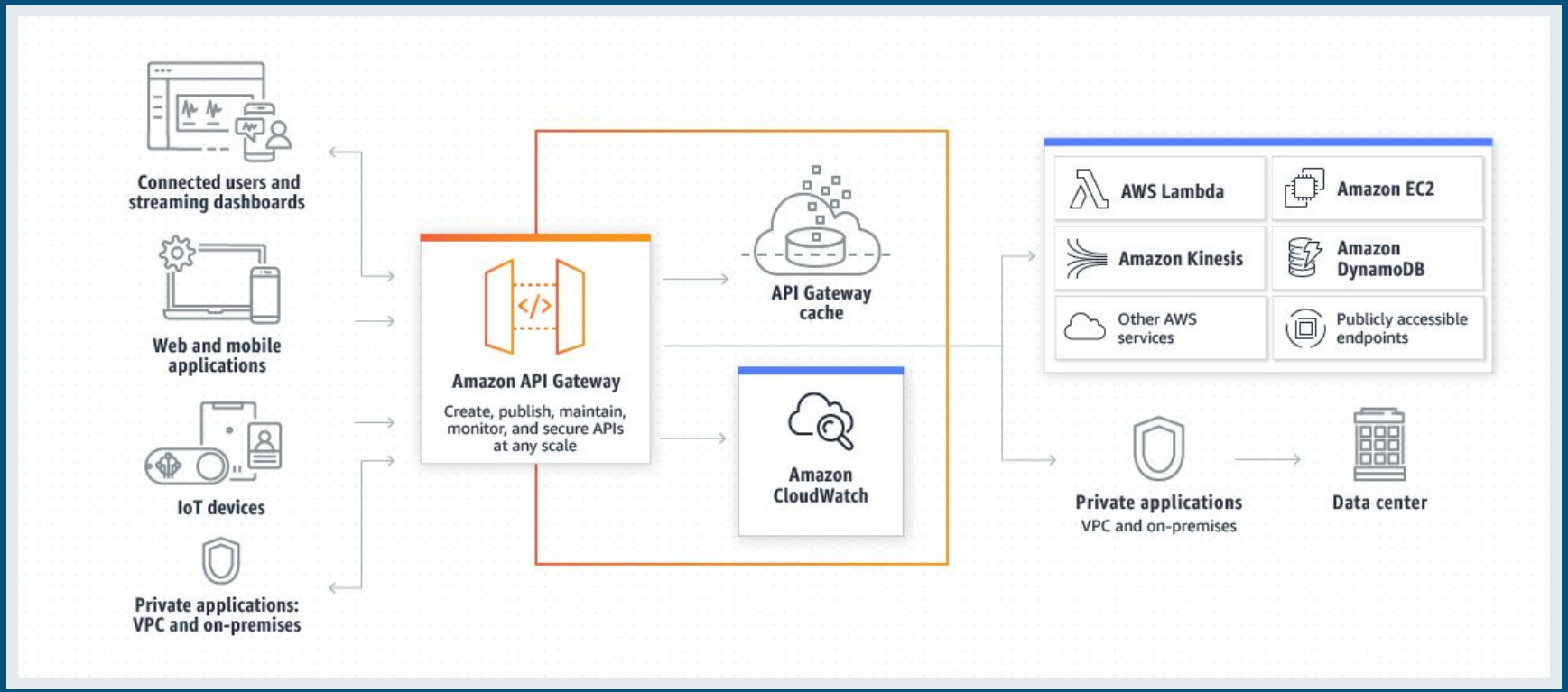
REST API

- AWS REST API is a scalable, secure and easy-to-use interface for accessing AWS resources.
- It allows applications to access and control the AWS services using HTTP requests.
- Provides a set of standard endpoints and methods for accessing various AWS services
- Tools like cURL or SDKs can be used to make requests to the API
- It supports features like authentication, authorization and pagination to ensure secure and efficient access to AWS resources.
- REST APIs and HTTP APIs are both RESTful API products.
  - REST APIs support more features than HTTP APIs
  - HTTP APIs are offered at a lower price.
  - REST APIs features - API keys, per-client throttling, request validation, AWS WAF integration or private API endpoints

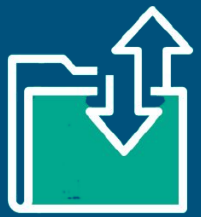
Amazon API Gateway

- Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring and securing REST, HTTP, and WebSocket APIs at any scale.
- APIs can be created that access AWS or other web services, as well as data stored in the AWS Cloud.
- API Gateway creates RESTful APIs that :
  - Are HTTP-based
  - Enable stateless client-server communication
  - Implement standard HTTP methods such as GET, POST, PUT, PATCH and DELETE
- API Gateway creates WebSocket APIs that :
  - Adhere to the WebSocket protocol which enables stateful, full-duplex communication between client and server.
  - Route incoming messages based on message content.

Architecture of Amazon API Gateway

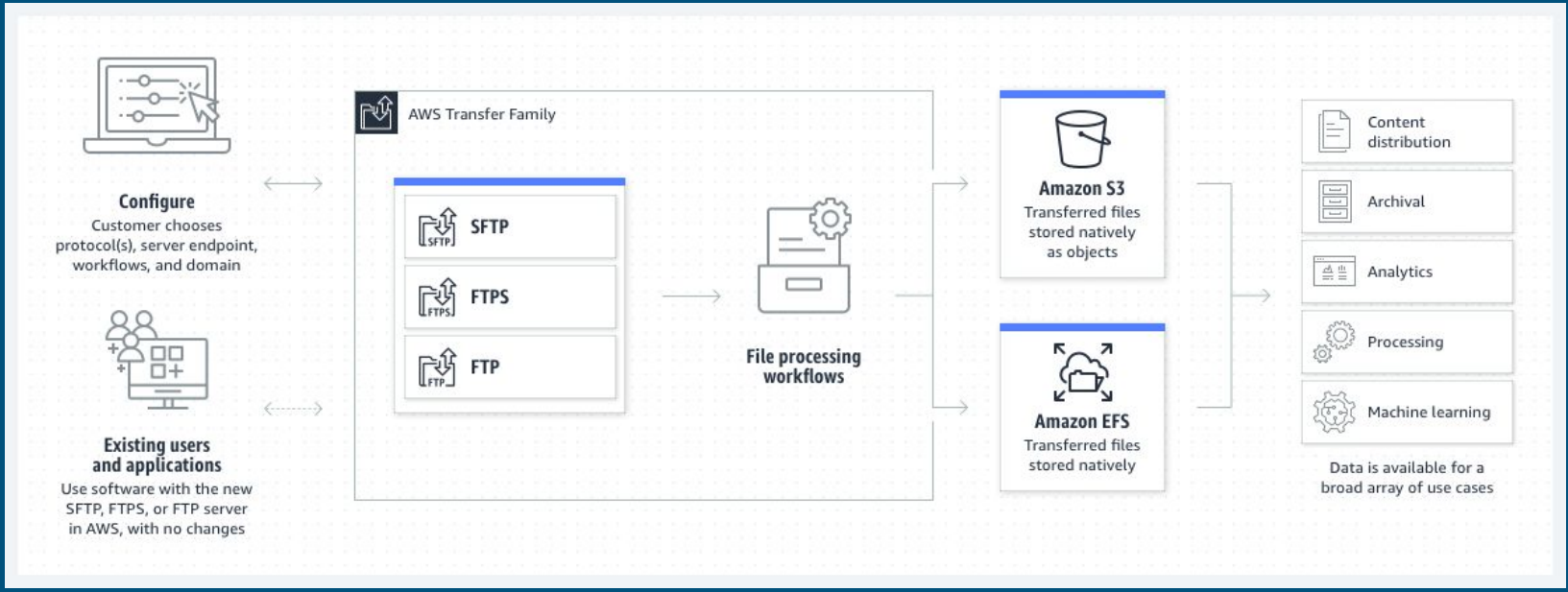


AWS Transfer Family



- AWS Transfer Family is a secure transfer service that enables you to transfer files into and out of AWS storage services.
- It is a part of the AWS Cloud platform.
- Support over following protocols
  - Secure File Transfer Protocol (SFTP-v3)
  - File Transfer Protocol Secure (FTPS)
  - File Transfer Protocol (FTP)
  - Applicability Statement (AS2)
- File transfer workflows can seamlessly migrate, automate and monitor the file transfer workflows
- Supports transferring data from or to the following Amazon web services
  - Amazon S3
  - Amazon Elastic File System
- Use Cases
  - Modernize your managed file transfers
  - Gain insights by growing your data lake
  - Improve collaboration across your trading partner network
  - Expand your content distribution business

AWS Transfer Family



Amazon Simple Queue Service

- Amazon SQS aks Amazon Simple Queue Service
- Offers a secure, durable and available hosted queue that integrate and decouple distributed software systems and components.
- It offers common constructs such as dead-letter queues and cost allocation tags.
- Benefits of using SQS
  - Security
  - Durability
  - Availability
  - Scalability
  - Reliability

Amazon Simple Queue Service

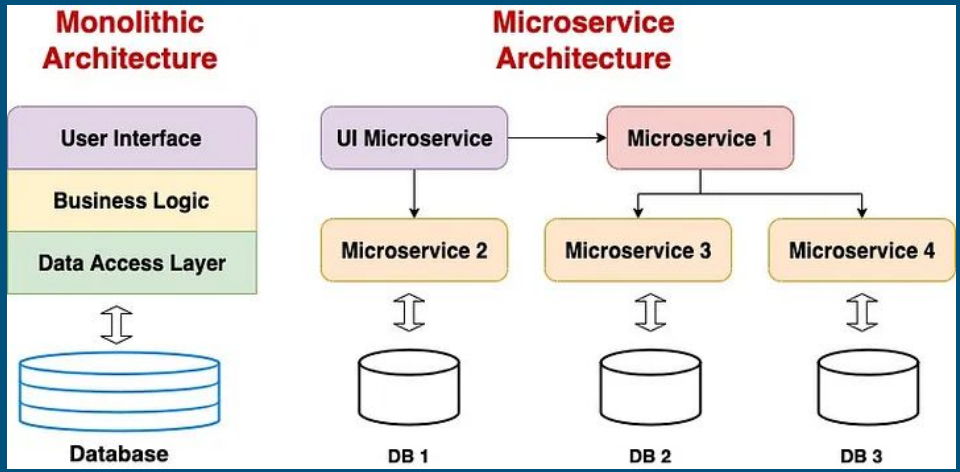
- ➔ Queues for Message Storage
- ➔ Producer-Consumer Model
- ➔ Message Lifecycle
- ➔ Dead-Letter Queues (DLQs)
- ➔ Scaling and Fault Tolerance

Caching Strategies

- Refers to the methods for caching data in Amazon's cloud-based infrastructure to maximize its performance and availability.
- AWS provides several caching options
  - Amazon ElastiCache
  - Amazon S3 Caching Acceleration
  - AWS Lambda Caching
  - Amazon File Cache
  - Amazon DynamoDB Accelerator (DAX)
  - Amazon Greengrass
- Each strategy has unique performance characteristics and is suited for different workloads.
- The choice of caching strategy depends on the specific requirements of the application including data access frequency, access patterns and data size.
- Caching strategies can help to reduce latency, improve throughput and reduce costs associated with storage and data retrieval.
- Different caching strategies
  - **Lazy loading**
  - **Write-Through**

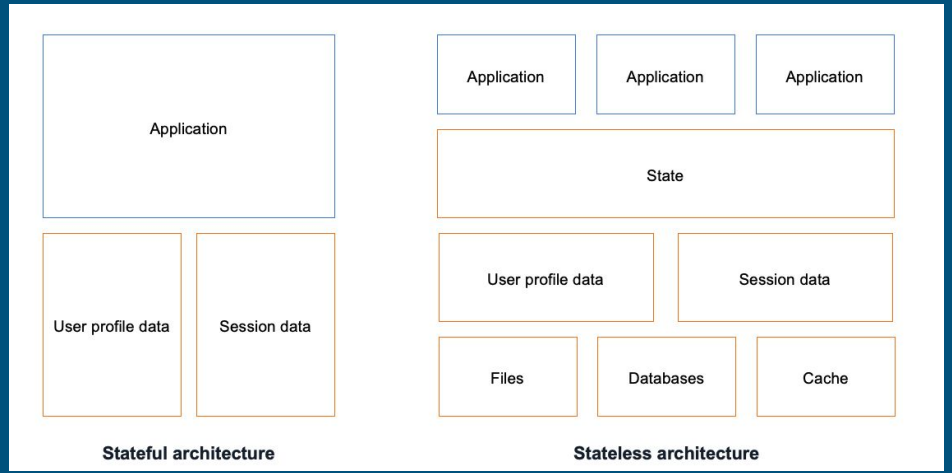
Design Principles for Microservices

- Approach of breaking down a large monolithic application into smaller, independent services that are more scalable, flexible and maintainable
- Key principles
  - **Loose coupling**
  - **High cohesion**
  - **Service contract design**
  - **Failure tolerance**
  - **Independent deployment**



Stateless/Stateful Workloads

- **Stateful applications**
  - Simple to deploy
  - Saves client session data on the server, allowing for faster processing and improved performance.
  - Stateful applications excel in predictable workloads and offer consistent user experiences.
- **Stateless architectures**
  - Align with the demands of dynamic workload and changing business requirements.
  - Stateless application design can increase flexibility with horizontal scaling and dynamic deployment.
  - This flexibility helps applications handle sudden spikes in traffic, maintain resilience to failures and optimize cost.



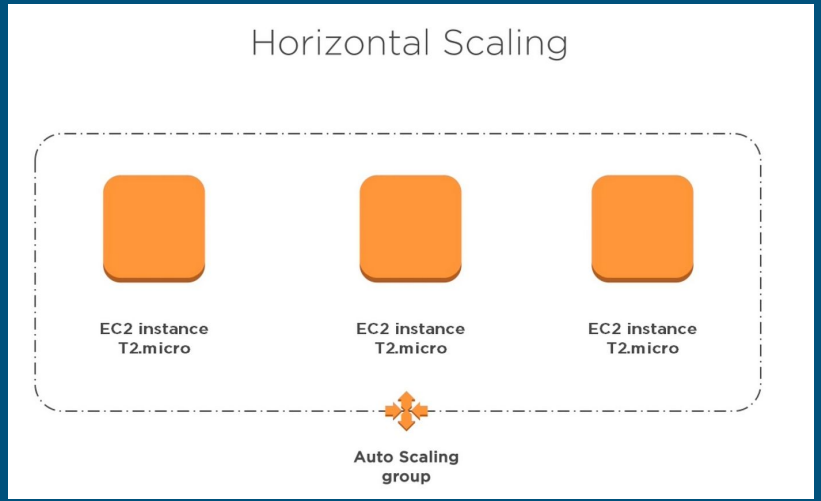
Event Driven Architectures

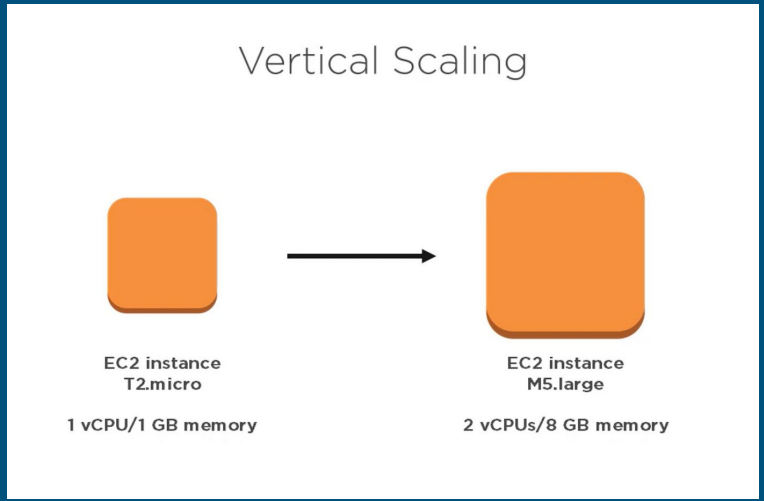
- An event-driven architecture uses events to trigger and communicate between decoupled services and is common in modern applications built with microservices.
- An event is a change in state or an update, like an item being placed in a shopping cart on an e-commerce website.
- Events can either carry the state or events can be identifiers
- Key components :
  - **Event producers**
  - **Event routers**
  - **Event consumers**
- A producer publishes an event to the router, which filters and pushes the events to consumers.
- Producer services and consumer services are decoupled, which allows them to be scaled, updated and deployed independently.

Horizontal/Vertical Scaling

- AWS horizontal scaling and vertical scaling are two different approaches to scaling applications in the cloud.
- **Horizontal scaling**
  - Involves adding more instances of a running application to handle increased traffic or workload.
  - This can be done using Amazon Elastic Load Balancing (ELB) and Amazon Auto Scaling.
- **Vertical scaling**
  - Involves increasing the resources available to a single instance, such as CPU, memory, or storage.
  - This can be done using Amazon EC2 instances with larger instance types.
- Both approaches have advantages and disadvantages and the choice of which to use depends on the specific needs of the application.

Horizontal/Vertical Scaling





Content Delivery Network

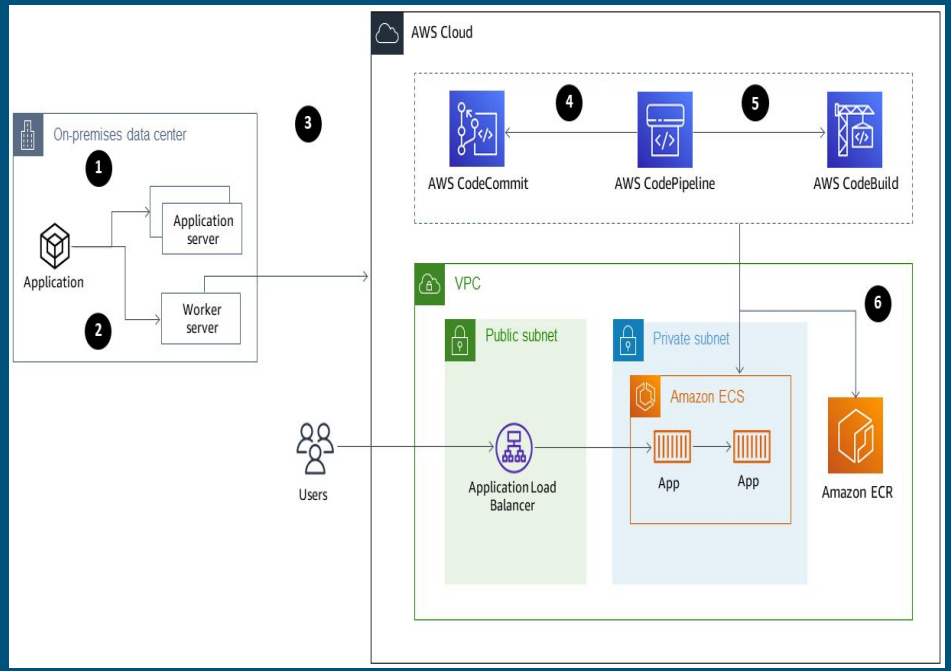
- CDN is a network of interconnected servers that speeds up webpage loading for data-heavy applications.
- It can stand for content delivery network or content distribution network.
- Use case : When a user visits a website, data from that website's server has to travel across the internet to reach the user's computer. If the user is located far from that server, it will take a long time to load a large file, such as a video or website image. Instead, the website content is stored on CDN servers geographically closer to the users and reaches their computers much faster.
- The primary purpose is to reduce latency/delay in communication created by a network's design
- Benefits of CDN
  - Reduce Page load time
  - Reduce Bandwidth costs
  - Increase Content availability
  - Improve Website security
- CDN can deliver 2 types of content
  - Static Content
  - Dynamic Content

Content Delivery Network (Edge Accelerators)

- AWS Edge Accelerator is a content delivery network (CDN) service that helps deliver content faster to end-users by caching content closer to them.
- It is designed to improve the performance of web applications and APIs by reducing latency and improving throughput.
- AWS Edge Accelerator can be used to cache static content, such as images and videos, as well as dynamic content, such as web pages and APIs.
- Supports a variety of protocols - HTTP, HTTPS, and WebSocket and can be integrated with other AWS services such as Amazon CloudFront and Amazon API Gateway.

Migrating Applications to containers

- AWS offers several services for migrating applications to containers
  - **AWS Elastic Container Service**
  - **AWS Elastic Kubernetes Service**
  - **AWS Fargate**
- These services provide a scalable and flexible infrastructure for deploying and managing containerized applications.
- It offers tools and resources for containerizing applications
  - **AWS Container Registry**
  - **AWS CodeBuild**
- AWS provides guidance and best practices for migrating applications to containers, including considerations for security, performance and scalability.



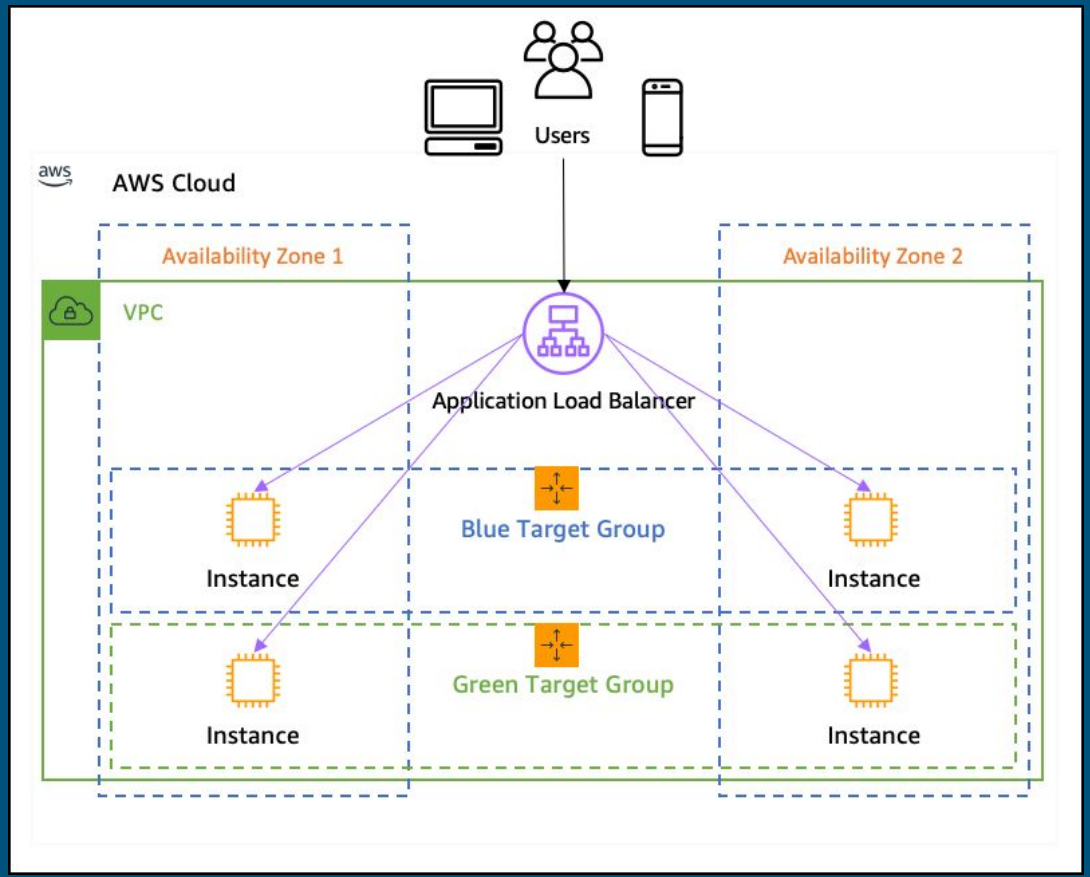
Migrating Applications to containers

- Steps for migrating applications to containers using AWS :
  - **Assess your application** - Determine which containers are appropriate for your application and how they should be configured.
  - **Modify your application** - Modify your application code to run in containers.
  - **Choose a container service** - Select the container service that best meets your needs.
  - **Create container images** - Create container images for your application using a tool such as Docker.
  - **Deploy to the container service** - Deploy your containers

Load Balancing

- Refers to the process of distributing network traffic across multiple servers or instances to ensure that no single server becomes overloaded.
- This helps to improve the performance, scalability, and reliability of cloud-based applications and services.
- AWS Load Balancing
  - Service offered by (AWS) that helps distribute incoming network traffic across multiple instances, containers or IP addresses
  - Improves application performance and availability, simplify scaling and configuration and enhance security.
  - Types of load balancing offered by AWS :
    - **Application Load Balancers**
    - **Network Load Balancers**
    - **Load Balancer Target Groups**
    - **EC2 Auto Scaling Groups**

Load Balancing



Multi-Tier Architecture

- AWS multi-tier architecture is a **type of system design** that involves breaking down an application into smaller, more manageable components, which are distributed across multiple layers.
- This provides better scalability, performance and cost efficiency.
- The layers includes
  - **Presentation**
  - **Application**
  - **Business logic**
  - **Data storage and retrieval**
  - **Data processing**
- This reduces coupling and makes the system easier to manage and maintain.
- Different AWS tools/services to implement a multi-tier architecture -
  - **Elastic Load Balancing**
  - **AWS Auto Scaling**
  - **Amazon Elastic Container Service.**

Serverless technologies

- In the serverless model, a cloud provider manages the provisioning, scaling, and maintaining the underlying infrastructure.
- The cloud provider handles tasks like operating system management, security patches, file system and capacity management, load balancing, monitoring and logging.
- Serverless use cases :
  - Business process automation
  - Real-time data analytics
  - Batch processing
  - Stateless application development
- Types of serverless architecture
  - **Functions as a service** - Can be used to write and deploy the function code directly to the cloud infrastructure
  - **Backend as a service** - Provides access to backend functions using an API

Serverless technologies (AWS Services)

- **Compute**
  - AWS Lambda
  - AWS Fargate
- **Application Integration**
  - Amazon EventBridge
  - AWS Steps Functions
  - Amazon SQS
  - Amazon SNS
  - Amazon API Gateway
  - AWS AppSync
- **Data Store**
  - Amazon S3
  - Amazon EFS
  - Amazon DynamoDB
  - Amazon RDS Proxy
  - Amazon Aurora Serverless
  - Amazon Redshift Serverless
  - Amazon Neptune Serverless
  - Amazon OpenSearch Serverless
  - Amazon ElasticCache Serverless

Storage types

- Three main cloud storage types and the corresponding AWS services :
  - **Object Storage**
    - Amazon S3
    - Amazon Glacier
    - Amazon S3 Glacier Deep Archive
    - Amazon S3 Intelligent-Tiering
  - **File Storage**
    - Amazon EFS
    - Amazon FSx for Windows File Server
    - Amazon FSx for Lustre
    - AWS Storage Gateway
  - **Block Storage**
    - Amazon EBS
    - Amazon EC2 instance store
    - Amazon EFS (NFSv4)

Container Orchestration

- Involves managing , automating the deployment, scaling and operation of containerized applications.
- AWS service for container orchestration
  - **Amazon Elastic Container Service**
  - **Amazon Elastic Kubernetes Service**
  - **AWS Fargate**
  - **AWS App Runner**
- **Amazon Elastic Container Service -**
  - Fully managed container orchestration service
  - Supports Docker containers.
  - Allows to run\manage containerized applications
  - Task scheduling, service discovery, auto scaling and integration with other AWS services.
- **Amazon Elastic Kubernetes Service**
  - Fully managed Kubernetes service
  - Fully managed control plane and integration with AWS services
  - Simplifies the deployment, management, and scaling of Kubernetes applications
  - Can run containerized applications using standard Kubernetes APIs and tooling.

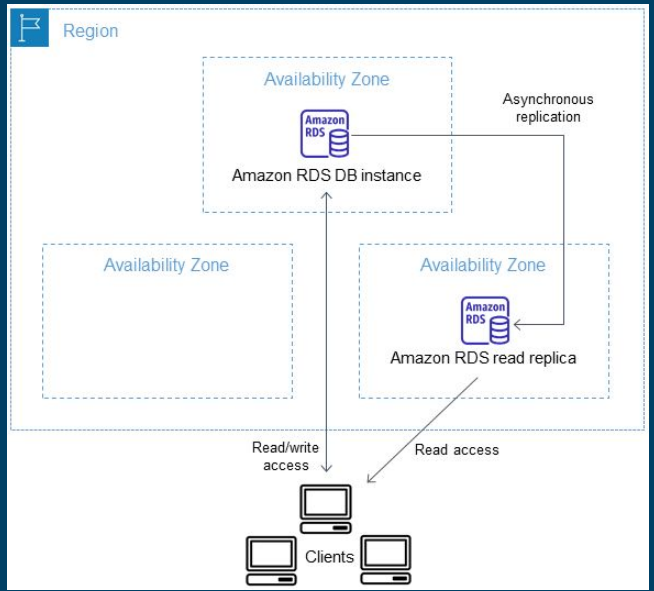
Container Orchestration

- **AWS Fargate**
  - Serverless compute engine for containers
  - Container requirements such as CPU and memory can be specified and AWS takes care of provisioning and scaling the compute resources automatically.
  - Integrates seamlessly with Amazon ECS and Amazon EKS
  - No need to manage servers or clusters
- **AWS App Runner**
  - Fully managed service that deploy and run containerized web applications at scale.
  - Automatically builds and deploys the container images from source code or Dockerfiles, manages the infrastructure, and scales the application based on traffic.
  - Suitable for running web applications, APIs, and microservices with minimal management overhead.

Container Orchestration

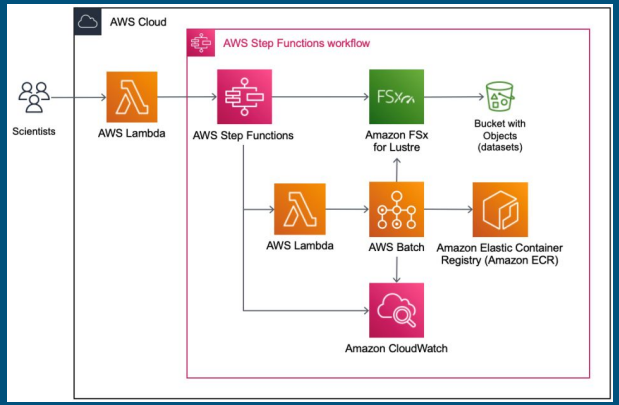
Read Replicas

- A read replica is a read-only copy of a DB instance
- Load on the primary DB instance can be reduced by routing queries from your applications to the read replica
- Can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads



Workflow Orchestration

- ➔ Technology that allows you to create and manage workflows to automate tasks and processes.
- ➔ Workflow orchestration is a key component of AWS step functions
- ➔ **AWS Step Functions**
  - ◆ A serverless orchestration service
  - ◆ Combines AWS Lambda functions and other AWS services to build to scalable, distributed applications using state machines
  - ◆ Based on state machine and tasks
  - ◆ State machine is a workflow
  - ◆ Task is a state in workflow that represents a single unit of work that another AWS service performs
  - ◆ Scales horizontally and provides fault-tolerant workflows
  - ◆ Workflow types
    - **Standard Workflows**
      - Exactly once workflow transformation
    - **Express Workflows**
      - At-Least once workflow transformation



AWS Solution Architect Associate

Version : C03

Domain 2

Task 1

The END