AWS Solution Architect Associate

Version : C03

Domain 3

Task 2

Design high-performing and elastic compute solutions

AWS Compute Services

- **Amazon Elastic Compute Cloud (EC2)** : Provides resizable compute capacity in the cloud, allowing you to quickly scale up or down as your computing requirements change
- **Amazon Elastic Container Service (ECS)** : Fully managed container orchestration service for Docker containers, allowing you to easily run, stop and manage containers on a cluster
- **Amazon Elastic Kubernetes Service (EKS)** : Managed Kubernetes service that simplifies the process of building, securing, operating and scaling Kubernetes clusters
- **AWS Lambda** : Serverless computing service that lets you run code without provisioning or managing servers. Charges are calculated on the compute time consumed
- **AWS Wavelength** : Brings AWS services to the edge of the 5G network, allowing you to build applications with ultra-low latency
- **AWS Batch** : Enables to run batch computing workloads on AWS, allowing to scale and manage batch computing jobs efficiently
- **Amazon Lightsail** : Simplified virtual private server (VPS) service that enables to launch and manage virtual private servers easily
- **AWS Elastic Beanstalk** : Automatically handles the deployment, provisioning, load balancing and scaling of web applications and services
- **AWS Fargate** : Serverless compute engine for containers that allows you to run containers without having to manage the underlying infrastructure
- **AWS Outposts** : Extends AWS infrastructure, services and tools to customer premises, providing a fully managed service that operates just like in the cloud

AWS Compute Services - Usecases

AWS Compute Service	Use Cases
EC2	Hosting web applications ,Running enterprise applications,Batch processing,High-performance computing, Machine learning (ML) and artificial intelligence (AI) tasks,Content delivery,Backup and disaster recovery
ECS / EKS	Containerized microservices architecture,CI/CD pipelines,Auto-scaling containerized applications,Orchestration of Docker containers
Lambda	Event-driven computing,Real-time file processing,Backend for mobile applications,Real-time stream processing, Automation of tasks using serverless architecture
Batch	Processing large-scale data sets,ETL (Extract, Transform, Load) jobs,Log processing and analysis
Lightsail	Hosting simple websites,Running blogs and content management systems (CMS),Development and testing environments,Small-scale applications
Elastic Beanstalk	Rapid deployment and management of web applications,Automatic scaling of web applications,Load balancing and capacity provisioning,Monitoring and logging
Outposts	Running workloads in on-premises environments with the benefits of AWS infrastructure,Low-latency applications that require local data processing
Serverless Application Repository (SAR)	Sharing and discovering serverless applications and components,Reusing pre-built serverless applications and components to accelerate development

Selecting the appropriate compute option

- **Evaluate Workload Characteristics**
  - Nature of the workload - CPU-intensive, Memory-intensive or I/O-intensive ?
  - Scale of the workload - Small, Predictable workload or does it require rapid scaling and elasticity ?
  - Stateless or stateful workload - Require persistent storage or Ephemeral ?
- **Consider Compute Services**
  - **Amazon EC2** - Full control over the underlying infrastructure and flexibility to customize compute environment.Ideal for legacy systems and workloads with specific software requirements.
  - **AWS Lambda** - Event-driven, short-lived, stateless functions Suitable for microservices, real-time data processing or tasks triggered by AWS events.
  - **ECS/EKS** - For containerized workloads ,Suitable for microservices architectures, CI/CD pipelines, and applications with complex dependencies
  - **AWS Batch** - For batch computing workloads, such as ETL jobs, data processing and scientific simulations.Provides automated scaling and resource provisioning for batch jobs
  - **Amazon Lightsail** - For simple, low-traffic applications that require a fixed amount of compute resources.Suitable for small websites and development environments
- **Evaluate Cost and Pricing Models**
  - Consider the pricing structure of each compute option
  - Estimate the total cost of ownership (TCO) including compute, storage, networking and any additional services required.

Selecting the appropriate compute option

- **Assess Management Overhead**
  - Evaluate the level of management overhead required for each compute option.
  - Consider factors such as provisioning, monitoring, scaling, patching, and security.
- **Security and Compliance Requirements**
  - Security features, compliance certifications and data residency requirements
  - Organization's security and compliance standards.
- **Plan for Scalability and High Availability**
  - Workload requires scalability and high availability ?
  - Auto-scaling, load balancing and fault tolerance is needed ?
- **Architectural Considerations**
  - Overall architecture and design principles
  - Integration with other AWS services, data storage solutions, networking requirements and DevOps practices

Distributed Computing

- Method of making multiple computers work together to solve a common problem.
- It makes a computer network appear as a powerful single computer that provides large-scale resources to deal with complex challenges
- Advantages of Distributed Computing
  - **Scalability** - Can grow with the workload requirements
  - **Availability** - It doesn't crash with a single computer failure as the design includes fault tolerance
  - **Consistency** - Same information is duplicated between different systems and the system automatically manages the data consistency
  - **Transparency** - Provides logical separation between the user and the physical devices
  - **Efficiency** - Offers faster performance with optimum resource usage . It can handle volume spikes
- Usecases
  - Healthcare and Life sciences
  - Engineering Research
  - Financial Services
  - Energy and Environment

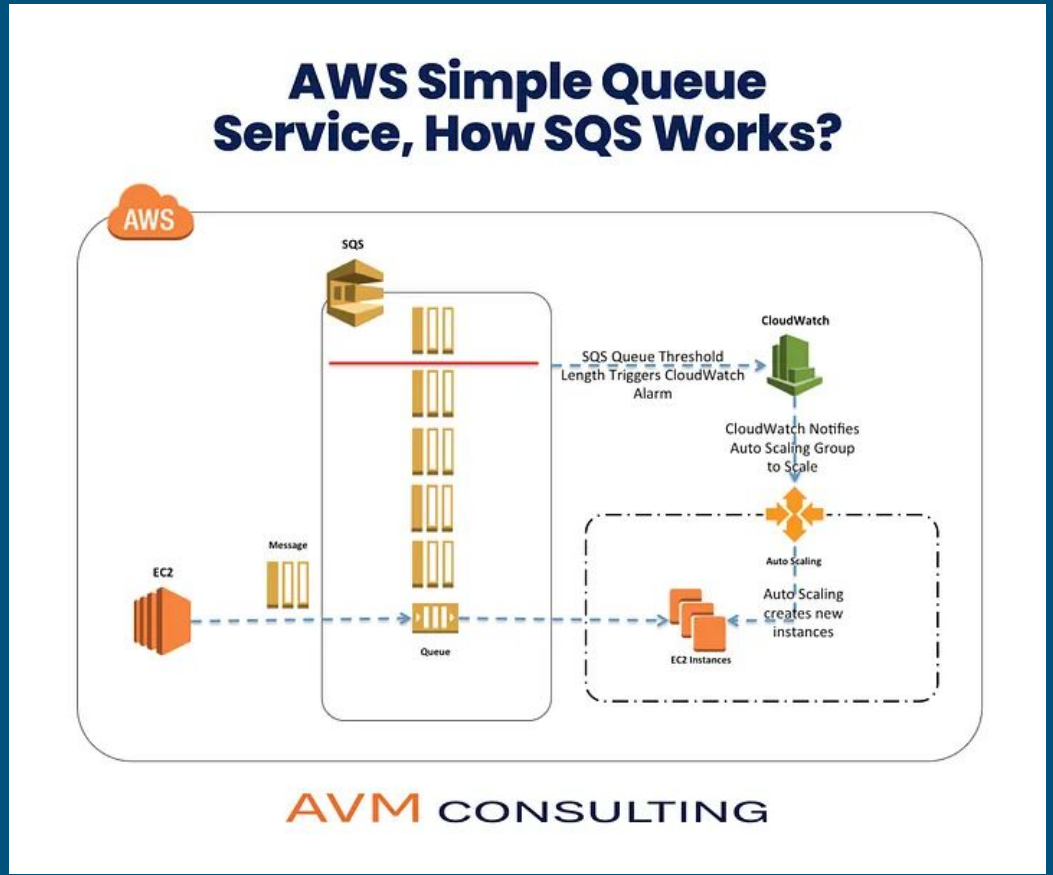
Distributed Computing

- Distributed computing architecture
  - **Client Server architecture**
  - **Three Tier architecture**
  - **N Tier architecture**
  - **Peer to Peer architecture**
- Working of Distributed Computing
  - Works by computers passing messages to each other within the distributed systems architecture.
  - Communication protocols or rules create a dependency between the components of the distributed system.
  - This interdependence is called coupling and there are two main types of coupling
    - **Loose coupling** - Components are weakly connected so that changes to one component do not affect the other
    - **Tight coupling**

Queuing and Messaging

- SQS follows a queue-based messaging system, where messages are sent to a queue and then retrieved by consumers.
- Publisher-subscriber pattern can be implement using SQS in conjunction with other AWS services like SNS
  - **Create an SQS Queue** - Create an SQS queue which serves as the message destination for publishers and the message source for subscribers
  - **Create a Publisher** - Publishers are applications or services that generate messages and send them to the SQS queue. To publish messages, the publisher needs permission to send messages to the SQS queue. This can be achieved by configuring appropriate IAM policies
  - **Send Messages to the Queue** - Publishers send messages to the SQS queue using the SQS API. Messages can be in various formats including JSON, XML or plain text.
  - **Create Subscribers** - Consume messages from the SQS queue. It needs permission to read messages from the SQS queue.This can be achieved by configuring appropriate IAM policies
  - **Configure Subscribers to Poll the Queue** - Subscribers continuously poll the SQS queue to retrieve messages. When a message is available, the subscriber fetches it from the queue and processes it. Long polling can be used to reduce the number of empty responses and to receive messages as soon as they are available.
  - **Process Messages** - Once a subscriber receives a message from the SQS queue, it processes the message according to its business logic

Working of Simple Queue Service (SQS)



Publish and Subscribe

- **Create an SNS Topic**
  - Topic is a communication channel to which messages can be published.
  - An SNS topic can be created to start
- **Subscribe SQS Queue to SNS Topic**
  - Subscribe an SQS queue to the topic
  - Any message published to the SNS topic will be automatically forwarded to the SQS queue
  - SQS queue ARN should be specified as an endpoint when creating the subscription
- **Set Up Permissions**
  - Configuring the required IAM policies to allow SNS to publish messages to the SQS queue
- **Create Publishers**
  - Publishers are applications or services that generate messages and send them to the SNS topic
  - Configure appropriate IAM policies
- **Publish Messages to SNS Topic**
  - Publishers send messages to the SNS topic using the SNS API and is forwarded to all the subscribed endpoints including the SQS queue
- **Process Messages**
  - Subscribers continuously poll the SQS queue to retrieve messages
  - Subscriber fetches the message as soon as it is available in the queue and processes it according to its business logic

Scalability Capabilities and Usecases

Scalability Feature	Use Cases
Auto Scaling	Web servers with fluctuating traffic,Batch processing tasks,Microservices architectures
Elastic Load Balancing (ELB)	Web applications,APIs,Microservices
AWS Lambda	Event-driven applications,Real-time file processing,Background tasks
Amazon Aurora Auto Scaling	Database workloads with unpredictable traffic patterns
Amazon DynamoDB Auto Scaling	Applications with fluctuating read and write demands
Amazon S3	Data storage,Backup,Content distribution
Amazon ECS	Containerized applications deployment and scaling,Microservices architectures
Amazon SQS	Decoupled and scalable communication between components or microservices
Amazon SNS	Event notification,Messaging between distributed systems

AWS Lambda

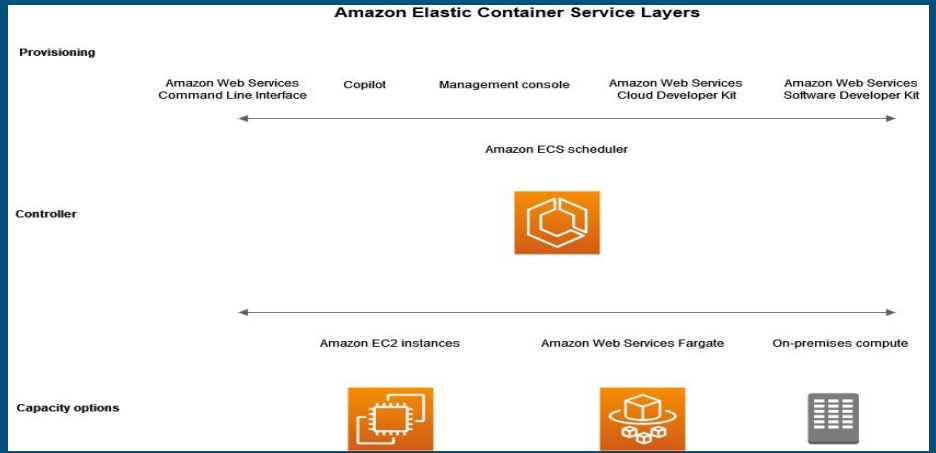
- Compute service that lets you run code without provisioning or managing servers
- Runs code on a high-availability compute infrastructure
- Performs all the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling and logging.
- Supply your code in one of the language runtimes that Lambda supports
- Code is organized into Lambda functions.
- Lambda service runs the function only when needed and scales automatically.
- Charges are based on the compute time consumed
- It can be used for
  - File Processing
  - Stream Processing
  - Web Applications
  - IoT Backends
  - Mobile Backends

AWS Fargate

- Fully managed container orchestration service
- Serverless technology to run Amazon ECS containers
- No need to provision, configure and scale clusters of virtual machines to run containers
- Capacity providers
  - Fargate
  - Fargate Spot
- Offers enhanced security features including support for container security and network isolation
- It can be integrated with **EKS** and **ECS**
- AWS Fargate components
  - **Clusters** - Logical grouping of the tasks and services
  - **Task Definitions** - Text files where the user can describe one or more containers that form the applications. Blueprint of the fargate application
  - **Tasks** - Initialization or Instantiation of the task definition . Multiple tasks can be created from the same task definition. Multiple tasks can run at the same time
  - **Services** - Tool to run the tasks . It can run multiple tasks at a time in a cluster. If the existing task fails or stops , it is replaced by a new task

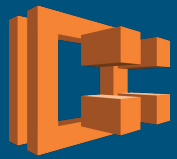
Amazon ECS

- Fully managed container orchestration service that easily deploy, manage and scale containerized applications
- It's built-in with AWS configuration and operational best practices
- Integrated with both AWS and third-party tools such as Amazon Elastic Container Registry and Docker
- Layers in ECS
  - **Capacity** - Infrastructure where containers run
  - **Controller** - Deploy and manage your applications that run on the containers
  - **Provisioning** - Tools to interface with the scheduler to deploy and manage the applications and containers



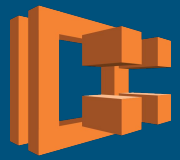
Amazon ECS

ECS Capacity



- Amazon ECS capacity is the infrastructure where the containers run.Capacity option includes
  - **Amazon EC2 instance in the AWS cloud** - Instance type , Number of instances and manage the capacity
  - **Serverless (AWS Fargate)** - Fargate , Pay as you go compute engine. Don't need to manage servers , handle capacity planning or isolate container workloads for security
  - **On-premises virtual machines or servers** - Provides support for registering an external instance such as an on-premises server or VM to the Amazon ECS cluster
- Capacity can be located in any of the following AWS resources
  - Availability Zones
  - Local Zones
  - Wavelength Zones
  - AWS Regions
  - AWS Outposts

Working of Amazon ECS

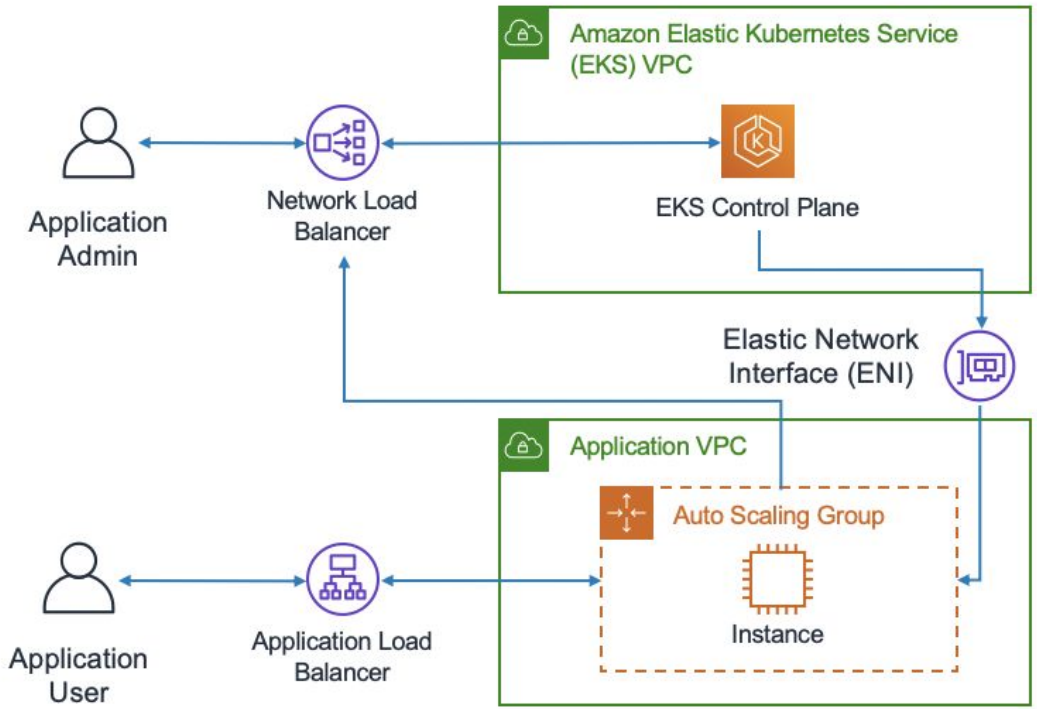


- After a container image is created and stored in the registry(Amazon ECR) , an ECS task definition(Blueprint of the application) has to be created.
- Task definition is a text file in JSON format that describes the parameters and containers that form the application
- Task definition has to be deployed as a service or a task in the cluster.
- Cluster is a logical grouping of tasks or services that runs on the capacity infrastructure that is registered in a cluster
- Task is the instantiation of a task definition within a cluster. Task can be executed as a standalone or a part of a service
- Amazon ECS service can be used to run and maintain the desired number of tasks simultaneously in an ECS cluster
- If any of the tasks fail or stop , ECS service scheduler launches another instance based on the task definition. It replaces to maintain the desired number of tasks in the service
- Container agent runs on each container instance within an ECS cluster
- Agent sends information about the current running tasks and resource utilization of the containers to ECS and start/stop tasks whenever a request is received from ECS
- After the task or service is deployed , following tools can be used to monitor the deployment and application
  - CloudWatch
  - Runtime Monitoring

Amazon EKS

- Fully managed certified Kubernetes conformant service that simplifies the process of building, securing operating and maintaining Kubernetes cluster on AWS
- Integrates with core AWS services such as Cloudwatch,Autoscaling Groups and IAM. It also integrates with AWS App Mesh and provides a Kubernetes-Native experience
- EKS provides scalable , highly-available control plane for Kubernetes workloads
- EKS provides 2 types of compute power for containers
  - EC2 instances
  - AWS Fargate
- Deployment Features
  - **Provision**
    - Load Balancers - If needed
    - Compute Resources - Windows and Linux
    - Application Container Instances - Pods
  - **Configure** - Supports customization of compute resources(workers)
  - **Deploy** - Supports the same deployment strategies as Kubernetes
  - **Scale**
    - Scales workers with Kubernetes Cluster Autoscaler
    - Scales pods with Kubernetes Horizontal Pod Autoscaler and Kubernetes Vertical Pod Autoscaler
  - **Monitor**
    - EKS control plane logs provide audit and diagnostic information directly yo Cloudwatch logs.
    - It integrates with AWS Cloudtrail to records actions taken in Amazon EKS

Amazon EKS working / Use Case



- Fully-managed Kubernetes control plane is accessed by application admins through an NLB with a static IP address
- Control plane monitors and manages application infrastructure
- Users access application from load balancer

Serverless Technologies Patterns

- Several serverless technologies and patterns available on AWS
  - AWS Lambda
  - Amazon API Gateway
  - Amazon DynamoDB
  - Amazon S3
  - Amazon SQS and Amazon SNS
  - AWS Step Functions
  - Amazon EventBridge

Decoupling Workloads

Strategy aimed at improving scalability, resilience and flexibility by breaking down monolithic architectures into smaller, loosely coupled components. Approach to decouple workloads -

- ➔ **Identify Components** - Analyze existing workload to decouple the components
- ➔ **Use Messaging Queues** - Amazon SQS or SNS
- ➔ **Leverage event driven architecture** - AWS Lambda, Amazon EventBridge and Amazon Kinesis
- ➔ **Containerization and Orchestration** - Amazon EKS and Amazon ECS
- ➔ **Data Decoupling** - Amazon RDS, Amazon DynamoDB and or Amazon S3
- ➔ **Microservices Architecture** - Break down application into smaller, independently deployable microservices
- ➔ **Implement Circuit Breakers and Retries**
- ➔ **Use AWS Well-Architected Framework**
- ➔ **Monitoring and Logging** - AWS CloudWatch, AWS X-Ray, or third-party tools
- ➔ **Automate deployments** - AWS CodePipeline, AWS CodeDeploy and AWS CDK

Metrics and Conditions to perform scaling action

- Scaling actions in AWS can be triggered based on various metrics and conditions
  - CPU Utilization
  - Memory Utilization
  - Network Utilization
  - Request Count or Transactions
  - Queue Length
  - Response Time or Latency
  - Custom Application Metrics
  - Scheduled Scaling
  - Combination of Metrics
  - Integration with Auto Scaling Policies

Selecting the appropriate resource type

- **Hosting a web application with variable traffic**
  - EC2 instances with Auto Scaling
  - Elastic Beanstalk
  - AWS Fargate
- **Real-time data processing**
  - AWS Lambda
  - Amazon Kinesis
- **High-performance computing (HPC)**
  - Amazon EC2 instances with optimized CPU and GPU configurations
- **Hosting a database**
  - Amazon RDS (Relational Database Service)
  - Amazon Aurora
  - Amazon DynamoDB
- **Storing files and objects**
  - Amazon S3 (Simple Storage Service)
  - Amazon EFS (Elastic File System)
- **Building and deploying containers**
  - Amazon ECS (Elastic Container Service)
  - Amazon EKS (Elastic Kubernetes Service)
  - AWS Fargate
- **Message queuing and processing**
  - Amazon SQS (Simple Queue Service)
  - Amazon SNS (Simple Notification Service)
- **Building serverless applications**
  - AWS Lambda
  - Amazon API Gateway
- **Content delivery network (CDN)**
  - Amazon CloudFront
- **Managing identity and access**
  - AWS IAM
  - Amazon Cognito
- **Monitoring and logging**
  - Amazon CloudWatch for monitoring
  - AWS CloudTrail for logging and auditing
- **Big data processing and analytics**
  - Amazon EMR (Elastic MapReduce)
  - Amazon Redshift
  - Amazon Athena
  - AWS Glue
- **Machine learning and AI services**
  - Amazon SageMaker
  - AWS AI services (e.g., Rekognition, Comprehend)

AWS Solution Architect Associate

Version : C03

Domain 3

Task 2

The END