**AWS Services and Concepts: Detailed Glossary**

**Networking and Security**

**Access Control List (ACL)**

Access Control Lists operate at the subnet level of your Virtual Private Cloud (VPC), providing a firewall-like security layer that controls traffic entering and exiting your subnets. Unlike Security Groups, ACLs are stateless, meaning you must explicitly define rules for both inbound and outbound traffic. ACLs filter traffic based on rule numbers (processed in numerical order), protocol, ports, and source/destination IP addresses. They provide an additional security layer beyond Security Groups and are particularly useful for blocking specific IP addresses or traffic patterns across an entire subnet.

**Domain Name System (DNS) Server**

A DNS server functions as the internet's phone book, maintaining a database that maps human-friendly domain names (like amazon.com) to machine-readable IP addresses (like 192.0.2.1). When you type a URL in your browser, a DNS query is initiated to translate the domain name into the corresponding IP address, allowing your computer to locate and connect to the correct web server. DNS servers are hierarchical and distributed globally, with root servers at the top, followed by Top-Level Domain (TLD) servers, authoritative name servers, and recursive resolvers that handle the lookup process for users.

**Firewall**

A firewall is a critical security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. It establishes a barrier between a trusted internal network and untrusted external networks (like the internet). Firewalls can be hardware-based, software-based, or both, and they work by examining data packets and determining whether they should be allowed through based on source/destination addresses, ports, and protocols. AWS implements firewalls through Security Groups, Network ACLs, and AWS WAF (Web Application Firewall), providing multiple layers of protection for your cloud resources.

**Principle of Least Privilege**

This fundamental security concept dictates that users, systems, and processes should be granted only the minimum permissions necessary to perform their required functions—and nothing more. In AWS, this means carefully defining IAM policies that provide exactly the access needed for specific tasks without granting excessive permissions. Implementing least privilege reduces the potential attack surface and limits the damage that could occur from compromised credentials or malicious insiders. It's a best practice to regularly audit permissions and remove unnecessary access rights as roles and responsibilities change.

**Security Group (SG)**

Security Groups act as virtual firewalls at the instance level (for EC2 instances, RDS databases, etc.), controlling inbound and outbound traffic. Unlike Network ACLs, Security Groups are stateful—if you allow inbound traffic, the corresponding outbound response is automatically allowed, regardless of outbound rules. Security Groups operate by allowing specified traffic; there is no explicit "deny" capability (anything not explicitly allowed is denied by default). You can reference other security groups as sources in your rules, which is particularly useful in microservice architectures where services need to communicate securely.

**Shared Responsibility Model**

This model clearly delineates which security responsibilities belong to AWS and which belong to you, the customer. AWS is responsible for "security of the cloud" (protecting infrastructure like data centers, hardware, networking, and the virtualization layer). You are responsible for "security in the cloud" (data encryption, resource configuration, identity management, operating system security, network traffic protection). As you move up the service model from IaaS to PaaS to SaaS, AWS assumes more responsibility, but you always retain responsibility for your data, access management, and compliance requirements. Understanding this model is crucial for implementing comprehensive security measures and avoiding dangerous security gaps.

**Compute Services**

**Elastic Compute Cloud (EC2)**

EC2 provides resizable virtual computing capacity in the cloud—essentially virtual servers that you can provision within minutes. Each EC2 instance represents a virtual machine with its own CPU, memory, storage, and networking capabilities, running the operating system of your choice. EC2 offers various instance types optimized for different workloads (compute-optimized, memory-optimized, storage-optimized, etc.) and multiple purchasing options (On-Demand, Reserved Instances, Spot Instances, Savings Plans) to optimize costs. EC2 instances can be integrated with other AWS services and can be configured with custom security settings, storage volumes, and networking features to meet specific application requirements.

**Auto Scaling**

Auto Scaling automatically adjusts the number of EC2 instances in your deployment based on demand, ensuring you have the right amount of compute capacity at all times. It works by defining scaling policies that trigger the addition or removal of instances based on metrics like CPU utilization, network traffic, or custom application metrics monitored through CloudWatch. Auto Scaling helps maintain application availability by replacing unhealthy instances automatically and enables cost optimization by scaling down during periods of low demand. It can be configured with scheduled scaling for predictable load changes or predictive scaling that uses machine learning to forecast future capacity needs.

**Elastic Load Balancing (ELB)**

ELB automatically distributes incoming application traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in multiple Availability Zones. This increases the fault tolerance of your applications and helps handle varying levels of traffic. AWS offers three types of load balancers: Application Load Balancer (for HTTP/HTTPS traffic routing), Network Load Balancer (for ultra-high performance and static IP addresses), and Classic Load Balancer (the original load balancer with basic features). ELB integrates with Auto Scaling to ensure that load balancers are aware of newly launched or terminated instances, providing seamless scaling under varying loads.

**Lambda**

Lambda represents AWS's serverless computing platform, allowing you to run code without provisioning or managing servers. You simply upload your code (in languages like Python, Node.js, Java, etc.), and Lambda handles everything required to run and scale your code with high availability. Lambda functions are triggered by events from various AWS services (like S3, DynamoDB, API Gateway) or direct invocations through the AWS SDK. Billing is based on the number of requests and the duration of code execution (calculated in milliseconds), making it extremely cost-effective for variable workloads. Lambda can replace traditional EC2 instances for many use cases, especially event-driven applications, real-time file processing, and backend services for web and mobile applications.

**Storage Services**

**Simple Storage Service (S3)**

S3 is a highly scalable, durable, and secure object storage service designed to store and retrieve any amount of data from anywhere on the web. Unlike block storage or file systems, S3 stores data as objects (files) within buckets (containers). Each object can range from zero bytes to 5 terabytes and includes the data, metadata, and a unique identifier. S3 offers different storage classes (Standard, Intelligent-Tiering, Standard-IA, One Zone-IA, Glacier, Glacier Deep Archive) with varying levels of availability, durability, and cost. S3 is commonly used for backup and restore, data archiving, content distribution, big data analytics, static website hosting, and as the origin for CloudFront distributions.

**Buckets**

In S3, buckets are the fundamental containers for storing objects (files). Bucket names must be globally unique across all AWS accounts, and they create a namespace at the root level of the S3 domain structure. Buckets can store an unlimited number of objects and serve as the basic unit for access control and usage reporting. They can be configured with various features including versioning, lifecycle policies, server-side encryption, object lock (for WORM compliance), logging, and event notifications. Each bucket is created in a specific AWS region, which affects latency, availability, and compliance with data sovereignty regulations. Bucket names become part of the URL when objects are accessed via the web.

**Folders**

In S3, folders are actually a visual abstraction rather than actual storage containers. Technically, S3 has a flat structure where all objects exist directly in buckets. However, by using prefixes (portions of object keys separated by slashes), S3 provides a folder-like hierarchy in the management console and APIs. For example, an object with the key "reports/2023/january/sales.pdf" appears as if it's in a "january" folder, inside a "2023" folder, inside a "reports" folder—but it's actually a single object with a key containing slashes. These virtual folders help organize objects logically while maintaining S3's flat architecture for performance and scalability.

**Object Availability**

Object availability in S3 refers to the percentage of time over a one-year period that a stored file will be accessible for retrieval operations. Standard S3 storage offers 99.99% availability, meaning objects will be unavailable for only about 53 minutes per year. Reduced availability storage classes like S3 Standard-IA (Infrequent Access) offer 99.9% availability (about 8.8 hours of potential downtime per year) in exchange for lower storage costs. Availability is distinct from durability and is affected by factors such as AWS service outages, network issues, or planned maintenance. S3's availability is backed by the AWS Service Level Agreement (SLA), which offers service credits if AWS fails to meet the promised availability levels.

**Object Durability**

Object durability measures the annual probability that a file stored in S3 will not be lost due to corruption, degradation, or system failures. S3 offers 99.999999999% (11 nines) durability for objects across all storage classes, meaning if you store 10 million objects, you can expect to lose only one object every 10,000 years on average. This extraordinary durability is achieved by automatically storing copies of your data across multiple facilities and devices within a region. For standard S3, data is replicated across at least three Availability Zones, each geographically separated within an AWS Region. Even the lowest-cost Glacier Deep Archive maintains this same durability level, making S3 suitable for long-term preservation of critical data.

**Object Lifecycle**

Object lifecycle management allows you to define rules that automatically transition objects between S3 storage classes or expire (delete) them based on their age. For example, you might configure a lifecycle policy to move objects from Standard storage to Standard-IA after 30 days, then to Glacier after 90 days, and finally delete them after 7 years. Lifecycle rules can be applied to an entire bucket, to objects with specific prefixes (folder-like structures), or to objects with specific tags. This automation helps optimize storage costs by moving less frequently accessed data to lower-cost storage tiers while maintaining appropriate access patterns. Lifecycle policies are particularly valuable for data with predictable access patterns or compliance requirements that dictate retention periods.

**Object Sharing**

Object sharing in S3 allows you to make specific objects accessible to others via direct URL links, without requiring AWS credentials. This can be configured through bucket policies, Access Control Lists (ACLs), or presigned URLs. Public access enables anyone with the URL to access the object, while presigned URLs provide temporary access for a specified duration (from minutes to days). Cross-origin resource sharing (CORS) configurations control which web domains can access your objects directly from browsers. S3 also offers features like S3 Object Lambda, which can transform data upon retrieval, and S3 Access Points, which create customized access policies for different use cases or applications accessing the same bucket.

**Object Versioning**

Versioning in S3 allows you to preserve, retrieve, and restore every version of every object in a bucket, providing an additional layer of protection against accidental deletions or overwrites. When enabled, instead of replacing objects with new uploads, S3 keeps multiple variants of the same object, each with a unique version ID. Deleting an object doesn't actually remove it but places a delete marker, which can be removed to restore access. Previous versions can be accessed directly if you know their version IDs. While versioning provides significant data protection benefits, it increases storage costs as you're storing multiple copies of objects. To manage these costs, you can use lifecycle rules to transition older versions to lower-cost storage classes or expire them after a specified period.

**Elastic Block Store (EBS)**

EBS provides persistent block-level storage volumes designed for use with EC2 instances. Unlike S3's object storage, EBS functions more like a traditional hard drive that you attach to a virtual server. EBS volumes remain independent from the instance lifecycle, allowing data to persist even if the associated instance is terminated. AWS offers several volume types optimized for different workloads: General Purpose SSD (gp2/gp3) for balanced price/performance, Provisioned IOPS SSD (io1/io2) for I/O-intensive applications like databases, Throughput Optimized HDD (st1) for big data and data warehouses, and Cold HDD (sc1) for infrequently accessed workloads. EBS features include point-in-time snapshots (stored in S3), encryption, and the ability to modify volume type, size, and performance characteristics while in use.

**Database Services**

**Relational Database Service (RDS)**

RDS simplifies the setup, operation, and scaling of relational databases in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks like hardware provisioning, database setup, patching, and backups. RDS supports multiple database engines including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and Microsoft SQL Server, allowing you to use the same tools and applications you're familiar with. Key features include automated backups with point-in-time recovery, Multi-AZ deployments for high availability (synchronous standby replicas in a different Availability Zone), Read Replicas for improved read performance and geographic distribution, and database encryption at rest and in transit. RDS handles routine database tasks, allowing you to focus on application development rather than database administration.

**DynamoDB**

DynamoDB is a fully managed, serverless NoSQL database service designed for applications that need consistent, single-digit millisecond performance at any scale. Unlike RDS, which offers various SQL engines, DynamoDB is AWS's proprietary NoSQL solution with a key-value and document data model. It automatically scales throughput capacity up or down based on your application's needs and replicates data across multiple Availability Zones for high availability and durability. DynamoDB features include point-in-time recovery, automated backups, global tables for multi-region replication, encryption at rest, fine-grained access control, and transactions for multiple, all-or-nothing operations. It's particularly well-suited for mobile, web, gaming, ad-tech, IoT, and other applications that need low-latency data access at any scale with minimal operational overhead.

**ElastiCache**

ElastiCache is a fully managed in-memory data store and cache service that supports Redis and Memcached engines. By storing frequently accessed data in memory rather than on disk, ElastiCache dramatically improves the performance and reduces the load on your primary databases. For Redis, ElastiCache provides advanced features like replication for high availability, automatic failover, backup and restore, and Redis Cluster for partitioning data across multiple nodes. For Memcached, it offers a simpler model focused on horizontal scaling through multiple nodes. ElastiCache integrates with other AWS services and is commonly used for real-time applications requiring sub-millisecond response times, such as real-time analytics, caching database queries, session stores, leaderboards, and message queuing.

**RedShift**

Redshift is a fully managed, petabyte-scale data warehouse service optimized for analyzing large datasets using standard SQL. Unlike transactional databases like RDS, Redshift is designed for analytical workloads, using columnar storage, data compression, and massively parallel processing (MPP) to deliver fast query performance on datasets ranging from gigabytes to petabytes. Redshift clusters consist of a leader node that coordinates queries and compute nodes that store data and execute queries in parallel. Features include automated backups, point-in-time recovery, data sharing across clusters or AWS accounts, integration with data lakes through Redshift Spectrum, machine learning integration with Amazon SageMaker, and concurrency scaling to handle thousands of concurrent queries. Redshift is ideal for business intelligence, predictive analytics, and reporting on large datasets.

**Content Delivery and Edge Services**

**CloudFront**

CloudFront is AWS's global content delivery network (CDN) that accelerates the delivery of your websites, APIs, video content, and other web assets by caching copies at edge locations worldwide. When a user requests content, they're automatically routed to the nearest edge location, reducing latency and improving performance. CloudFront integrates with other AWS services like S3, EC2, Elastic Load Balancing, and Lambda@Edge (which allows you to run functions closer to users). Features include SSL/TLS encryption, field-level encryption for sensitive data, geo-restriction capabilities, real-time metrics and logging, customizable cache behaviors, and protection against common web exploits through integration with AWS WAF and AWS Shield. CloudFront's pay-as-you-go pricing model makes it cost-effective for delivering content to users anywhere in the world with high transfer speeds and low latency.

**Monitoring and Management Services**

**CloudTrail**

CloudTrail provides detailed event history of all actions taken within your AWS account, enabling governance, compliance, operational auditing, and risk monitoring. It records API calls made on your account, including calls made through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. For each action, CloudTrail logs who made the request, which services were accessed, what actions were performed, what parameters were requested, and when the action occurred. Events are typically delivered within 15 minutes of activity, and you can configure CloudTrail to deliver log files to an S3 bucket, CloudWatch Logs, or CloudWatch Events for further analysis and automation. CloudTrail is essential for security analysis, resource change tracking, troubleshooting, and maintaining compliance with internal policies and regulatory standards.

**CloudWatch**

CloudWatch is a comprehensive monitoring and observability service that provides data and actionable insights for AWS resources and applications. It collects and tracks metrics (such as CPU utilization, network traffic, and disk I/O), monitors log files, sets alarms, and automatically reacts to changes in your AWS environment. The service has several components: CloudWatch Metrics for time-series data, CloudWatch Logs for log management, CloudWatch Events (now Amazon EventBridge) for event processing, CloudWatch Alarms for threshold-based alerting, and CloudWatch Dashboards for visualization. CloudWatch can monitor AWS resources, applications, and services running on AWS and on-premises servers. It integrates deeply with other AWS services, enabling automated responses to operational changes or events, such as scaling resources or executing Lambda functions when specific conditions are met.

**Trusted Advisor**

Trusted Advisor is a service that provides real-time guidance to help you provision resources following AWS best practices. It analyzes your AWS environment and offers recommendations in five categories: Cost Optimization (identifying idle resources or cost-saving opportunities), Performance (suggestions to improve service performance), Security (detecting potential vulnerabilities), Fault Tolerance (helping improve application resiliency), and Service Limits (notifying when you approach service quotas). The service combines the expertise of AWS solutions architects with automated checks to provide actionable recommendations with estimated monthly savings calculations and step-by-step implementation guidance. Basic Trusted Advisor is available to all AWS customers, while the full suite of checks is available to customers with Business or Enterprise Support plans. Regular reviews of Trusted Advisor recommendations help maintain a well-architected AWS environment.

**Identity and Access Management**

**Identity and Access Management (IAM)**

IAM is a web service that helps you securely control access to AWS resources. It allows you to create and manage AWS users and groups, and use permissions to allow or deny their access to AWS resources at a granular level. IAM is built on a principle of "deny by default" – users have no access until explicitly granted permissions through policies. IAM provides several key elements: Users (individual people or applications needing AWS access), Groups (collections of users with common permissions), Roles (sets of permissions that can be assumed by users or services temporarily), and Policies (documents that define permissions). IAM also supports features like multi-factor authentication (MFA), identity federation with external identity providers (SAML, OIDC), credential rotation policies, and detailed access auditing through CloudTrail. IAM is a foundational service for implementing security best practices in AWS, including least privilege access and separation of duties.

**IAM Users**

IAM Users are entities created within AWS, representing the person or application that interacts with AWS services. Each IAM user has a unique name within your AWS account and a set of security credentials not shared with other users. These credentials can be a password for AWS Management Console access, access keys for programmatic access (API, CLI, SDK), or SSH keys for AWS CodeCommit. IAM users receive permissions through attached policies that specify which AWS actions they can or cannot perform on which resources. Users can be organized into groups for easier permission management, and they can be assigned tags for attribute-based access control. Best practices include creating individual IAM users for each person needing access, enforcing strong password policies, enabling MFA, and regularly rotating credentials.

**User Credentials**

IAM user credentials are the authentication methods used to verify a user's identity when accessing AWS resources. These include: Console passwords (for signing in to the AWS Management Console), Access keys (consisting of an access key ID and secret access key for programmatic access via the AWS API, CLI, or SDKs), SSH keys (for AWS CodeCommit repositories), and server certificates (for SSL/TLS connections to AWS). AWS enforces security best practices for credentials, including password complexity requirements, temporary security credentials, and access key rotation. The AWS Security Token Service (STS) can provide temporary credentials with limited privileges for users who federate from an identity provider or assume IAM roles. Proper credential management—including regular rotation, avoiding credential sharing, and removing unused credentials—is crucial for maintaining AWS account security.

**Roles**

IAM Roles are AWS identity entities with specific permissions that determine what actions the entity can perform. Unlike IAM users, roles don't have associated long-term credentials; instead, they provide temporary security credentials for sessions that last from minutes to hours. Roles serve several purposes: allowing AWS services to act on your behalf (service roles), providing cross-account access (enabling users from one AWS account to access resources in another), enabling federated user access (allowing users with identities outside of AWS to assume roles after authenticating with their identity provider), and implementing instance profiles (roles for EC2 instances). When a user or service assumes a role, AWS provides temporary security credentials that include the role's permissions. Roles are essential for implementing the principle of least privilege and reducing the security risks associated with long-term credentials.

**Organizational Management**

**Organizations**

AWS Organizations is a service that enables you to centrally manage and govern multiple AWS accounts as a single unit. It provides account management and consolidated billing capabilities, with hierarchical organization through Organizational Units (OUs) that can reflect your company's structure. Key features include: Service Control Policies (SCPs) that centrally control permissions for all accounts, Tag Policies that standardize tags across accounts, integration with other AWS services for cross-account capabilities, consolidated billing with volume discounts across all accounts, and automated account creation for scalable, secure multi-account environments. Organizations helps implement common security and compliance requirements across accounts, simplifies resource sharing, and provides better visibility into resource usage and costs. It's particularly valuable for enterprises, partners managing client accounts, and organizations with complex compliance requirements.

**Consolidated Billing**

Consolidated billing is a feature of AWS Organizations that allows you to combine the usage across all accounts in your organization to share volume pricing discounts, Reserved Instance discounts, and Savings Plans. A single account serves as the management (payer) account, responsible for paying the charges of all linked member accounts. This consolidation provides several benefits: simplified accounting with a single monthly bill for all accounts, the ability to track charges and allocate costs across accounts, combined usage across accounts for volume discounts (starting at 10TB of S3 standard storage, the price per GB decreases), sharing of Reserved Instances and Savings Plans across accounts in the same region, and centralized view of all costs through AWS Cost Explorer and AWS Budgets. While billing is consolidated, each account maintains its independence for resource creation and management, preserving security boundaries between accounts.

**Messaging and Notification**

**Simple Notification Service (SNS)**

SNS is a managed messaging service for communication between decoupled microservices, distributed systems, and serverless applications. It implements a publish/subscribe (pub/sub) model where publishers send messages to topics, and subscribers receive messages from topics they're interested in. SNS delivers messages through multiple protocols including HTTP/S, email, SMS, SQS, and mobile push notifications. Key features include: message filtering (so subscribers receive only messages matching specific attributes), message attributes for metadata, message archiving and analytics, dead-letter queues for undelivered messages, and automatic retries with exponential backoff. SNS can be triggered by various AWS services like CloudWatch Alarms, Lambda, and S3 event notifications. It's designed for high-throughput, many-to-many messaging with fanout capabilities (sending the same message to multiple endpoints) and is often used for application alerts, monitoring notifications, and workflow processing.

**Publishers**

In the context of SNS, publishers are entities that send messages to SNS topics. A publisher can be an AWS service (like CloudWatch Alarms, Lambda, or S3), an application component, or a monitoring system that needs to send notifications. Publishers create messages and send them to a topic without any knowledge of who will receive them—this decoupling is a key architectural advantage. Publishers can include message attributes that enable filtering at the subscriber level, avoiding unnecessary message delivery. They can also specify message structures that determine how the message will be formatted for different endpoints (e.g., different formats for email, SMS, and mobile push). AWS services that integrate with SNS can automatically publish messages when specific events occur, such as when objects are created in S3 buckets, when CloudWatch alarms change state, or when CloudFormation stacks are updated.

**Topics**

SNS Topics are communication channels that publishers send messages to and subscribers receive messages from. Each topic is uniquely identified by its Amazon Resource Name (ARN) and can support up to 12.5 million subscriptions. Topics provide an access point that decouples message senders from receivers, allowing for highly scalable, flexible notification delivery. Topics can be configured with a delivery policy that specifies message delivery retry strategies, a message retention period for undelivered messages, and an optional dead-letter queue for messages that couldn't be delivered after repeated attempts. Topics can be encrypted with AWS KMS keys for sensitive information, and access to topics is controlled through IAM policies or topic policies. A single AWS account can create up to 100,000 standard topics (higher limits available upon request), making SNS suitable for large-scale notification systems.

**Subscriptions**

SNS Subscriptions represent the endpoints that receive messages published to topics. When you create a subscription, you specify both the topic and the endpoint that should receive the notifications. SNS supports multiple endpoint types: Amazon SQS queues, AWS Lambda functions, HTTP/S webhooks, email addresses, SMS phone numbers, mobile push notifications, and the Firehose delivery stream. Each subscription can include filter policies that determine which messages from the topic will be delivered to that endpoint. Subscription features include raw message delivery (bypassing SNS's JSON formatting for endpoints that need the original message format), subscription confirmation for HTTP/S, email, and SMS endpoints (requiring endpoint owners to confirm they want to receive messages), delivery status logging, and cross-account subscriptions. Subscriptions can be created, listed, and managed through the AWS Management Console, CLI, or SDK.

**Cloud Architecture Concepts**

**Elasticity**

Elasticity refers to a system's ability to automatically expand or contract its resource capacity in response to changes in demand. In AWS, elasticity is implemented through services like Auto Scaling (for compute resources), DynamoDB auto scaling (for database throughput), and other services that can adjust capacity dynamically. The key concept is that resources are added or removed automatically without manual intervention, often in response to metrics like CPU utilization, request rates, or network traffic. Elasticity differs from simple scalability in that it works in both directions—not just growing but also shrinking to reduce costs during periods of low demand. This automatic adjustment helps maintain performance during traffic spikes while optimizing costs during quieter periods. Effective elasticity requires proper monitoring, well-defined scaling policies, and applications designed to handle resources being added or removed gracefully.

**Scalability**

Scalability refers to a system's ability to handle growing amounts of work by adding resources, whether that growth is planned or unexpected. There are two main types of scalability: Vertical scaling (scaling up) involves increasing the capacity of individual resources (like upgrading to a more powerful EC2 instance), while horizontal scaling (scaling out) involves adding more resources of the same size (like adding more EC2 instances behind a load balancer). AWS services are designed with scalability in mind—storage services like S3 can handle virtually unlimited data, databases like DynamoDB can scale to handle millions of requests per second, and compute services like EC2 can scale from one to thousands of instances. Well-architected AWS applications implement loosely coupled components, stateless applications, distributed processing, and caching strategies to enable seamless scaling. Unlike elasticity (which is automatic and bidirectional), scalability represents the overall system capability to grow, which might require manual planning and configuration.

**Fault Tolerance**

Fault tolerance is the property that enables a system to continue operating properly even when one or more of its components fail. In AWS, this is achieved through redundancy, failover mechanisms, and architectural design patterns that eliminate single points of failure. Key fault-tolerant features include: Multi-AZ deployments (replicating resources across isolated Availability Zones), data replication (maintaining synchronized copies of data across different storage systems), automated instance recovery (replacing failed instances), and automatic failover (rerouting traffic away from failed components). Services like RDS Multi-AZ, DynamoDB global tables, and S3's 11 nines of durability are designed with built-in fault tolerance. To maximize fault tolerance, applications should implement loose coupling, stateless design, idempotent operations, and graceful degradation. The goal is not just to recover from failures but to continue providing service—perhaps with reduced capacity or functionality—even while components are being restored.

**High Availability**

High availability describes systems designed to operate continuously without failure for extended periods. While fault tolerance focuses on continued operation during component failures, high availability encompasses broader concerns including planned maintenance, human error, and external dependencies. AWS enables high availability through: Regional services (deployed across multiple Availability Zones), global services (operating across multiple Regions), automated monitoring and recovery, redundant infrastructure, and backup strategies. A highly available architecture in AWS typically includes load balancers distributing traffic across multiple instances, data replication across zones or regions, regular automated backups, and intelligent monitoring with automated remediation. The AWS Service Level Agreements (SLAs) provide availability commitments for various services, and availability is measured as a percentage of uptime over a period (e.g., 99.99% availability translates to less than 53 minutes of downtime per year). Achieving true high availability requires addressing all potential failure points, including application code, infrastructure, network connectivity, and operations processes.

**Route and Domain Management**

**Route 53**

Route 53 is AWS's highly available and scalable Domain Name System (DNS) web service. It performs three main functions: domain registration (allowing you to register and manage domain names), DNS routing (translating domain names like example.com into IP addresses), and health checking (monitoring your resources and routing traffic only to healthy endpoints). Route 53 supports various routing policies including: Simple routing (for single resources), Weighted routing (for load balancing across resources with specified proportions), Latency-based routing (directing users to the lowest-latency resource), Geolocation routing (routing based on user location), Geoproximity routing (routing based on resource location and optional bias), Failover routing (for active-passive setups), and Multivalue answer routing (returning multiple healthy resources). Route 53 is designed for 100% availability, with DNS servers distributed globally and integrated with other AWS services like Elastic Load Balancing, CloudFront, S3, and API Gateway for complete application routing solutions.

**Networking**

**Virtual Private Cloud (VPC)**

A VPC is a logically isolated section of the AWS cloud where you can launch resources in a virtual network that you define. It gives you complete control over your virtual networking environment, including IP address ranges, subnets, route tables, and network gateways. Key VPC components include: Subnets (segments of IP address ranges in your VPC assigned to specific Availability Zones), Route tables (rules that determine where network traffic is directed), Internet Gateways (for communication between VPC resources and the internet), NAT Gateways (allowing private subnets to access the internet while remaining private), VPC Endpoints (for private connectivity to supported AWS services), and Security Groups and Network ACLs (for security at the instance and subnet levels). VPCs can be connected to other networks through VPN connections, AWS Direct Connect, or VPC peering. They provide the foundation for securely deploying multi-tier applications, implementing network segmentation, and creating hybrid cloud architectures.

**Subnet**

A subnet is a segmented portion of a VPC's IP address range where you place groups of isolated resources. Each subnet resides entirely within one Availability Zone, providing proximity-based resource grouping and isolation from failures in other zones. Subnets can be categorized as public (with routes to the internet via an Internet Gateway) or private (without direct internet access). Public subnets typically contain resources like load balancers and bastion hosts, while private subnets hold application servers and databases for enhanced security. When you create a subnet, you specify its IP address range as a CIDR block (e.g., 10.0.1.0/24), which must be a subset of the VPC's CIDR block. Each subnet has its own route table that controls traffic routing and can have specific Network ACLs for additional security. AWS reserves the first four IP addresses and the last IP address in each subnet for internal networking purposes.