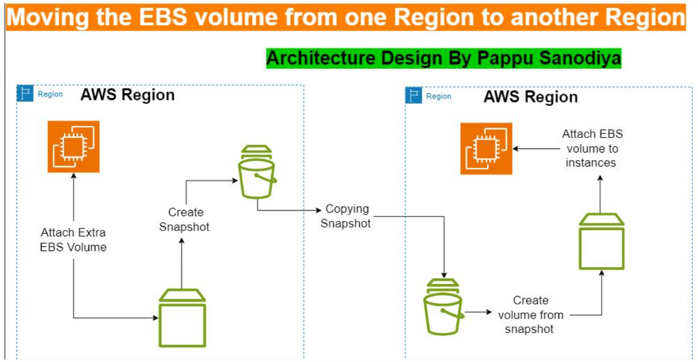
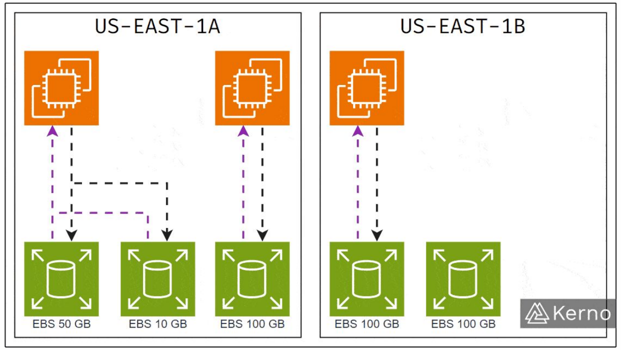
EBS

Amazon EBS (Elastic Block Store) is a block storage service for EC2 instances that provides persistent, high-performance storage.

- Persistence: Data is retained even if the EC2 instance is stopped or terminated.
- Snapshots: Supports point-in-time backups stored in Amazon S3 for recovery, cloning, or migration.
- Encryption: Provides encryption at rest and in transit to secure sensitive data.
- Multi-Attach: Allows certain EBS volume types to be attached to multiple EC2 instances at the same time, useful for high-availability scenarios.

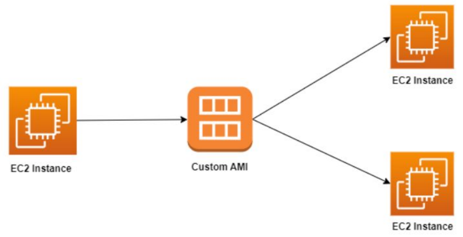

Moving the EBS volume from one Region to another Region
Architecture Design By Pappu Sanodiya
Region
AWS Region
Region
AWS Region
Attach Extra EBS Volume
Create Snapshot
Copying Snapshot
Attach EBS volume to instances
Create volume from snapshot



AMI

Amazon Machine Image (AMI) is a template that contains the software configuration required to launch an EC2 instance.

- Preconfigured Image: An AMI includes the operating system, application server, applications, and any custom settings.
- Fast Launch: Using an AMI lets you quickly launch new EC2 instances with the same setup.
- Custom AMIs: You can create your own AMIs from a configured instance to standardize deployments or save time.
- Backup & Recovery: AMIs are commonly used to back up server states or replicate environments across regions or accounts.
- AMI Types: There are Amazon-provided, Marketplace, and custom user-created AMIs available for various use cases.



Instance Store

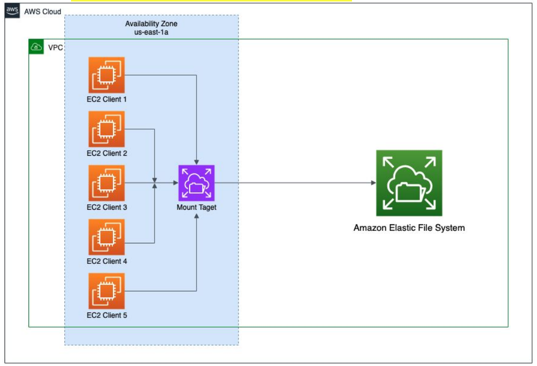
Amazon EC2 Instance Store provides temporary block-level storage for EC2 instances, physically attached to the host machine.

- High Performance: Instance store offers very fast I/O, making it suitable for temporary storage like buffers, caches, or scratch data.
- Ephemeral: Data is lost when the instance stops, hibernates, or terminates. It’s not persistent like EBS.
- No Snapshots: You cannot take snapshots or back up data from instance store volumes.

EFS

Amazon EFS is a scalable, fully managed network file system that can be mounted across multiple EC2 instances.

- Shared Access: Multiple EC2 instances across Availability Zones can access the same file system concurrently.
- Fully Managed: EFS handles provisioning, scaling, and maintenance automatically.
- Scalable & Elastic: Storage grows and shrinks automatically as files are added or removed—no need to pre-provision size.
- POSIX-Compatible: Supports standard file system semantics, making it easy to use with Linux-based applications.
- Durable & Available: Data is automatically stored across multiple AZs for high availability and durability.



FlipTheScript

EBS + EFS Exam Questions

A company needs a shared file system that can be mounted concurrently by multiple EC2 instances across different Availability Zones.

Which storage solution meets this requirement?

A. Amazon EBS

B. Amazon EBS with Multi-Attach

C. Amazon S3

D. Amazon EFS

A company runs a web application on a single EC2 instance with a General Purpose SSD (gp2) EBS volume. Due to increased traffic, the application now requires higher storage throughput and capacity. The solution should have minimal downtime and avoid data loss.

What should a Solutions Architect do to meet these requirements?

A. Stop the instance, create a new larger EBS volume, and attach it

B. Create a snapshot of the volume, then restore it as a larger volume

C. Modify the existing EBS volume to increase its size and switch to gp3

D. Detach the volume, resize it manually, and reattach it

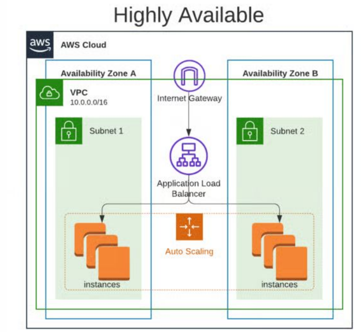
High Availability and Scalability

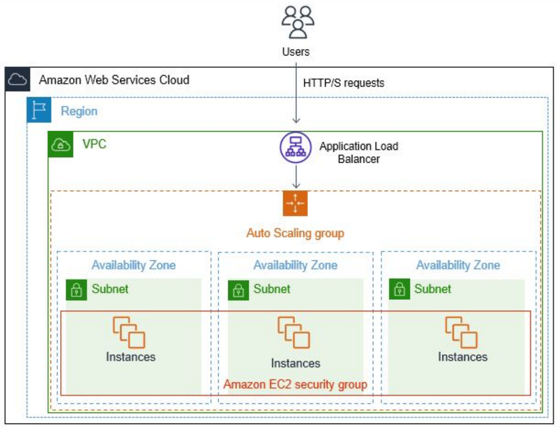
High Availability means designing systems to minimize downtime and ensure continuous operation, even in the event of failures.

- Redundancy: Resources are deployed across multiple Availability Zones (AZs) to avoid single points of failure.
- Auto Recovery: Some AWS services like RDS, Elastic Load Balancer, and ECS offer built-in failover. For EC2, high availability must be manually architected using tools like Auto Scaling Groups and Elastic Load Balancers.

Scalability is the ability of a system to handle increasing workloads by adding resources either vertically or horizontally.

- Vertical Scaling: Increasing the size (CPU, RAM) of an existing instance.
- Horizontal Scaling: Adding more instances or nodes to distribute the load.

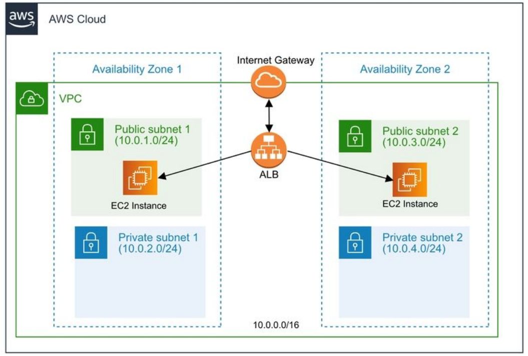




Application Load Balancer (ALB)

An Application Load Balancer is a Layer 7 (HTTP/HTTPS) load balancer that distributes incoming application traffic across multiple targets (like EC2, ECS, or Lambda) within one or more Availability Zones.

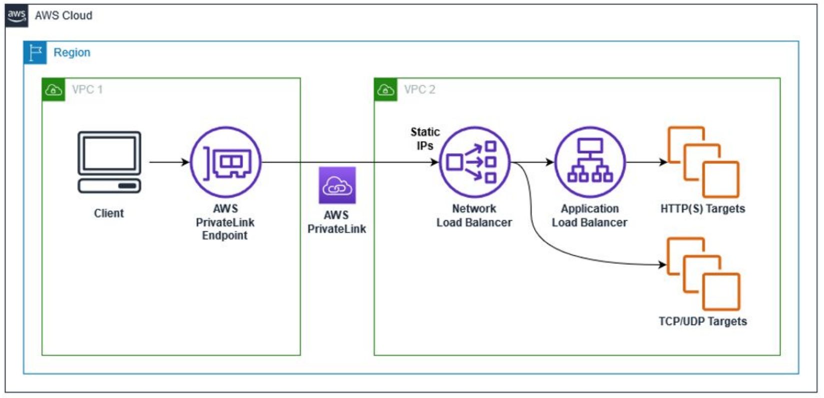
- Layer 7 Load Balancing: Operates at the application layer (OSI layer 7), allowing routing based on content like path or host-based rules.
- Target Groups: Traffic is routed to registered targets (EC2 instances, IPs, Lambda) defined in target groups.
- Health Checks: Continuously monitors the health of targets and routes traffic only to healthy ones.
- Multi-AZ Support: Spans multiple AZs for high availability and fault tolerance.
- Security Integration: Works with Security Groups, AWS WAF, and supports SSL termination (HTTPS).
- Path-based & Host-based Routing: Can route /api/* to one set of instances and /images/* to another, or route app1.example.com vs app2.example.com.


AWS Cloud
Availability Zone 1
Internet Gateway
Availability Zone 2
VPC
Public subnet 1 (10.0.1.0/24)
ALB
Public subnet 2 (10.0.3.0/24)
EC2 Instance
EC2 Instance
Private subnet 1 (10.0.2.0/24)
Private subnet 2 (10.0.4.0/24)
10.0.0.0/16

Network Load Balancer (NLB)

A Network Load Balancer operates at Layer 4 (Transport Layer) and is designed to handle millions of requests per second with ultra-low latency.

- Layer 4 Load Balancing: Routes TCP, UDP, and TLS traffic based on IP protocol data, not HTTP content.
- Extreme Performance: Built to handle sudden and volatile traffic patterns while maintaining high throughput and ultra-low latency.
- Static IP or Elastic IP: Each NLB provides a static IP per AZ and can be associated with an Elastic IP.
- Targets: Can route traffic to EC2 instances, IP addresses, or an ALB.

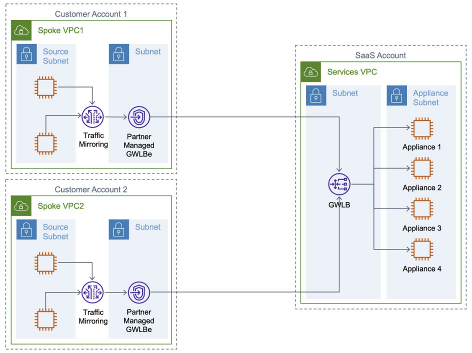


FlipTheScript

Gateway Load Balancer (GWLB)

A Gateway Load Balancer operates at Layer 3 (Network Layer) and is designed to simplify the deployment and scaling of transparent network appliances such as firewalls, intrusion detection/prevention systems (IDS/IPS), and deep packet inspection tools.

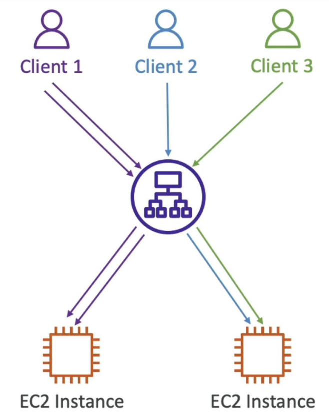
- Layer 3 Load Balancing: Uses the Geneve protocol (Generic Network Virtualization Encapsulation) over UDP port 6081 to forward traffic.
- Security Use Cases: Commonly used to insert security appliances into the network path for inspection and protection.
- Target Groups: Can route traffic to either EC2 instances or private IP addresses that represent network appliances.
- Transparent Network Gateway – acts as a single entry and exit point for all traffic needing inspection.



Sticky Sessions (Session Affinity)

Sticky sessions ensure that all requests from a single user are consistently routed to the same backend target during their session.

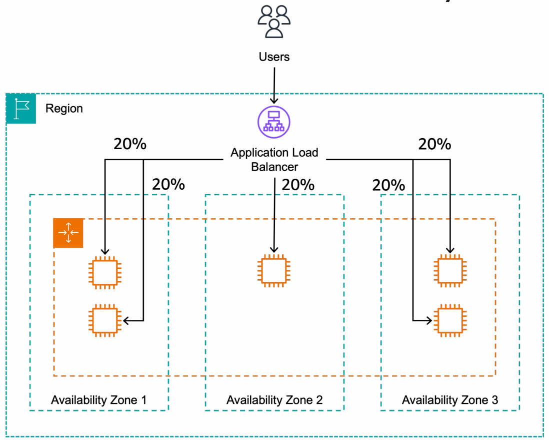
- Session Binding: The load balancer uses a cookie to associate a client with a specific target in the target group.
- Consistent User Experience: Maintains session data (like a shopping cart or login state) by routing traffic to the same instance.
- ALB Support: Uses an AWSALB cookie; supports stickiness at the Target Group level with configurable cookie duration (1 second to 7 days).



Cross-Zone Load Balancing

Cross-zone load balancing allows the load balancer to distribute incoming traffic evenly across all registered targets in all Availability Zones, regardless of which AZ receives the request.

- Even Distribution: Prevents targets in one AZ from being overloaded when other AZs have more capacity.
- ALB: Enabled by default and free of charge.
- NLB: Disabled by default; must be manually enabled and cross-AZ traffic incurs additional cost.
- Improved Resilience: Ensures balanced traffic and availability across zones, especially when target counts are uneven.



FlipTheScript

Load Balancer SSL Certificates

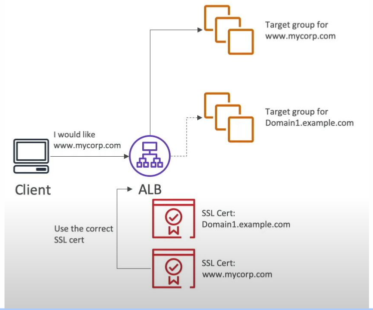
SSL certificates enable AWS Load Balancers to terminate HTTPS connections and securely handle encrypted traffic between clients and the load balancer.

- HTTPS Listener Configuration: You create an HTTPS listener and associate it with an SSL certificate to handle encrypted traffic.
- TLS Termination at the Load Balancer: The SSL/TLS connection is terminated at the load balancer, which decrypts the traffic and forwards it unencrypted to backend targets.
- Certificate Management: SSL/TLS certificates are issued and managed using AWS Certificate Manager (ACM) or can be imported manually.
- ALB Support: Application Load Balancers fully support SSL termination using ACM certificates, including automatic renewal.
- NLB Support: Network Load Balancers support TLS listeners and certificate attachments, but configuration must be done using the AWS CLI, API, or IaC tools (not via the Console UI).
- Security Benefit: Ensures encrypted communication between clients and the load balancer, protecting sensitive data in transit.

SNI (Server Name Indication)

Server Name Indication (SNI) is an extension of the TLS protocol that allows a client to indicate the hostname it is trying to connect to during the SSL handshake, enabling the load balancer to present the appropriate SSL certificate.

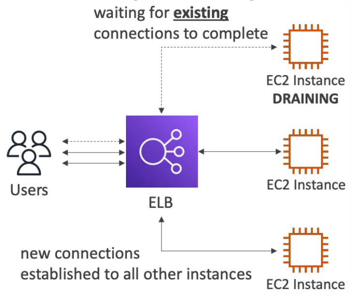
- Multiple SSL Certificates: SNI allows a single HTTPS listener on an ALB or NLB to serve multiple domains by associating multiple SSL certificates with the same listener.
- TLS Negotiation: During the TLS handshake, the client specifies the hostname, and the load balancer selects the correct certificate accordingly.
- ALB Support: Fully supports SNI via ACM-managed certificates, allowing you to serve multiple secure applications on the same port.
- NLB Support: Supports TLS listeners and SNI via certificate configuration using CLI or API.
- Use Case: Hosting multiple HTTPS applications (e.g., app.example.com, api.example.com) behind the same load balancer on port 443.


Target group for www.mycorp.com
Target group for Domain1.example.com
I would like www.mycorp.com
Client
ALB
Use the correct SSL cert
SSL Cert: Domain1.example.com
SSL Cert: www.mycorp.com

Connection Draining (Deregistration Delay)

Connection draining ensures that when a target (such as an EC2 instance) is removed from a load balancer's target group, the load balancer stops sending new requests to it but allows existing in-flight requests to complete before fully deregistering the target.

- Graceful Shutdown: Helps avoid cutting off active user sessions or API calls during instance termination, scaling events, or deployments.
- ALB & NLB: Use a setting called deregistration delay, which by default is 300 seconds (can be configured between 0–3600 seconds).
- Applies Per Target Group: Each target group can have its own deregistration delay settings, giving flexibility across different applications.
- Use Case: Ensures smooth scaling and deployment by allowing backends to shut down without interrupting user traffic.


waiting for existing connections to complete
EC2 Instance
DRAINING
Users
ELB
EC2 Instance
new connections established to all other instances
EC2 Instance

AWS Load Balancer Exam Questions

A Solutions Architect is designing a fault-tolerant web application behind a load balancer. The app is deployed across multiple Availability Zones, but traffic is not evenly distributed when an AZ has fewer healthy targets.

What should the architect enable to ensure even traffic distribution across all AZs?

A. Sticky sessions

B. Health checks

C. Cross-zone load balancing

D. DNS failover

A company needs to route incoming HTTPS requests to different microservices based on the URL path (e.g., /api/v1, /login, /admin).

Which load balancer should the Solutions Architect choose?

A. Network Load Balancer (NLB)

B. Application Load Balancer (ALB)

C. Classic Load Balancer (CLB)

D. Gateway Load Balancer (GWLB)

Auto Scaling Group (ASG)

An Auto Scaling Group automatically manages a fleet of EC2 instances to meet demand and maintain availability, based on defined rules and metrics.

- Automatic Scaling: Launches or terminates EC2 instances based on target utilization or CloudWatch alarms.
- Launch Template / Launch Configuration: Defines how new instances are created (AMI, instance type, key pair, security groups, etc.).
- Min/Max/Desired Capacity: You can define the minimum, maximum, and desired number of instances in the group.
- Health Checks: Supports EC2 and ELB health checks to automatically replace unhealthy instances.
- Integration with Load Balancers: Can register new instances with an ALB or NLB, ensuring seamless traffic handling.
- Scaling Policies: Supports target tracking, step scaling, and scheduled scaling based on metrics like CPU, requests, or custom CloudWatch metrics.
- Lifecycle Hooks: Lets you perform custom actions (e.g., software install) before launching or terminating instances.
- Elasticity and High Availability: Ensures applications scale out when needed and reduce costs when demand drops.

Scaling Policies

Scaling policies define when and how an Auto Scaling Group adjusts the number of EC2 instances based on demand, metrics, or schedules.

- Target Tracking Scaling: Dynamically adjusts capacity to maintain a specific target value (e.g., average CPU at 50%).
- Step Scaling: Adjusts the number of instances in steps, based on how much a metric deviates from the threshold (e.g., add 1 instance at 70% CPU, 2 at 90%).
- Simple Scaling: Executes a single scaling action when a CloudWatch alarm is triggered.
- Scheduled Scaling: Scales to a predefined number of instances at specific times (e.g., scale to 6 instances every weekday at 8 AM).
- Custom Metrics: Uses application-specific CloudWatch metrics like memory usage or request count.
- Scaling Cooldown: A pause after a scaling activity during which no further scaling actions are allowed. This gives new instances time to launch and stabilize, preventing unnecessary scaling.
- Default Cooldown Period: By default, the cooldown is 300 seconds (5 minutes), but it can be customized or overridden per scaling policy.

Auto Scaling Groups Exam Questions

A company uses an ASG with EC2 instances registered behind an ALB. The architect wants to ensure that unhealthy instances are replaced only if they fail ALB health checks, not just EC2 status checks.

What should the architect configure?

A. Enable EC2 status checks only

B. Use Elastic Load Balancer health checks in the ASG

C. Attach the instances to the ALB manually

D. Enable instance recovery

A company runs a stateless web application behind an Application Load Balancer and uses an Auto Scaling Group (ASG). They want the number of instances to adjust automatically based on average CPU utilization.

Which scaling policy should be used?

A. Scheduled Scaling

B. Step Scaling

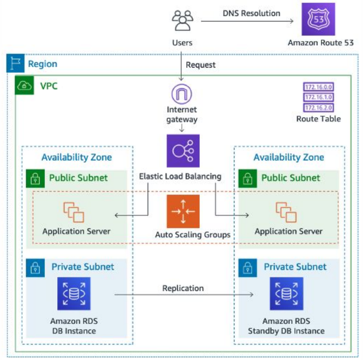
C. Target Tracking Scaling

D. Manual Scaling

Amazon RDS

Amazon RDS is a managed relational database service that simplifies the setup, operation, and scaling of databases in the cloud.

- Fully Managed: Automates database tasks such as provisioning, patching, backups, and recovery.
- Supported Engines: Offers multiple database engines including Amazon Aurora, MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server.
- Scalability: You can scale compute and storage independently with just a few clicks or API calls.
- High Availability: Supports Multi-AZ deployments for automatic failover and durability across Availability Zones.
- Read Replicas: Enables read replicas to offload read traffic and improve performance (for supported engines).
- Security: Integrates with IAM, VPC, KMS, and supports encryption at rest and in transit.
- Monitoring: Offers built-in monitoring via CloudWatch, Enhanced Monitoring, and Performance Insights.

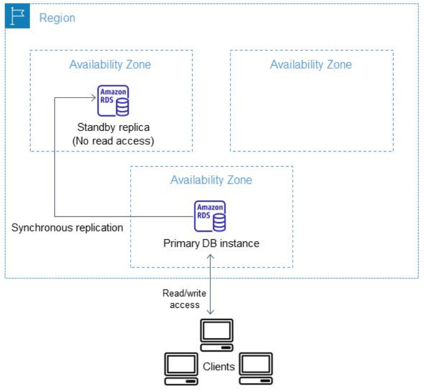


FlipTheScript

Multi-AZ DB - (High Availability)

Multi-AZ provides automatic failover and is designed for high availability and durability, not scaling.

- Failover Protection: Maintains a synchronous standby replica in another AZ; if the primary DB fails, it automatically fails over.
- Same Endpoint: Applications connect using a single endpoint—no changes needed during failover.
- No Performance Benefit: The standby instance cannot be used for reads or writes; it's only for recovery.
- Use Case: Production databases where availability and resilience are critical.

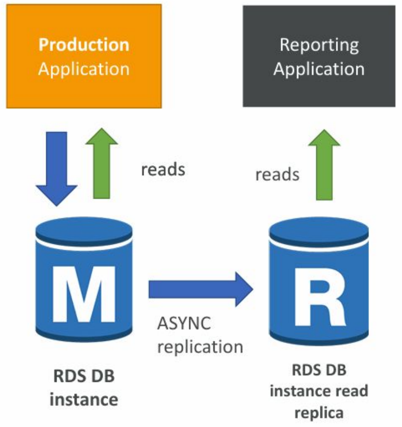


FlipTheScript

Read Replicas DB - (Scalability)

Read replicas are for scaling read-heavy workloads and improving read performance.

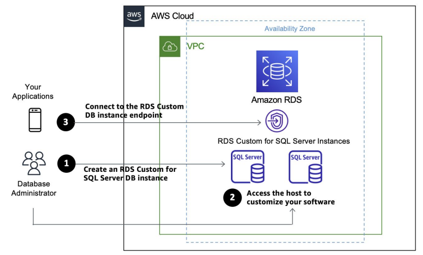
- Asynchronous Replication: Updates are sent from the source DB to replicas with a slight delay.
- Separate Endpoints: Applications must explicitly connect to replicas to perform read queries.
- Multiple Replicas: You can create several replicas (some engines allow up to 15).
- Promotion Option: A read replica can be promoted to become a standalone DB instance.
- Use Case: Applications with high read traffic (e.g., analytics, dashboards) or for offloading reporting queries.



Amazon RDS Custom

Amazon RDS Custom is a managed database service designed for applications that require access to the underlying operating system and database environment, which is not possible with standard RDS.

- Full OS and DB Access: Gives you the ability to access and customize the operating system and database settings. Ideal for running legacy, packaged, or custom database applications.
- Supports Oracle and SQL Server: Currently available for RDS Custom for Oracle and RDS Custom for SQL Server.
- Use Case: Suitable for applications needing features like custom agents, special patches, advanced auditing, or third-party tools installed on the DB host.
- Responsibility Shift: While AWS automates provisioning and monitoring, you are responsible for operations like backups, patching, and high availability, which are automatic in standard RDS.
- EC2-like Control: Offers full administrative privileges, but with some managed benefits like CloudWatch monitoring and automated instance recovery.
- Security: You must take more ownership over OS-level security, patch management, and access controls.


AWS Cloud
Availability Zone
VPC
Amazon RDS
Your Applications
Connect to the RDS Custom DB instance endpoint
3
RDS Custom for SQL Server Instances
SQL Server
SQL Server
1
Create an RDS Custom for SQL Server DB instance
Database Administrator
2
Access the host to customize your software

FlipTheScript

Amazon Aurora

Amazon Aurora is a high-performance, fully managed relational database engine built for the cloud. It is compatible with MySQL and PostgreSQL, but offers enhanced speed, availability, and scalability.

- High Performance: Up to 5x faster than standard MySQL and 3x faster than PostgreSQL, thanks to its distributed and highly optimized storage engine.
- Fully Managed: AWS handles backups, patching, failure detection, and repair without user intervention.
- Storage Auto-Scaling: Storage automatically grows in 10 GB increments, up to 128 TiB.
- Multi-AZ Architecture: Aurora separates compute and storage; storage is spread across 6 copies across 3 Availability Zones, providing high durability and availability.
- Aurora Replicas: Supports up to 15 low-latency read replicas that can be promoted instantly for failover.
- Aurora Serverless: An on-demand, auto-scaling version of Aurora that adjusts capacity based on application needs—ideal for unpredictable workloads.
- Global Databases: Aurora supports global deployments with cross-region replication, useful for disaster recovery and low-latency reads worldwide.

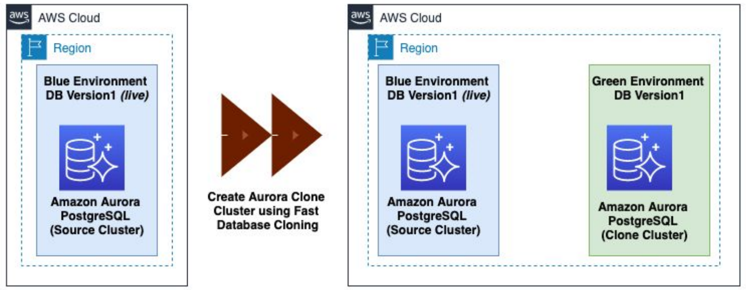
Backups in Amazon RDS and Aurora

- Automated Backups: Both RDS and Aurora support automated backups, allowing point-in-time recovery within a user-defined retention period (up to 35 days).
- Backups are stored in S3.
- No performance impact during backup operations.
- Manual Snapshots: Users can create snapshots manually at any time. These snapshots are stored until explicitly deleted and can be used to restore a DB instance.
- Aurora Continuous Backup: Aurora continuously backs up data to Amazon S3, with no performance impact, allowing restore to any point in time within the backup retention period.
- Shared Snapshots: Both services support snapshot sharing between accounts or AWS Regions for migration or backup purposes.

Aurora Cloning

Aurora Cloning enables you to create a copy of an existing Aurora database cluster almost instantly, without duplicating the entire dataset.

- Fast and Efficient: Aurora uses a copy-on-write model, meaning the cloned cluster shares the same underlying storage as the source and only copies data when changes occur.
- Same Region: Cloning is supported within the same AWS Region and is limited to the same Aurora engine type.
- No Downtime: Cloning does not impact the performance or availability of the source database.
- Use Cases: Ideal for development, testing, analytics, or troubleshooting using real production data without impacting the production environment.
- Differences from Snapshot Restore: Cloning is much faster and more storage-efficient than restoring from a snapshot, which creates a full copy.


AWS Cloud
Region
Blue Environment DB Version1 (live)
Amazon Aurora PostgreSQL (Source Cluster)
Create Aurora Clone Cluster using Fast Database Cloning
AWS Cloud
Region
Blue Environment DB Version1 (live)
Amazon Aurora PostgreSQL (Source Cluster)
Green Environment DB Version1
Amazon Aurora PostgreSQL (Clone Cluster)

RDS Security

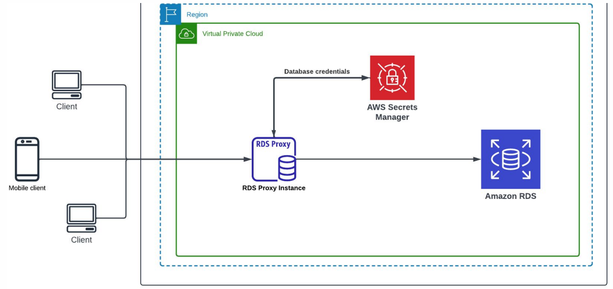
Amazon RDS includes multiple layers of security to protect your databases, including network isolation, encryption, and access control.

- VPC Isolation: RDS runs inside your Amazon VPC, allowing you to isolate databases and control inbound/outbound traffic using security groups, NACLs, and route tables.
- IAM Integration: You can manage authentication and authorization for administrative tasks using IAM policies. IAM database authentication is supported for MySQL and PostgreSQL, allowing users to log in using temporary tokens instead of passwords.
- Encryption at Rest: RDS supports encryption using AWS Key Management Service (KMS). Once enabled at database creation, all data, including backups, logs, and snapshots, is encrypted.
- Encryption in Transit: Data is encrypted using SSL/TLS when traveling between your application and the database instance.
- Integrates with CloudTrail and CloudWatch Logs for auditing login activity, API calls, and query logs.
- Supports database-level audit logs for engines like Oracle and SQL Server.
- Automatic Patching: Minor version patches can be applied automatically during a defined maintenance window to maintain security.

RDS Proxy

Amazon RDS Proxy is a fully managed, highly available database proxy that improves the scalability, availability, and security of RDS and Aurora databases by managing connections more efficiently.

- Connection Pooling: RDS Proxy maintains a pool of established database connections and reuses them across multiple client requests. This reduces the overhead of opening/closing connections and improves application scalability.
- Reduced Failover Time: During a database failover (e.g., in Multi-AZ setups), RDS Proxy can maintain application connections and reroute traffic, helping reduce downtime.
- Supports IAM and Secrets Manager: You can use IAM authentication and AWS Secrets Manager to manage database credentials securely and rotate them automatically without changing the application code.
- Improves Security: Applications connect to the proxy instead of directly to the database. This allows tighter control over credentials, and better protection from compromised applications



FlipTheScript

RDS + Aurora Exam Questions

A company is running an application on Amazon Aurora MySQL and experiences read-heavy traffic during peak hours. They want to improve read scalability without impacting write performance.

Which solution should a Solutions Architect recommend?

A. Increase the instance size of the primary writer node

B. Enable Multi-AZ failover

C. Add Aurora Replicas and configure the application to direct reads to them

D. Create a Read Replica in another AWS Region

A company is running a production MySQL database on Amazon RDS. They want to improve availability and ensure automatic failover in case of an Availability Zone failure.

Which configuration should they choose?

A. Create a Read Replica in another AZ

B. Enable Multi-AZ deployment

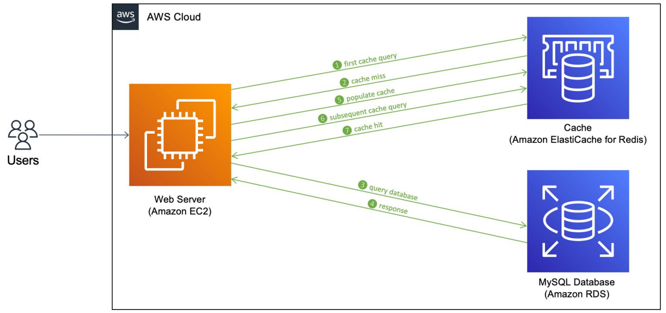
C. Launch another standalone RDS instance and sync manually

D. Use Amazon S3 for cross-region backup

Amazon ElastiCache

Amazon ElastiCache is a fully managed in-memory data store, used to accelerate application performance by retrieving data from fast, low-latency memory instead of slower disk-based databases.

- Purpose: Ideal for use cases that require microsecond response times, such as caching, real-time analytics, session storage, and gaming leaderboards.
- Engines Supported: Offers two engines: Redis and Memcached.
- Managed Service: Automates provisioning, patching, setup, failure recovery, monitoring, and backups (for Redis).
- High Availability: Supports replication, automatic failover, and Multi-AZ deployments (Redis only).
- Scalable: Easily scales out with clustering (Redis) or horizontal scaling (Memcached).
- Integration: Can be used alongside databases (RDS, DynamoDB), analytics systems, and application layers to reduce load and improve latency.



Redis vs Memcached in ElastiCache

Redis

- Data Structures: Supports advanced types like strings, hashes, lists, sets, sorted sets, bitmaps, streams, and geospatial indexes.
- Persistence: Supports data persistence with RDB snapshots and AOF logs, making it suitable for use cases needing durability.
- Replication & Failover: Redis supports replication, automatic failover, and Multi-AZ deployments.
- Clustering: Allows partitioning of data across multiple nodes (sharding) for horizontal scaling.
- Pub/Sub Messaging: Built-in support for publish/subscribe messaging patterns.
- Best For: Session stores, real-time analytics, leaderboard systems, chat applications.

Memcached

- Data Structures: Only supports simple key-value pairs (no advanced types).
- Persistence: In-memory only with no persistence or replication—data is lost on restart.
- Scaling: Scales horizontally by adding/removing nodes, using client-side sharding.
- Simplicity: Easier to deploy and manage with fewer features.
- Best For: Simple caching use cases where high-speed reads/writes and statelessness are needed (e.g., caching database query results).

ElastiCache Security

Amazon ElastiCache provides multiple layers of security to help protect your in-memory data environment:

- VPC Integration: ElastiCache must be launched within an Amazon VPC, allowing you to control access using subnets, route tables, NACLs, and security groups.
- Security Groups: Function like virtual firewalls to control inbound/outbound traffic to cache nodes.
- Redis AUTH: Redis supports AUTH tokens (password protection) to restrict access to authorized clients. These can be managed via the ElastiCache console or API.
- Role-Based Access Control (RBAC): In Redis 6 and above, you can define users and assign them fine-grained access to specific commands.
- In-Transit Encryption: Protects data using TLS (Transport Layer Security) as it moves between clients and ElastiCache nodes.
- At-Rest Encryption: Encrypts data stored on disk (only supported by Redis, not Memcached).
- Managed Keys: Encryption uses AWS Key Management Service (KMS) for key handling.
- IAM policies are used for managing access to ElastiCache resources via the AWS Management Console, CLI, or API (not for connecting to the cache itself).

ElastiCache Patterns

Cache-Aside (Lazy Loading)

The application checks the cache before querying the database.

- If the item exists in the cache, it is returned immediately.
- If the item is not found, the app queries the database, stores the result in the cache, and returns the data.
- Efficient for read-heavy workloads but may serve stale data if the database is updated directly.
- The cache is populated only when needed (lazy).

Write-Through Cache

The application writes data to both the database and the cache simultaneously.

- Ensures the cache is always up to date with the database.
- Simple read logic: always trust the cache.
- May introduce write latency since two operations occur per write.

Session Store

Stores short-lived, user-specific session data (e.g., login state, shopping cart) in memory.

- Ideal for fast access and expiration, with support for TTL (time-to-live) on keys.
- Often used with Redis, which supports persistence and advanced data structures.
- Common in distributed applications where session data must be shared across multiple servers.

ElastiCache Exam Questions

A company is evaluating Amazon ElastiCache for a new application. They require advanced data structures, replication, automatic failover, and encryption.

Which ElastiCache engine should they choose?

A. Memcached

B. Redis

C. DynamoDB

D. Amazon RDS

A company has a high-traffic web application and wants to reduce load on its relational database by caching frequently accessed but rarely changed data. They want the cache to be populated only when needed, not pre-loaded.

Which caching strategy should they implement?

A. Write-through

B. Cache-Aside

C. Write-behind

D. Read-through