■■ Amazon RDS

Relational Database Service

Complete Enterprise Guide

Interview Ready • Production SOP • Architecture • Costing

GUI Steps • POC • GitHub README • Multi-AZ • Aurora

Read Replicas • Backup • Security • Parameter Groups

■■ RDS	**■ Aurora**	**■ Multi-AZ**	**■ Replicas**	**■ Backup**	**■ Security**

■ Table of Contents

1. What is Amazon RDS?

■ Interview Explanation

Amazon RDS (Relational Database Service) is a fully managed relational database service that automates time-consuming administration tasks: hardware provisioning, database setup, patching, and backups. You focus on your application — AWS manages the database infrastructure.

In simple terms: Instead of installing MySQL on a server, managing backups, applying patches, configuring replication, and monitoring disk space — RDS does all of that automatically. You just connect to an endpoint and run queries.

The Perfect Interview Analogy:

"RDS is like hiring a professional DBA (Database Administrator) who works 24/7 for free. They handle backups, patching, replication, failover, monitoring, and scaling. You only worry about writing SQL and designing your schema."

What RDS Manages For You:

- OS patching and database engine version upgrades (scheduled maintenance windows).
- Automated daily backups with point-in-time recovery (PITR) up to 35 days.
- Multi-AZ synchronous replication and automatic failover in under 60 seconds.
- Hardware failure detection and automatic replacement.
- Storage autoscaling — expands storage automatically when threshold reached.
- Read Replica creation for horizontal read scaling across regions.
- SSL/TLS encryption in transit and KMS encryption at rest.
- CloudWatch metrics, Performance Insights, Enhanced Monitoring.

Key Facts:

Fact	Value
Service Type	Managed PaaS — AWS manages OS, DB engine, hardware
Supported Engines	MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, Amazon Aurora
Max Storage	64 TB for MySQL/PostgreSQL/MariaDB; 16 TB for Aurora (auto-grows)
Max IOPS	256,000 IOPS (io1/io2) | Aurora: millions (distributed storage)
Availability	Multi-AZ: 99.95% SLA | Single-AZ: 99.95% (best effort)
Failover Time	Multi-AZ automatic failover: typically 60–120 seconds
Backup Retention	1–35 days automated backups. Manual snapshots kept indefinitely.
Access Method	Standard SQL over TCP (MySQL:3306, PostgreSQL:5432, etc.)

2. RDS Database Engines

RDS supports 6 database engines. Choosing the right engine is critical for performance, cost, and licensing compliance. Aurora is AWS's proprietary cloud-native engine.

■ MySQL

Most popular open-source RDBMS. Versions: 5.7, 8.0. Best for: web apps, e-commerce, CMS (WordPress/Drupal). Free community edition. Max storage: 64 TB. Supports Read Replicas in 5 regions.

■ *Use Aurora MySQL instead of MySQL for production — 5x faster, same cost.*

■ PostgreSQL

Most advanced open-source RDBMS. Versions: 13, 14, 15, 16. Best for: complex queries, JSON/JSONB, geospatial (PostGIS), analytics, data science. Excellent ACID compliance. Extensible with custom functions and types.

■ *Preferred for complex workloads. Aurora PostgreSQL for production.*

■ MariaDB

Community-driven MySQL fork. Versions: 10.5, 10.6, 10.11. Faster than MySQL for some workloads. Better storage engines (Aria, ColumnStore). 100% MySQL compatible. Good for MySQL migrations wanting open-source guarantee.

■ *Good MySQL alternative. Less popular than MySQL/PostgreSQL in cloud.*

■ Amazon Aurora

AWS proprietary cloud-native database. MySQL and PostgreSQL compatible. 3x faster than PostgreSQL, 5x faster than MySQL. 6 copies of data across 3 AZs. Storage auto-grows in 10 GB increments up to 128 TB. Sub-10ms failover. Aurora Serverless v2 for variable workloads. Global Database for multi-region.

■ *ALWAYS prefer Aurora for new production workloads. Best performance and HA.*

■ Oracle

Enterprise RDBMS. Versions: 19c, 21c. Requires BYOL or License Included pricing. Best for: legacy enterprise apps, complex PL/SQL, Oracle Forms applications. Supports Oracle RAC? NO — RDS Oracle is single-instance only (use RDS Custom for RAC).

■ *BYOL: bring your existing Oracle license. License Included: pay per hour including license.*

■ SQL Server

Microsoft RDBMS. Editions: Express, Web, Standard, Enterprise. Versions: SQL Server 2016, 2017, 2019, 2022. Best for: .NET applications, Windows workloads, SharePoint, existing SQL Server databases. Supports SQL Server Agent, Transparent Data Encryption, Always Encrypted.

■ *License Included pricing. Dedicated option available for licensing compliance.*

3. RDS Core Components

■ DB Instance

The compute unit running your database. Defined by instance class (db.t3.micro, db.m5.large, db.r6g.xlarge). Each instance class provides different vCPU, RAM, and network bandwidth. Can be modified (resized) with a maintenance window.

■ DB Instance Class

3 families: Standard (db.m), Memory Optimized (db.r, db.x), Burstable (db.t). Example: db.r6g.4xlarge = Memory Optimized, 6th gen Graviton, 4xlarge size (16 vCPU, 128 GB RAM).

■ DB Subnet Group

Collection of subnets (in different AZs) that RDS can use. Required for VPC deployment. Multi-AZ needs subnets in at least 2 AZs. Always use private subnets.

■ Parameter Group

Configuration settings for the DB engine. Like a config file. Example: max_connections=1000, innodb_buffer_pool_size=8G. Default parameter group is read-only — create custom ones.

■ Option Group

Additional features for specific engines. Example: Oracle Timezone, SQL Server TDE, MySQL memcached. Engine-specific optional configurations.

■ Multi-AZ Deployment

RDS maintains a synchronous standby replica in a different AZ. Automatic failover in 60–120 seconds. Standby is NOT readable — it's only for HA. Enabled with one checkbox.

■ Read Replicas

Asynchronous read-only copies. Up to 5 per source DB (15 for Aurora). Can be in different regions (cross-region). Promote to standalone DB. Reduces read load. Has its own endpoint.

■ Endpoints

DNS hostnames to connect to RDS. Writer endpoint (primary), Reader endpoint (round-robins across replicas), Instance endpoint (specific instance). Aurora cluster has Cluster and Reader endpoints.

■ Security Groups

Virtual firewall for DB instance. Allow inbound on DB port (3306/5432) only from your app tier security group. Never open to 0.0.0.0/0.

■ Maintenance Window

Weekly 30-min window for patching, minor version upgrades. Choose low-traffic time. For Multi-AZ: standby patched first, then failover, then primary — zero downtime.

4. Amazon Aurora — Deep Dive

Aurora is AWS's crown jewel database. It's NOT just managed MySQL/PostgreSQL — it's a completely re-architected cloud-native database that separates compute from storage. For interviews: know Aurora cold, it appears in nearly every database architecture question.

Aurora Architecture — How It's Different:

```
Traditional RDS (MySQL/PostgreSQL): [DB Instance] ■■■■ [EBS Volume in 1 AZ] Replication: Full
redo logs shipped over network Failover: Minutes (detach EBS, attach to standby) Amazon
Aurora: [Writer Instance] [Reader 1] [Reader 2] ■ ■ ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ AURORA DISTRIBUTED STORAGE LAYER ■ ■ 6
storage nodes across 3 AZs ■ ■ AZ-1: Node1, Node2 ■ ■ AZ-2: Node3, Node4 ■ ■ AZ-3: Node5,
Node6 ■ ■ Quorum writes (4/6), reads (3/6) ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
Key insight: Only REDO LOGS are written to storage (not full pages) → 1/10th the I/O of
traditional MySQL → Storage failure tolerant: loses 2 nodes, still reads; loses 3 nodes,
still writes
```

Aurora Key Features:

Feature	Detail
Performance	3x faster than PostgreSQL, 5x faster than MySQL benchmarks
Storage	Auto-grows 10 GB at a time, up to 128 TB. No pre-provisioning.
Replicas	Up to 15 Aurora Replicas. Sub-10ms replica lag. Failover promotes one.
Failover	Sub-30 seconds automatic failover. Application reconnects to cluster endpoint.
Backtrack	Aurora MySQL only. Rewind DB to any point without restore. Up to 72 hours.
Aurora Global Database	One primary region + up to 5 secondary regions. Cross-region DR in <1 second RPO.
Aurora Serverless v2	Auto-scales compute 0.5–128 ACUs in seconds. Pay per second. Ideal for variable load.
Parallel Query	Pushes query processing to storage layer. Speeds up analytics 100x. No separate cluster.
Aurora ML	Native SQL integration with SageMaker and Comprehend for ML inference in SQL.
Zero-Downtime Patching	Most patches applied without restart. True zero-downtime for apps.
Multi-Master	Aurora MySQL: multiple writer instances. Write to any node. Conflict detection.

Aurora Serverless v2 — When to Use:

Aurora Serverless v2 automatically scales compute capacity based on application demand. Scales from 0.5 ACUs (Aurora Capacity Units) to 128 ACUs in fine-grained increments. 1 ACU ≈ 2 GB RAM + proportional CPU and network.

- Development/test environments — scale to near-zero when idle.
- Applications with unpredictable or variable traffic patterns.

- New applications where usage is unknown.
- Multi-tenant SaaS with variable per-tenant load.

5. RDS Storage Types

■ gp2 — General Purpose SSD

Baseline 3 IOPS/GB. Burst up to 3,000 IOPS using credit bucket. Min 20 GB, Max 64 TB. IOPS scales with storage size. Legacy option — gp3 is better in almost all cases.

■ *Migrate to gp3 for cost savings and better IOPS control.*

■ gp3 — General Purpose SSD v3

Baseline 3,000 IOPS and 125 MB/s throughput regardless of storage size. Can provision up to 16,000 IOPS and 1,000 MB/s independently from storage. 20% cheaper than gp2. No burst mechanism — consistent performance.

■ *DEFAULT choice for most workloads. Better performance, lower cost than gp2.*

■ io1 / io2 — Provisioned IOPS SSD

High-performance for I/O-intensive workloads. io1: up to 64,000 IOPS (256,000 for Nitro instances). io2: higher durability (99.999%) and same IOPS as io1. Max ratio: 50 IOPS per GB for io1, 500 IOPS per GB for io2. Best for: OLTP, large databases, high-transaction applications.

■ *Use when you need >16,000 IOPS or consistent sub-millisecond latency.*

■ Magnetic (Standard)

Legacy HDD-based storage. Limited to 4 TB. Very low cost. NOT recommended for new deployments. Use only for infrequently accessed data or development with extremely tight budgets.

■ *Do NOT use for production. Use gp3 instead.*

Storage Autoscaling:

Enable Storage Autoscaling to have RDS automatically expand storage when free space drops below threshold (10% or 5 GB, whichever is larger). Set Maximum Storage Threshold to control costs. Expansion: increases by 10 GB or 10% (whichever is larger), max once per 6 hours. No downtime. Works with gp2, gp3, io1, io2.

■■ Storage autoscaling is ONE-WAY — you can only increase storage, never decrease. Plan your maximum storage threshold carefully.

6. RDS Backup & Recovery

■ Automated Backups

Daily full backup during your backup window + transaction logs every 5 minutes. Stored in S3 (not visible in your S3 console). Retention: 1–35 days. Enables PITR (Point-In-Time Recovery) to any second within retention period. FREE storage up to 100% of your provisioned DB storage size. Automatically deleted when retention period expires.

■ *Enable on day 1. Set retention to 7 days minimum, 35 days for critical DBs.*

■ Manual Snapshots

User-initiated snapshots stored in S3. Retained INDEFINITELY until you delete them. Full copy of DB instance. Not affected by DB deletion. Can be shared with other AWS accounts or copied to other regions. Basis for creating new DB instances or restoring.

■ *Take snapshot before major changes: engine upgrades, schema migrations.*

■ Point-In-Time Restore (PITR)

Restore to ANY second within your backup retention window. Creates a NEW DB instance (does not overwrite existing). Minimum 5-minute restore granularity (transaction log replay). Useful for: recovering from human error (accidental DELETE/UPDATE), data corruption.

■ *Test PITR quarterly to verify RTO meets your DR requirements.*

■ Cross-Region Backup Copy

Copy automated backups or snapshots to another region. Provides geographic redundancy for DR. Configure: RDS → Automated backups → Enable cross-region backup replication. Cost: snapshot storage in destination region + cross-region data transfer.

■ *Required for multi-region DR strategy. Copy to at least one other region.*

RTO and RPO Targets by Architecture:

Architecture	RPO	RTO	Cost
Single-AZ + Automated Backup	Up to 5 min	Minutes–Hours	Lowest
Multi-AZ Deployment	0 (sync)	60–120 seconds	Medium
Multi-AZ + Read Replica	0 (sync)	60–120 seconds	Medium+
Aurora Multi-AZ	0 (6 copies)	<30 seconds	Higher

Aurora Global Database	<1 second	<1 minute	Highest

7. RDS Security

■ Network Isolation (VPC)

Always deploy RDS in private subnets — NEVER public. DB subnet group spans multiple AZs. Security Group: allow only from app tier SG on DB port. Never open 0.0.0.0/0 or use 'Publicly Accessible: Yes' for production.

■ Encryption at Rest

Enable KMS encryption at launch — cannot add later! Encrypts: DB storage, automated backups, read replicas, snapshots. Uses AWS KMS Customer Managed Keys (CMK) or AWS-managed keys. Transparent to applications — no code changes needed.

■ Encryption in Transit (SSL/TLS)

All DB engines support TLS connections. Enforce TLS: set require_secure_transport=ON (MySQL) or ssl=ON parameter. Download RDS CA certificate bundle and configure your app/JDBC/driver to verify certificate.

■ IAM Database Authentication

MySQL and PostgreSQL only. Authenticate with IAM token instead of password. Token valid for 15 minutes. Generated with aws rds generate-db-auth-token. Benefits: No password in connection string, centralized access control, audit via CloudTrail.

■ Secrets Manager Integration

Store DB master credentials in Secrets Manager. Auto-rotation every N days without application downtime. Applications fetch credentials at runtime — no hardcoded passwords. Zero-downtime rotation: creates new password, tests it, then rotates.

■ Security Groups & NACLs

Security Group: stateful, applies to RDS instance. Allow ONLY from app server security group on specific DB port. NACL: stateless, subnet-level, adds extra layer but manage carefully (return traffic rules).

■ CloudTrail & Enhanced Monitoring

CloudTrail logs all RDS API calls (create, modify, delete). Enhanced Monitoring: OS-level metrics (CPU, memory, disk I/O) at 1-second granularity. Performance Insights: SQL-level wait events, identify slow queries.

IAM DB Authentication — Code Example:

```
# Generate IAM auth token (valid 15 minutes) TOKEN=$(aws rds generate-db-auth-token \
--hostname mydb.cluster-xxxx.us-east-1.rds.amazonaws.com \ --port 3306 \ --region us-east-1 \
--username iam_user) # Connect using token as password mysql -h
mydb.cluster-xxxx.rds.amazonaws.com \ -u iam_user \ --password=$TOKEN \
--ssl-ca=rds-ca-2019-root.pem \ --enable-cleartext-plugin
```

8. RDS Pricing & Costing

RDS pricing has 5 cost dimensions. Always calculate all components for accurate estimates. Use the AWS Pricing Calculator for your specific workload.

- **1. DB Instance Hours:** Per-hour (billed per second) for running instance. Varies by instance class, engine, and Multi-AZ.
- **2. Storage:** Per GB-month for provisioned storage. gp3: $0.115/GB-mo. io1: $0.125/GB-mo + $0.10/IOPS-mo.
- **3. I/O Requests:** Only for Magnetic storage: $0.10 per million I/O requests. gp2/gp3/io1 have no per-I/O charge.
- **4. Backup Storage:** Free up to 100% of provisioned DB size. Beyond that: $0.095/GB-month.
- **5. Data Transfer:** Inbound: free. Same-AZ: free. Cross-AZ: $0.01/GB. Cross-region: $0.02–$0.09/GB. Internet: standard rates.

■ Real Cost Examples (us-east-1, MySQL, 2024):

Instance	Engine	Multi-AZ	On-Demand/hr	Monthly	1-yr RI/hr	Monthly RI
db.t3.micro	MySQL	No	$0.017	$12.24	$0.011	$7.92
db.t3.medium	MySQL	No	$0.068	$48.96	$0.044	$31.68
db.t3.medium	MySQL	Yes	$0.136	$97.92	$0.088	$63.36
db.m5.large	MySQL	No	$0.171	$123.12	$0.108	$77.76
db.m5.large	MySQL	Yes	$0.342	$246.24	$0.216	$155.52
db.m5.xlarge	MySQL	Yes	$0.684	$492.48	$0.432	$311.04
db.r6g.large	Aurora	Yes	$0.260	$187.20	$0.164	$118.08
db.r6g.xlarge	Aurora	Yes	$0.520	$374.40	$0.328	$236.16
db.r6g.4xlarge	Aurora	Yes	$2.080	$1497.60	$1.312	$944.64

Cost Optimization Strategies:

- Use Reserved Instances (1-yr All Upfront): save up to 40% vs On-Demand.
- Use gp3 instead of gp2: 20% cheaper, better IOPS baseline.
- Right-size with Performance Insights: identify over-provisioned instances.
- Use Aurora Serverless v2 for dev/test: pay per second, near-zero when idle.
- Stop RDS instances during off-hours: save up to 70% for dev environments (7PM–9AM).
- Delete manual snapshots regularly: automated backups are free up to DB size.
- Use cross-AZ read replicas for read scaling instead of larger instance type.
- Graviton (db.r6g, db.m6g) instances: 20% cheaper than Intel with better performance.

9. How to Create RDS — GUI Step-by-Step

Complete step-by-step walkthrough for launching an Aurora MySQL Cluster using the AWS Console.

STEP 1

Open RDS Console

AWS Console → Search 'RDS' → RDS Dashboard → Click 'Create database' (orange button).

STEP 2

Choose a database creation method

Select 'Standard create' (full control) → NOT Easy create (hides options). Easy create has fewer options — always use Standard for production.

STEP 3

Select Engine

Engine type: Amazon Aurora → Edition: Aurora (MySQL Compatible) → Engine version: Aurora MySQL 3.x (MySQL 8.0 compatible) → Templates: Production (enables Multi-AZ by default).

STEP 4

Settings

DB cluster identifier: prod-aurora-mysql-01 → Master username: admin (or custom) → Credentials management: AWS Secrets Manager (RECOMMENDED — auto-rotation) → OR set Master password manually → Confirm password.

STEP 5

Instance Configuration

DB instance class: Serverless v2 (variable) OR db.r6g.large (fixed) → For prod: db.r6g.large (Memory Optimized, Graviton, cost-effective) → Multi-AZ deployment: Create an Aurora Replica in a different AZ ■

STEP 6

Connectivity

VPC: select your production VPC (NOT default) → DB subnet group: select your private subnet group → Public access: No (NEVER for production!) → VPC Security Group: Create new → Name: rds-aurora-sg → Add inbound rule: MySQL/Aurora (3306) → Source: your app server security group ID → Availability Zone: No preference (AWS chooses for Multi-AZ).

STEP 7

Database Authentication

Select: Password and IAM database authentication → This allows both traditional password AND IAM token authentication.

STEP 8

Additional Configuration

Initial database name: appdb → DB cluster parameter group: default (or custom if you have one) → Backup retention: 7 days (or 35 for critical) → Backup window: 03:00–04:00 UTC (low traffic) → Enable Enhanced Monitoring: 60 second granularity → Enable Performance Insights: 7 days retention (free) → Enable auto minor version upgrade: Yes → Maintenance window: Sunday 05:00–06:00 UTC → Deletion protection: Enable ■ (prevents accidental deletion).

STEP 9

Encryption

Enable Encryption ■ → AWS KMS key: aws/rds (AWS managed) or select your CMK → NOTE: Cannot enable encryption after creation — must be set at launch!

STEP 10

Review & Create

Review Estimated monthly costs panel on the right → Click 'Create database' → Status: Creating (5–10 minutes) → Available → Note the Cluster endpoint (writer) and Reader endpoint.

STEP 11

Connect to Database

Get endpoint from RDS Console → Connectivity & security tab → Writer endpoint: prod-aurora-mysql-01.cluster-xxxx.us-east-1.rds.amazonaws.com → From app server: mysql -h -u admin -p → Test: SELECT @@aurora_version; → Connection from Lambda: use Lambda VPC config + RDS Proxy for connection pooling.

10. How RDS Works — Under the Hood

Multi-AZ Failover — Exact Sequence:

```
Normal operation: App → Route 53 CNAME → Primary DB (AZ-1) ←synchronous→ Standby (AZ-2)
Failure detected (any of): - Primary host failure - AZ outage - DB instance crash - EBS
storage failure - Network connectivity lost Failover sequence (60–120 seconds total): 1. RDS
detects failure (health checks every 10 seconds) 2. RDS promotes standby to new primary 3.
Route 53 CNAME updated to point to new primary 4. TTL expires (typically 30 seconds) 5.
Application reconnects to same endpoint → hits new primary 6. Old primary (now recovered)
becomes new standby App behavior: connection errors during failover Solution: Implement retry
logic with exponential backoff in app code!
```

Read Replica Replication Flow:

```
Asynchronous replication (not synchronous like Multi-AZ): Primary DB → Binary Log (binlog) →
ASYNC → Read Replica ↑ Replica lag: typically milliseconds, can grow under high write load
Endpoints: Writer endpoint → always primary → use for INSERT/UPDATE/DELETE Reader endpoint →
load balances across ALL read replicas → use for SELECT Instance endpoint → specific replica
→ use for testing Aurora vs RDS Read Replicas: RDS: replicate via binlog, up to 5 replicas,
replication lag Aurora: shared storage, up to 15 replicas, near-zero lag, faster failover
```

RDS Proxy — Connection Pooling:

RDS Proxy sits between your application and RDS. It pools and reuses database connections, reducing the overhead of opening new connections — critical for Lambda functions and microservices that open many short-lived connections.

- Reduces DB load from connection overhead by up to 66%.
- Automatic failover: proxy maintains connections during Multi-AZ failover, reducing failover time by 66%.
- IAM authentication at proxy layer — no credentials in application code.
- Scales with Lambda: Lambda can open 1000s of concurrent connections without overwhelming RDS.
- Supports MySQL, PostgreSQL, and Aurora engines.

11. Benefits & Features

Fully Managed
AWS handles OS patching, DB upgrades, backups, monitoring, hardware. Your DBA team focuses on schema and optimization, not plumbing.

High Availability
Multi-AZ with synchronous replication and automatic failover. 99.95% SLA. Aurora: 6 copies across 3 AZs, sub-30s failover.

Read Scaling
Up to 5 read replicas (15 for Aurora). Direct read traffic to replicas for reporting and analytics without impacting primary.

Auto Scaling
Storage autoscaling expands disk automatically. Aurora Serverless v2 scales compute. RDS can scale instance class during maintenance.

Automated Backup
Daily backups + transaction logs enable PITR to any second in retention window. Snapshots stored indefinitely. Cross-region copy.

Enterprise Security
VPC isolation, SG firewall, KMS encryption at rest, TLS in transit, IAM auth, Secrets Manager rotation, CloudTrail audit.

Observability
CloudWatch metrics, Performance Insights (SQL wait events), Enhanced Monitoring (OS-level), slow query logs, error logs.

Multi-Region
Cross-region read replicas. Aurora Global Database: RPO <1s, RTO <1min. Disaster recovery across continents.

Ecosystem Integration
Native VPC, IAM, CloudWatch, Secrets Manager, Lambda, ECS, Elastic Beanstalk integration. Works with any SQL client.

Aurora Performance
5x MySQL, 3x PostgreSQL speed. Parallel Query for analytics. Up to millions of IOPS on Aurora storage layer.

Data Durability
Aurora: 6 copies across 3 AZs. Self-healing storage corrects errors. 99.999999999% (11 nines) storage durability.

Migration Tools
AWS DMS (Database Migration Service) for online migration with minimal downtime. Schema Conversion Tool for engine changes.

12. Enterprise Architecture Diagram

```
PRODUCTION 3-TIER AURORA ARCHITECTURE ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
INTERNET | [Route 53] → [CloudFront CDN] | [Application Load Balancer] (Public Subnets AZ-1,
AZ-2) | ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ PRIVATE SUBNET
— App Tier (Auto Scaling Group) ■ ■ [EC2/ECS App AZ-1] [EC2/ECS App AZ-2] ■ ■ IAM Role:
SecretsManagerReadWrite ■ ■ Fetches DB credentials from Secrets Manager ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ | [RDS PROXY] ←
Connection pooling IAM Auth, Secrets Manager integration |
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ PRIVATE SUBNET — Data
Tier ■ ■ ■ ■ AURORA MYSQL CLUSTER ■ ■ Writer Endpoint → [Primary Instance AZ-1] ■ ■ ↕
synchronous storage ■ ■ Reader Endpoint → [Replica Instance AZ-2] ■ ■ ↕ synchronous storage
■ ■ ■ ■ AURORA STORAGE (6 copies across 3 AZs): ■ ■ [Node1,2 AZ-1] [Node3,4 AZ-2] [Node5,6
AZ-3] ■ ■ ■ ■ Backup: S3 (automated daily + PITR 35 days) ■ ■ Snapshot: Cross-region copy to
us-west-2 ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ SECURITY
CONTROLS: SG: App SG → RDS SG (port 3306 only) Encryption: KMS CMK on all EBS/Aurora storage
SSL: require_secure_transport=ON enforced Secrets Manager: auto-rotation every 30 days
MONITORING: CloudWatch → RDS Metrics → Alarm → SNS → PagerDuty Performance Insights → Slow
query dashboard Enhanced Monitoring → OS-level metrics (60s)
```

Aurora Global Database Architecture:

```
PRIMARY REGION (us-east-1) SECONDARY REGION (eu-west-1) ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ ■ Aurora Cluster ■ ■ Aurora Cluster ■ ■ Primary
(Write + Read) ■ ■■■■■■■→ ■ Read-Only Replicas (0–15) ■ ■ Aurora Storage: 6 AZs ■ <1sec ■
Aurora Storage: 6 AZs ■ ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ RPO
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■ Failover: promote secondary to primary in <1 minute Use
case: multi-region reads, disaster recovery, data sovereignty
```

13. Proof of Concept (POC) — Aurora MySQL + Python App

Phase 1: Create Aurora Cluster via AWS CLI

```
# Create DB subnet group aws rds create-db-subnet-group \ --db-subnet-group-name
poc-aurora-subnet-group \ --db-subnet-group-description 'Aurora POC Subnet Group' \
--subnet-ids subnet-1234abcd subnet-5678efgh # Create Aurora cluster aws rds create-db-cluster
\ --db-cluster-identifier poc-aurora-cluster \ --engine aurora-mysql \ --engine-version
8.0.mysql_aurora.3.04.0 \ --master-username admin \ --master-user-password 'MySecurePass123!'
\ --db-subnet-group-name poc-aurora-subnet-group \ --vpc-security-group-ids sg-xxxxxxxx \
--storage-encrypted \ --backup-retention-period 7 \ --database-name pocdb # Add writer
instance aws rds create-db-instance \ --db-instance-identifier poc-aurora-writer \
--db-cluster-identifier poc-aurora-cluster \ --engine aurora-mysql \ --db-instance-class
db.r6g.large # Add reader instance aws rds create-db-instance \ --db-instance-identifier
poc-aurora-reader \ --db-cluster-identifier poc-aurora-cluster \ --engine aurora-mysql \
--db-instance-class db.r6g.large
```

Phase 2: Python Application with Connection Pooling

```
import boto3 import pymysql import json from sqlalchemy import create_engine from
sqlalchemy.pool import QueuePool def get_db_credentials(): '''Fetch credentials from Secrets
Manager''' client = boto3.client('secretsmanager', region_name='us-east-1') secret =
client.get_secret_value(SecretId='poc/aurora/master') return
json.loads(secret['SecretString']) def create_db_engine(): creds = get_db_credentials() engine
= create_engine( f"mysql+pymysql://{creds['username']}:{creds['password']}"
f"@{creds['host']}:{creds['port']}/{creds['dbname']}", poolclass=QueuePool, pool_size=10, #
Base connections max_overflow=20, # Extra connections under load pool_pre_ping=True, # Detect
stale connections (Multi-AZ failover) pool_recycle=3600, # Recycle connections every hour
connect_args={'ssl': {'ca': '/etc/ssl/certs/rds-ca-2019-root.pem'}} ) return engine # Writer
connection (INSERT/UPDATE/DELETE) WRITER_HOST =
'poc-aurora-cluster.cluster-xxxx.us-east-1.rds.amazonaws.com' # Reader connection (SELECT
queries) READER_HOST = 'poc-aurora-cluster.cluster-ro-xxxx.us-east-1.rds.amazonaws.com' #
Always use retry logic! from tenacity import retry, stop_after_attempt, wait_exponential
@retry(stop=stop_after_attempt(3), wait=wait_exponential(min=1, max=8)) def
execute_with_retry(engine, query, params=None): with engine.connect() as conn: result =
conn.execute(query, params or {}) return result.fetchall()
```

Phase 3: Validation Checklist

- ■ Writer endpoint resolves and accepts connections on port 3306
- ■ Reader endpoint load balances across both instances
- ■ Multi-AZ failover: stop primary → verify app reconnects within 2 minutes
- ■ PITR: delete test table → restore to 5 minutes ago → verify data
- ■ Backup appears in RDS Console → Automated backups tab
- ■ Performance Insights shows query wait events
- ■ CloudWatch alarm fires when CPU > 80%
- ■ Secrets Manager rotation: credentials updated without app downtime
- ■ Encryption: confirm StorageEncrypted = true in cluster details

- ■ SSL: verify connection uses TLS (show status in MySQL)

14. Standard Operating Procedures (SOP)

SOP-RDS-001: Create Production RDS/Aurora Cluster

```
Pre-requisites:
- VPC with private subnets in 2+ AZs
- DB subnet group created
- Custom parameter group prepared
- KMS CMK key available
- Secrets Manager secret template ready

Steps:
1. Choose engine: Aurora MySQL/PostgreSQL preferred for new workloads
2. Enable Multi-AZ (mandatory for production)
3. Enable storage encryption with KMS CMK (cannot add later!)
4. Set backup retention: minimum 7 days, 35 for critical
5. Deploy in private subnets ONLY (Publicly accessible: No)
6. Configure SG: allow only from app tier SG on DB port
7. Enable Performance Insights and Enhanced Monitoring
8. Enable deletion protection
9. Store master credentials in Secrets Manager with auto-rotation
10. Tag: Env=Production, Team=Backend, CostCenter=1234
```

SOP-RDS-002: RDS Instance Resize (Vertical Scaling)

```
1. NEVER resize without maintenance window during business hours
2. Create manual snapshot BEFORE resize as safety net
3. For Multi-AZ: resize applies to standby first, then failover, then primary
→ For RDS: downtime during final failover (60–120s)
→ For Aurora: zero downtime (replicas updated one at a time)
4. Select: Apply during next scheduled maintenance window (safer)
OR Apply immediately if urgent
5. Monitor: CloudWatch CPU/Memory after resize
6. Verify: application performance improvement
7. Update CMDB with new instance class
8. Adjust CloudWatch alarms if needed (e.g., memory threshold)
```

■ SOP-RDS-003: Backup Verification and PITR Test

```
Monthly procedure:
1. Verify automated backups exist:
aws rds describe-db-cluster-snapshots --db-cluster-identifier prod-aurora
2. Verify backup retention period is configured correctly
3. Perform PITR test to non-production environment:
aws rds restore-db-cluster-to-point-in-time \
--source-db-cluster-identifier prod-aurora \
--db-cluster-identifier prod-aurora-pitr-test \
--restore-to-time 2024-01-15T10:30:00Z
4. Connect to restored cluster and verify data integrity
5. Measure restore time (documents actual RTO)
6. Delete test cluster after validation
7. Document: restore time, data completeness, issues found
8. Report to CISO/team: DR readiness confirmed
```

■ SOP-RDS-004: Multi-AZ Failover Test

```
Quarterly DR test:
1. Notify all stakeholders of planned failover test
2. Start time: during low-traffic window
3. Record current primary AZ (RDS Console → Connectivity & Security)
4. Enable enhanced logging to capture failover timeline
5. Initiate failover:
aws rds failover-db-cluster --db-cluster-identifier prod-aurora
6. Measure: time from failover initiation to application recovery
7. Verify: new primary in different AZ
8. Check: application error logs for connection errors during failover
9. Verify: retry logic worked correctly
10. Document: actual RTO achieved vs target RTO
```

■ SOP-RDS-005: Security Incident — Suspected DB Compromise

```
IMMEDIATE (within 30 minutes):
1. Rotate master credentials immediately via Secrets Manager
aws secretsmanager rotate-secret --secret-id prod/aurora/master
2. Change all app user passwords
3. Review Security Groups — remove any unexpected rules
4. Enable enhanced logging if not already active

INVESTIGATION:
5. Review CloudTrail for unusual RDS API calls (modifications, snapshots)
6. Review DB audit logs for unusual queries (large exports, new users)
7. Check CloudWatch for unusual connection counts or query volumes
8. Review VPC Flow Logs for unexpected traffic to DB port

REMEDIATION:
9. Snapshot current DB (forensic evidence)
10. Restore from known-good snapshot to new cluster if compromised
11. File security incident report within 72 hours
12. Notify DPA if PII data potentially exposed (GDPR compliance)
```

15. GitHub README (Professional Format)

~~~
# ■■ AWS Aurora Production Deployment

![AWS](https://img.shields.io/badge/AWS-Aurora-teal?logo=amazonaws)
![License](https://img.shields.io/badge/license-MIT-blue)
![Terraform](https://img.shields.io/badge/IaC-Terraform-purple)

Production-grade Aurora MySQL cluster with Multi-AZ, RDS Proxy, Secrets Manager,
and full observability stack.

## ■ Prerequisites
- AWS CLI configured
- Terraform >= 1.5.0
- VPC with private subnets in 2+ AZs
- KMS key for encryption

## ■■ Architecture
App Tier → RDS Proxy → Aurora Writer + Aurora Reader
Multi-AZ: 6 storage copies across 3 AZs
Backup: Automated (35 days) + Cross-region snapshots

## ■ Quick Start
```bash
git clone https://github.com/your-org/aws-aurora-prod.git
cd aws-aurora-prod
cp terraform.tfvars.example terraform.tfvars
# Edit: cluster_identifier, engine_version, instance_class
terraform init && terraform plan && terraform apply
```

## ■■ Configuration Variables
| Variable            | Default          | Description              |
|---------------------|------------------|--------------------------|
| engine              | aurora-mysql     | Aurora engine type       |
| engine_version      | 8.0.mysql_aurora | Aurora version           |
| instance_class      | db.r6g.large     | Instance class           |
| backup_retention    | 35               | Backup retention days    |
| multi_az            | true             | Enable Multi-AZ          |

## ■ Security
- Private subnets only (no public access)
- KMS encryption at rest
- TLS enforced in transit
- Secrets Manager auto-rotation (30 days)
- RDS Proxy for IAM authentication
- Deletion protection enabled

## ■ Cost Estimate (us-east-1)
- 2x db.r6g.large (Multi-AZ): ~$374/month
- Storage 100 GB gp3: ~$11.50/month
- RDS Proxy: ~$0.015/vCPU-hr
- Backup storage (35 days): included up to 100 GB
- Estimated total: ~$400/month
~~~

```
## ■ License
MIT License
```

16. Top 25 RDS Interview Q&A;

Q1: What is Amazon RDS?	Fully managed relational database service. AWS handles hardware, OS, patching, backups, failover. You manage schema and queries.
Q2: RDS vs EC2 database?	RDS: managed (backups, HA, patching auto). EC2: self-managed (full control, custom configs, cheaper for large scale). Use RDS unless you need RAC or unsupported engine.
Q3: What engines does RDS support?	MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora. Aurora is the preferred choice for cloud-native workloads.
Q4: What is Multi-AZ?	Synchronous standby replica in different AZ. Automatic failover in 60–120 seconds. Standby is NOT readable. Increases availability and data durability.
Q5: Multi-AZ vs Read Replica?	Multi-AZ: sync replication, HA/DR, standby not readable, auto-failover. Read Replica: async replication, read scaling, readable, in same or different region.
Q6: What is Aurora?	AWS cloud-native DB. MySQL/PostgreSQL compatible. 6 copies across 3 AZs. 5x MySQL speed. Storage auto-grows to 128 TB. Sub-30s failover. Up to 15 replicas.
Q7: Aurora vs RDS MySQL?	Aurora: faster, better HA, more replicas, shared storage, backtrack, serverless. RDS MySQL: cheaper for small DBs, exact MySQL compatibility. Aurora costs ~20% more.
Q8: What is Aurora Serverless?	Auto-scales compute based on load. Pay per second. Scales from 0.5 to 128 ACUs. Ideal for variable/unpredictable workloads and dev/test environments.
Q9: What is PITR?	Point-In-Time Recovery. Restore DB to any second within retention period using transaction logs. Creates NEW DB instance. Retention: 1–35 days.
Q10: How to encrypt RDS?	Enable at creation — cannot add encryption later! Uses KMS. Encrypts storage, backups, replicas, snapshots. Transparent to applications.
Q11: What is RDS Proxy?	Connection pooler between app and RDS. Reduces connection overhead, improves failover time by 66%, supports IAM auth. Essential for Lambda + RDS.

Q12: What storage types does RDS support?	gp3 (default, 3000 baseline IOPS), gp2 (legacy, burst-based), io1/io2 (high IOPS for intensive workloads), Magnetic (legacy HDD, avoid).
Q13: What is Aurora Global Database?	One primary region + up to 5 read-only secondary regions. Cross-region replication lag <1 second. RPO <1s, RTO <1 min. For multi-region DR and low-latency global reads.
Q14: What is Backtrack?	Aurora MySQL only. Rewind DB to any point in time without restoring from backup. Up to 72 hours back. Fast — no restore required. For human error recovery.
Q15: How does Aurora storage work?	Distributed storage layer across 6 nodes in 3 AZs. Only redo logs written (not full pages). Quorum: 4/6 writes, 3/6 reads. Auto-grows in 10 GB increments.
Q16: What is DB parameter group?	Configuration file for DB engine. Controls settings like max_connections, buffer pool size, log settings. Default groups are read-only — create custom ones.
Q17: How to secure RDS?	Private subnets (no public access), Security Groups restrict to app tier, KMS encryption, TLS enforced, IAM auth, Secrets Manager for credentials, CloudTrail audit.
Q18: What is RDS storage autoscaling?	Auto-increases storage when free space drops below 10% or 5 GB. One-way (can only increase). Set Maximum Storage Threshold to control costs. No downtime.
Q19: Cross-region Read Replicas?	Async replication to different region. Readable. Can be promoted to standalone. Used for DR, regional reads, migration. Data transfer cost applies.
Q20: What is the maintenance window?	Weekly 30-min window for OS/DB patching. Choose low-traffic time. Multi-AZ: standby patched first → failover → primary patched (near-zero downtime).
Q21: RDS for Oracle vs self-managed?	RDS Oracle: managed backups, Multi-AZ, patching. Doesn't support RAC (use RDS Custom). BYOL or License Included pricing. Self-managed: full control including RAC.
Q22: What is IAM DB Authentication?	Authenticate to MySQL/PostgreSQL using IAM token (15 min validity) instead of password. No password in connection string. Centralized access control via IAM.
Q23: What causes RDS failover?	Primary host failure, AZ outage, DB crash, EBS failure, network loss, user-initiated failover, or OS/DB patching (Multi-AZ). RDS detects within 10–30 seconds.

Q24: How to migrate to RDS?	AWS DMS (Database Migration Service) for online migration with minimal downtime. AWS Schema Conversion Tool (SCT) for engine changes (Oracle to PostgreSQL).
Q25: RDS best practices summary?	Aurora for prod, Multi-AZ always, private subnets, encryption at launch, Secrets Manager, gp3 storage, Performance Insights, retry logic in apps, deletion protection.

■■ YOU ARE NOW RDS & AURORA CERTIFIED READY ■■