Describe your experience with RDS in a production environment ?

In production, I've used Amazon RDS — mainly MySQL and PostgreSQL — as the main database for our applications. I usually provision it using **Terraform**, so the database setup is fully automated and version-controlled. RDS sits inside a **private VPC** with proper **security groups**, and for all production systems, we enable **Multi-AZ** for high availability.
In our **CI/CD pipeline**, the database part is handled carefully.
The RDS instance itself is not recreated in every build — it's managed separately as infrastructure-as-code.
When developers add a new feature, sometimes they also need to **change the database** — for example, add a new column like phone_number in the users table.
Instead of doing that manually in the RDS console or with SQL commands,
we use a **migration tool** such as **Flyway** or **Liquibase**.

So whenever we deploy a new release, the pipeline first runs migrations in **staging**, runs automated tests, and only then applies them to **production**.
If a migration might be risky, we take a **blue/green approach** — clone the DB, apply changes, test, and then cut over to the new one.
For **high availability**, all production RDS databases run in **Multi-AZ mode**.
That means AWS automatically maintains a standby database in another availability zone with synchronous replication.
If the main DB goes down, failover happens automatically — usually within 30–60 seconds.
We've tested this using the **"Reboot with failover"** option during off-peak hours to make sure the app reconnects properly.
We also added **connection retry logic** and **pooling (PgBouncer or HikariCP)** in the app so users don't notice small failovers.
For **disaster recovery**, we set up **cross-region read replicas** — for example, our main DB was in Mumbai (ap-south-1) and a replica was in Virginia (us-east-1).
If the primary region ever fails, we can **promote the replica** and point the app to the new endpoint.
We also copy automated **snapshots** every night to another region as a backup.
Our DR plan is documented — it includes steps like promoting the replica, updating DNS, and running smoke tests to confirm recovery.
We even do **quarterly DR drills** to practice this and make sure our RTO and RPO targets are met.

So in short —
I manage RDS as **infrastructure-as-code**, integrate it safely into the **CI/CD pipeline** for schema changes, ensure **high availability** with Multi-AZ and failover testing, and maintain **disaster recovery** through cross-region replicas and snapshot backups.
That setup gives us both uptime and confidence that we can recover from failures quickly.

Which RDS configuration provides **automatic failover** to a standby instance in another Availability Zone?
A. Read Replica
B. Multi-AZ Deployment
C. Aurora Serverless
D. Parameter Group
✅ **Correct Answer: B. Multi-AZ Deployment**
💡 **Explanation:** Multi-AZ provides **synchronous replication** to a standby DB in another AZ. On failure, RDS automatically promotes the standby.

During an RDS failover, what happens to the database endpoint?
A. The endpoint IP changes, but the DNS name remains the same.
B. The endpoint DNS name changes permanently.
C. You must manually update application config.
D. Failover requires manual endpoint remapping.
✅ **Correct Answer: A.**
💡 **Explanation:** RDS updates the **DNS record** to point to the new primary. The DNS name stays constant, ensuring app transparency.

Which RDS feature should you use to replicate data across AWS **regions** for disaster recovery?
A. Multi-AZ deployment
B. Read Replica (Cross-Region)
C. Elastic Beanstalk
D. Parameter Group replication
✅ **Correct Answer: B. Cross-Region Read Replica**
💡 **Explanation:** Multi-AZ is intra-region; **Cross-region read replicas** enable asynchronous replication for regional DR setups.

How do you typically integrate RDS schema migrations into a CI/CD pipeline?
A. Manually apply scripts after deployment
B. Include migration tools like Flyway/Liquibase as a pipeline stage
C. Use AWS Lambda triggers for DB updates
D. Apply migrations via AWS Config
✅ **Correct Answer: B.**
💡 **Explanation:** Tools like **Flyway, Liquibase, or Alembic** are commonly integrated into CI/CD to apply schema changes automatically and safely.

Which Infrastructure-as-Code tool is most commonly used to **provision RDS** in automated pipelines?

A. Chef
B. Terraform
C. Puppet
D. AWS Inspector
✅ **Correct Answer: B. Terraform**
💡 **Explanation: Terraform** is widely used for managing RDS configurations — VPC, subnet group, security, Multi-AZ, backups — through IaC.

What is the **primary difference** between Multi-AZ and Read Replica in RDS?
A. Multi-AZ improves read scaling; Read Replica improves availability.
B. Multi-AZ is synchronous for HA; Read Replica is asynchronous for read scaling.
C. Both are synchronous replication methods.
D. Multi-AZ is for analytics; Read Replica is for DR.
✅ **Correct Answer: B.**
💡 **Explanation:** Multi-AZ → **synchronous**, for HA; Read Replica → **asynchronous**, for **read scaling or cross-region DR**.

In a production-grade CI/CD pipeline, how are **risky schema changes** (like column drops or type changes) typically handled?
A. Directly deployed to production.
B. Using blue/green or clone → migrate → cutover strategy.
C. Executed via Lambda.
D. Ignored during deployment.
✅ **Correct Answer: B.**
💡 **Explanation:** A **blue/green migration** isolates the schema change to a cloned DB, validated before final cutover — ensuring zero downtime and rollback safety.
