What strategies do you use for RDS cost optimization?

In our environment, RDS costs were getting high, so I started by checking how each database was being used. My main approach is to **match the DB type and size to the workload** — what we call *right-sizing*.
I monitor key CloudWatch metrics like **CPU utilization**, **memory**, and **connections**.
If a database is using very little CPU or memory, I downgrade it to a smaller instance.
If it's heavily used, I keep it at the right size or move it to a Reserved Instance.
For **production databases** that run all the time, I prefer **Reserved Instances** — they can save up to 60–70% compared to On-Demand.
For **testing or development databases**, I use smaller **burstable instances** like db.t3 and schedule them to automatically stop at night and on weekends. That alone can save 50% or more.
For **workloads that change a lot**, I use **Aurora Serverless**, which automatically scales up or down and charges only for what's used.
I also look at **storage costs**. Many teams over-provision storage, so I switch them to **gp3 volumes**, which are cheaper and allow adjustable IOPS. I also delete old manual snapshots and move old data to **S3 or Athena** instead of keeping it in RDS.
For **read-heavy workloads**, I use **read replicas** or add **ElastiCache (Redis)** to reduce load on the main database — this helps us avoid scaling to a bigger, more expensive instance.
Finally, I set up **CloudWatch dashboards** and use **Trusted Advisor** to identify underutilized databases and cost anomalies each month.
In one case, our staging RDS was running 24/7 on a large instance, even though it was only used in office hours. I set up a simple Lambda function to stop it at night and right-sized it to a smaller instance. That single change saved about 55% in monthly RDS costs.
So in short —
I optimize RDS by **right-sizing**, **using Reserved Instances for steady workloads**, **Serverless or On-Demand for variable ones, automating stop/start for non-prod, using caching and replicas**, and **keeping an eye on metrics through CloudWatch and Trusted Advisor**."

Which RDS pricing option is **best for a steady, predictable production workload** that runs 24/7?
A. On-Demand Instances
B. Reserved Instances

C. Spot Instances
D. Aurora Serverless
✅ **Correct Answer: B. Reserved Instances**

💡 **Explanation:** Reserved Instances provide up to **69% savings** for long-term, predictable workloads by committing for 1–3 years.

What's the most cost-efficient instance type for **development or testing environments** that don't run 24/7?
A. db.m6g.large
B. db.r6g.xlarge
C. db.t3.micro or db.t4g.micro (burstable)
D. db.z1d.2xlarge
✅ **Correct Answer: C. db.t3.micro or db.t4g.micro**
💡 **Explanation:** Burstable instances like t3 or t4g are **low-cost** and ideal for non-continuous or light workloads.

For a **spiky or unpredictable workload**, which RDS option gives **automatic scaling and pay-per-second billing**?
A. Reserved Instance
B. Aurora Serverless v2
C. Multi-AZ Deployment
D. Provisioned IOPS
✅ **Correct Answer: B. Aurora Serverless v2**
💡 **Explanation: Aurora Serverless v2** scales compute automatically and charges only for actual usage — ideal for bursty or unpredictable traffic.

Which of the following strategies directly reduces **storage costs** for RDS?
A. Use io2 volumes for all workloads
B. Disable snapshots
C. Move cold/archival data to S3 and query via Athena
D. Keep all snapshots forever
✅ **Correct Answer: C. Move cold/archival data to S3 and query via Athena**
💡 **Explanation:** Offloading old data from RDS to **S3 + Athena/Redshift Spectrum** reduces expensive RDS storage usage.

What should you do if CloudWatch shows **CPU <20% and memory barely used** for weeks?
A. Scale up instance
B. Enable Multi-AZ deployment
C. Right-size to a smaller instance
D. Increase read replicas
✅ **Correct Answer: C. Right-size to a smaller instance**
💡 **Explanation:** Persistent underutilization indicates over-provisioning — downsizing saves cost without hurting performance.

How can you automatically reduce RDS costs for **non-production environments**?
A. Enable deletion protection
B. Schedule stop/start using Lambda or EventBridge during off-hours
C. Create daily snapshots
D. Run read replicas for each region
✅ **Correct Answer: B. Schedule stop/start using Lambda or EventBridge**
💡 **Explanation:** Automating RDS **stop/start** outside office hours saves up to **70%** for dev/test databases.
