Choosing Right AWS Database

Questions to choose the right database based on your architecture:

1. Read-heavy, write-heavy, or balanced workload? Throughput needs? Will it change, does it needs to scale or fluctuate during the day?
2. How much data to store and for how long? Will it grow? Average object size? How are they accessed?
3. Data durability? Source of truth for the data?
4. Latency requirements? Concurrent users?
5. Data model? How will you query the data? Joins? Structured? Semi-structured?
6. Strong schema? More flexibility? Reporting? Search? RDBMS/NoSQL?
7. License costs? Switch to Cloud Native DB such as Aurora? [**Tip**: To estimate the cost for your architecture solution, refer AWS Pricing Calculator]

Database Types:

1. RDBMS (=SQL/OLTP):**RDS**, **Aurora** – great for joins
2. NoSQL database: **DynamoDB** (~JSON), **ElastiCache** (key / value pairs), **Neptune** (graphs) – no joins, no SQL
3. Object Store: **S3** (for big objects) / Glacier (for backups / archives)
4. Data Warehouse (= SQL Analytics / BI) : **Redshift** (OLAP), **Athena**
5. Search: **ElasticSearch** (JSON) – free text, unstructured searches
6. Graphs: **Neptune** – displays relationships between data

Sl. No.	Types of Database	Overview	Point to remember for Solutions Architect	Use Case
1	**RDS**	• Managed PostgreSQL / MySQL / MariaDB / Oracle / SQL Server • Must Provision an EC2 instance & EBS Volume type and size • Support for Read Replicas and Multi-AZ • Security through IAM, Security, Groups, KMS, SSL in transit • Backup / Snapshot / Point in time restore feature • Managed and scheduled maintenance • Monitoring through Cloudwatch	1. **Operations**: small downtime when failover happens, when maintenance happens, scaling in read replicas / ec2 instance / restore EBS implies manual intervention, application changes 2. **Security**: AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, authoring users in DB, using SSL 3. **Reliability**: Multi AZ feature, failover in case of failures 4. **Performance**: depends on EC2 instance type, EBS volume type, ability to add Read Replicas. Doesn't auto-scale 5. **Cost**: Pay per hour based on provisional EC2 and EBS.	Store relational datasets (RDBMS/OLTP), perform SQL queries, transactional inserts/update/delete is available.

2	Aurora	• Compatible API for PostgreSQL / MySQL • Data is held in 6 replicas, across 3 AZ – lot of durability • Auto healing capability • Multi AZ, Auto scaling Read Replicas • Read Replicas can be Global • Aurora database can be Global for DR or latency purposes • Auto scaling of storage from 10GB to 64TB • Define EC2 instance type for aurora instances • Same security / monitoring / maintenance features as RDS • "Aurora Serverless" option	1. **Operations**: less operations, auto scaling storage 2. **Security**: AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, authoring users in DB using SSL 3. **Reliability**: multi AZ, highly available, possibly more than RDS, Aurora Serverless option 4. **Performance**: 5x performance (according to AWS) due to architectural optimization. Up to 15 Read Replicas (only 5 for RDS) 5. **Cost**: Pay per hour based on EC2 and storage usage. Possibly lower costs compared to Enterprise grade databases such as Oracle	Same as RDS, but with less maintenance / more flexibility / more performance. ***For enterprise grade applications, Aurora is the best option to choose.**
3	ElastiCache	• Managed Redis / Memcached (similar offering as RDS, but for caches) • In-memory data store, sub-millisecond latency • Must proviso an EC2 instance type • Support for Clustering (Redis) and Multi AZ, Read Replicas (sharding) • Security through IAM, Security groups, KMS, Redis Auth • Backup / Snapshot / Point in time restore feature • Managed scheduled maintenance • Monitoring through CloudWatch	1. **Operations**: same as RDS 2. **Security**: AWS responsible for OS security, we are responsible for setting up KMS, security groups, IAM policies, users (Redis Auth), using SSL 3. **Reliability**: multi AZ, Clustering, Sharding 4. **Performance**: Sub-millisecond performance, in memory, read replicas for sharding, very popular cache option 5. **Cost**: Pay per hour based on EC2 and storage usage.	Key/Value store, Frequent reads, less writes, cache results for DB queries, store session data for websites, cannot use SQL.
4	DynamoDB	• DynamoDB is a pure cloud native technology, it's a serverless • AWS proprietary technology, managed NoSQL database • Serverless, provisioned capacity, auto scaling, on demand capacity (Nov 2018) – scales based on your load • Can replace ElastiCache as a key/value store (storing session data for example) • Highly available, multi AZ by default, Reads and Writes are decoupled, DAX for read cache • Reads can be eventually consistent or strongly consistent • Security, authentication and authorization is done through IAM • DynamoDB streams to integrate with AWS Lambda • Backup / Restore feature, Global Table feature • Monitoring through CloudWatch • Can only query on primary key, sort key, or indexes	1. **Operations**: no operations needed, auto scaling capability, serverless 2. **Security**: full security through IAM policies, KMS encryption, SSL in flight 3. **Reliability**: multi AZ, Backups 4. **Performance**: single digit millisecond performance, DAX for caching reads, performance doesn't degrade if your application scales 5. **Cost**: Pay per provisioned capacity and storage usage (no need to guess in advance any capacity – can use auto scaling).	Serverless applications development (small documents 100s KB), distributed serverless cache, doesn't have SQL query language available, has transactions capability from **NOV 2018**.
5	S3	S3 is a database, it's not a conventional database • S3 is a …key / value store for objects • Great for big objects, not so great for small objects • S3 doesn't replace RDS / DynamoDB • Serverless, scales infinitely, max object size is 5TB • Eventually consistency for overwrites and deletes	1. **Operations**: no operations needed 2. **Security**: IAM, Bucket Policies, ACL, Encryption (Server/Client), SSL 3. **Reliability**: 99.999999999% durability / 99.99% availability, multi AZ, CRR 4. **Performance**: scales to thousands of reads / writes per second, transfer acceleration / multi-part for big files	Static files, key value store for big files, website hosting.

		• Tiers: S3 Standard, S3 IA (Infrequent Access), S3 One Zone IA, Glacier for backups • Features: versioning, encryption, CRR (Cross Region Replication), etc… • Security: IAM, Bucket policies, ACL (Access Control List) • Encryption: SSE-S3, SSE-KMS, SSE-C, client side encryption, SSL in transit	5. **Cost**: pay per storage usage, network cost, request number	
6	Athena	It is not a database it terms but it holds the data but it does provide a query engine on top of S3. • Fully serverless database with SQL capabilities • Used to query data in S3 • Pay per query • Output results back to S3 • Secured through IAM	1. **Operations**: no operations needed, serverless 2. **Security**: IAM + S3 security 3. **Reliability**: managed service, used Presto* engine, highly available 4. **Performance**: queries scale based on data size 5. **Cost**: pay per query / per TB of data scanned, serverless	One time SQL queries, serverless queries on S3, log analytics.
7	Redshift	• Redshift is based on PostgreSQL, but it's not used for OLTP • It's OLAP – online analytical processing (analytics and data warehousing) • 10x better performance than other data warehouses, scale to PBs of data • Columnar storage of data (instead of row based) • Massively Parallel Query Execution (MPP), highly available • Pay as you go based on the instance provisioned • Has a SQL interface for performing the queries • BI tools such as AWS Quicksight or Tableau integrate with it • Data is loaded from S3, DynamoDB, DMS, other DBs… • From 1 node to 128 nodes, up to 160 GB of space per node • Leader node: for query planning, results aggregation • Compute node: for performing the queries, send results to leader • Redshift Spectrum: perform queries directly against S3 (no need to load) • Backup & Restore, Security VPC / IAM / KMS, Monitoring • Redshift Enhanced VPC Routing: COPY / UNLOAD goes through VPC	1. **Operations**: similar to RDS 2. **Security**: IAM, VPC, KMS, SSL (Similar to RDS) 3. **Reliability**: highly available, auto healing features 4. **Performance**: 10x performance vs other data warehousing, compression 5. **Cost**: pay per node provisioned, 1/10th of the cost vs other warehouses	**Remember**: Redshift = Analytics / BI/ Data Warehouse
8	Neptune	• Fully managed graph database • When do we use Graphs? - High relationship data - Social Networking: Users friends with Users, replied to comment on post of user and likes other comments - Knowledge graphs (Wikipedia) • Highly available across 3 AZ, with up to 15 read replicas	1. **Operations**: similar to RDS 2. **Security**: IAM, VPC, KMS, SSL (similar to RDS) + IAM Authentication 3. **Reliability**: Multi-AZ, clustering 4. **Performance**: best suited for graphs, clustering to improve performance 5. **Cost**: pay per node provisioned (similar to RDS)	**Remember**: Neptune = Graphs

		• Point-in-time recovery, continuous backup to Amazon S3 • Support for KMS encryption at rest + HTTPS		
9	ElasticSearch	• With ElasticSearch, you can search any field, even partially matches • It's common to use ElasticSearch as a complement to another database • ElasticSearch also has some usage for Big Data applications • You can provision a cluster of instances • Built-in integrations: Amazon Kinesis Data Firehose, AWS IoT, and Amazon CloudWatch Logs for data ingestion • Security through Cognito & IAM, KMS encryption, SSL & VPC • Comes with Kibana (visualization) & Logstash (log ingestion) – ELK stack	1. **Operations**: similar to RDS 2. **Security**: Cognito, IAM, VPC, KMS, SSL 3. **Reliability**: Multi-AZ, clustering 4. **Performance**: best on ElasticSearch project (open source), petabyte scale 5. **Cost**: pay per node provisioned (similar to RDS)	**Remember**: ElasticSearch = Search / Indexing

* **Presto** is a high performance, distributed SQL query engine for big data. Its architecture allows users to query a variety of data sources such as Hadoop, AWS S3, Alluxio, MySQL, Cassandra, Kafka, and MongoDB. One can even query data from multiple data sources within a single query