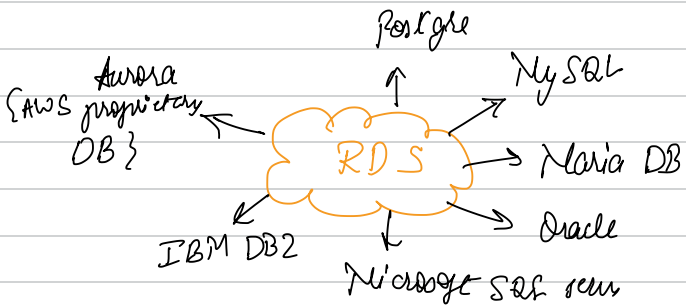
Database

RDS { Relational Database Service }

→ AWS manage DB service for DB use as a query language

→ no need to visit & provision servers to run the databases, all underlying softwares & patch installation done by AWS itself.

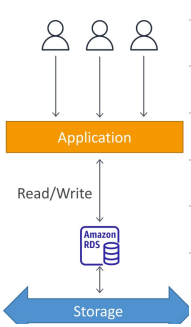


- RDS is a managed service:
  - Automated provisioning, OS patching
  - Continuous backups and restore to specific timestamp (Point in Time Restore)!
  - Monitoring dashboards
  - Read replicas for improved read performance
  - Multi AZ setup for DR (Disaster Recovery)
  - Maintenance windows for upgrades
  - Scaling capability (vertical and horizontal)
  - Storage backed by EBS (gp2 or io1)
- BUT you can't SSH into your instances

Don't have access to underlying EC2 instance

RDS – Storage Auto Scaling

- Helps you increase storage on your RDS DB instance dynamically
- When RDS detects you are running out of free database storage, it scales automatically
- Avoid manually scaling your database storage
- You have to set **Maximum Storage Threshold** (maximum limit for DB storage)
- Automatically modify storage if:
  - Free storage is less than 10% of allocated storage
  - Low-storage lasts at least 5 minutes
  - 6 hours have passed since last modification
- Useful for applications with **unpredictable workloads**
- Supports all RDS database engines



Read Replicas

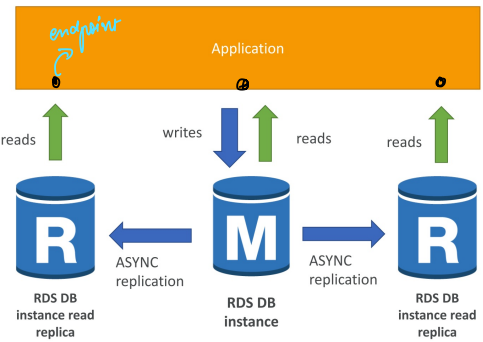
* each endpoint acts like a new DNS

RDS Read Replicas for read scalability

→ useful to distribute the read workloads using an asynchronous replication strategy!
→ creates endpoint for each replica hence we need to → update the connection string

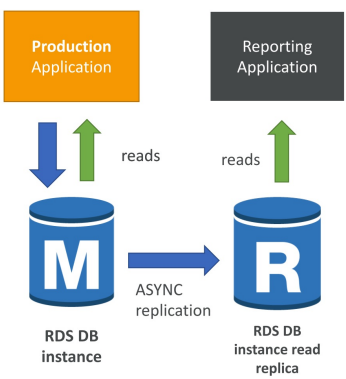
* need not do with multi AZ!

- Up to 15 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is ASYNC, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



Use Case!

- You have a production database that is taking on normal load
- You want to run a reporting application to run some analytics
- You create a Read Replica to run the new workload there
- The production application is unaffected
- Read replicas are used for SELECT (=read) only kind of statements (not INSERT, UPDATE, DELETE)



just so that we don't overload the main DB

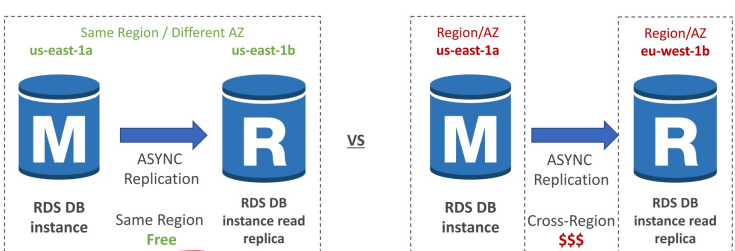
also security pov

i.e. only "select" statements!

RDS Read Replicas – Network Cost

* generally for managed service inter zone data transfer may have exception because in AWS moving data from 1 AZ to other costs!

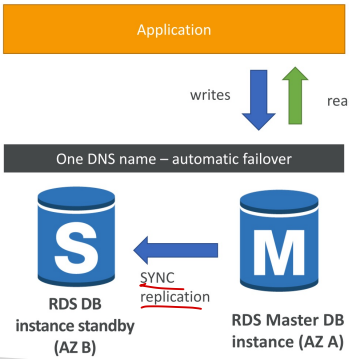
- In AWS there's a network cost when data goes from one AZ to another
- For RDS Read Replicas within the same region, you don't pay that fee



Multi AZ Setup!

RDS Multi AZ (Disaster Recovery)

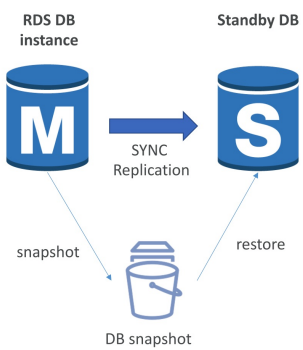
- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling
- Note: The Read Replicas be setup as Multi AZ for Disaster Recovery (DR)



→ Read replicas can be set as Multi AZ for disaster recovery but remember that read replicas use ASYNC replication so if the main DB is being written and suppose a failover occurs, then the older data in read replica will be used!

RDS – From Single-AZ to Multi-AZ

- Zero downtime operation (no need to stop the DB)
- Just click on "modify" for the database
- The following happens internally:
  - A snapshot is taken
  - A new DB is restored from the snapshot in a new AZ
  - Synchronization is established between the two databases



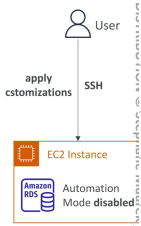
download for windows!

* create a database from console
→ use SQL electron to connect to the RDS database endpoint so that we can perform query operations on the DB!
→ to make read replicas create from action.
→ check other actions available!

RDS Custom

→ Allows us to manage the underlying database & OS which RDS automatically does!

- Managed Oracle and Microsoft SQL Server Database with OS and database customization
- RDS: Automates setup, operation, and scaling of database in AWS
- Custom: access to the underlying database and OS so you can
  - Configure settings
  - Install patches
  - Enable native features
  - Access the underlying EC2 Instance using SSH or SSM Session Manager
- De-activate Automation Mode to perform your customization, better to take a DB snapshot before
- RDS vs. RDS Custom
  - RDS: entire database and the OS to be managed by AWS
  - RDS Custom: full admin access to the underlying OS and the database



AWS Aurora { AWS proprietary }

→ AWS' Managed PostgreSQL & MySQL DB

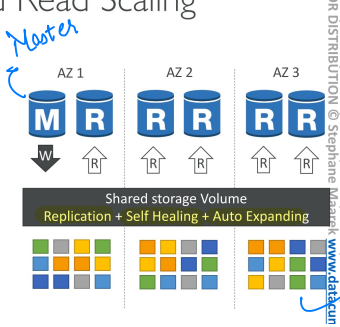
→ also has a serverless implementation

Amazon Aurora

- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is "AWS cloud optimized" and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 128 TB.
- Aurora can have up to 15 replicas and the replication process is faster than MySQL (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It's HA (High Availability) native.
- Aurora costs more than RDS (20% more) – but is more efficient

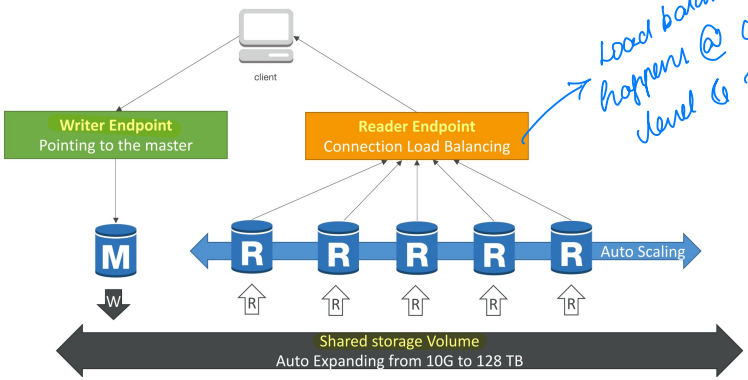
Aurora High Availability and Read Scaling

- 6 copies of your data across 3 AZ:
  - 4 copies out of 6 needed for writes
  - 3 copies out of 6 need for reads
  - Self healing with peer-to-peer replication
  - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds
- Master + up to 15 Aurora Read Replicas serve reads = 16
- Support for Cross Region Replication



data { each color has 6 copies }

Aurora DB Cluster



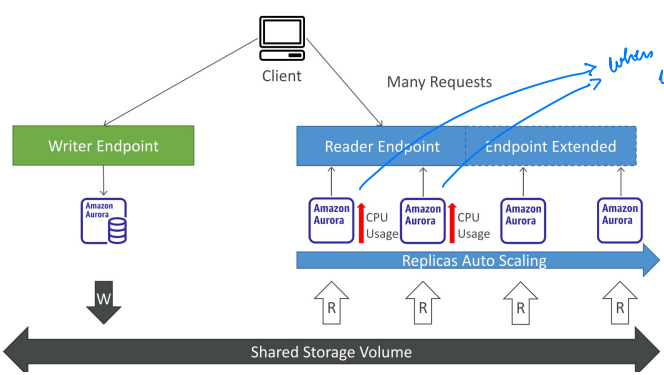
Load balancing happens @ connection level & not statement level

* connection level → physical hardware that links
* statement level → they execute only once for each single transaction

Features of Aurora

- Automatic fail-over
- Backup and Recovery
- Isolation and security
- Industry compliance
- Push-button scaling
- Automated Patching with Zero Downtime
- Advanced Monitoring
- Routine Maintenance
- Backtrack: restore data at any point of time without using backups

Aurora Replicas - Auto Scaling



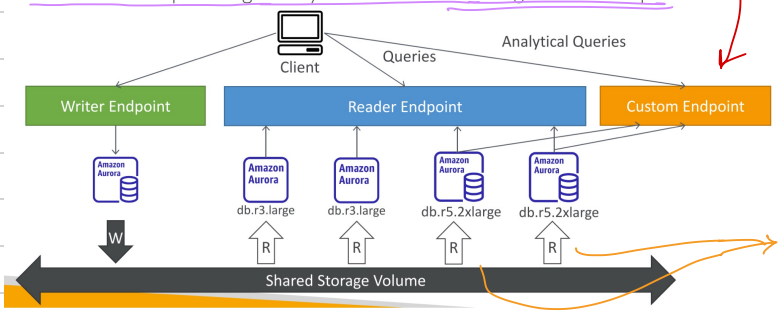
when CPU usage for these two increase the reader endpoint will create online read replicas as an action of the auto scaling feature!

Allows querying only a subset of read replicas

Aurora – Custom Endpoints

- Define a subset of Aurora Instances as a Custom Endpoint
- Example: Run analytical queries on specific replicas
- The Reader Endpoint is generally not used after defining Custom Endpoints

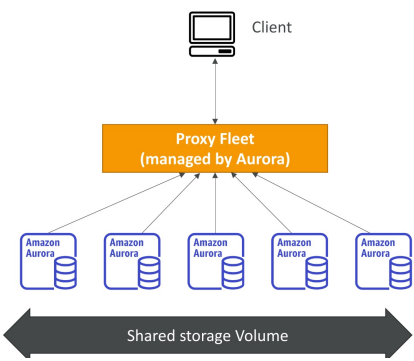
Instead we will be having different custom endpoints for different types of queries!



Larger read replicas

Aurora Serverless

- Automated database instantiation and auto-scaling based on actual usage
- Good for infrequent, intermittent or unpredictable workloads
- No capacity planning needed
- Pay per second, can be more cost-effective



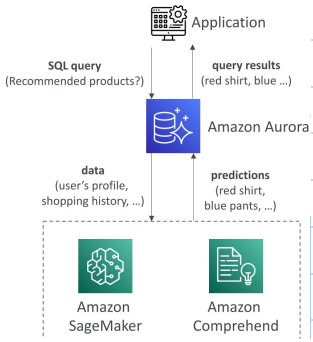
Global Aurora

- Aurora Cross Region Read Replicas:
  - Useful for disaster recovery
  - Simple to put in place
- Aurora Global Database (recommended):
  - 1 Primary Region (read / write)
  - Up to 5 secondary (read-only) regions, replication lag is less than 1 second
  - Up to 16 Read Replicas per secondary region
  - Helps for decreasing latency
  - Promoting another region (for disaster recovery) has an RTO of < 1 minute
  - Typical cross-region replication takes less than 1 second

RTO (Recovery time Objective)

Aurora Machine Learning

- Enables you to add ML-based predictions to your applications via SQL
- Simple, optimized, and secure integration between Aurora and AWS ML services
- Supported services
  - Amazon SageMaker (use with any ML model)
  - Amazon Comprehend (for sentiment analysis)
- You don't need to have ML experience
- Use cases: fraud detection, ads targeting, sentiment analysis, product recommendations



SQL query → what are the recommended products! take users data from Aurora & make predictions

RDS Backup

Automated backups!

* Daily backup of the DB
* Earliest backup of logs is 5 minutes in RDS → gives the ability to restore to a point in time (oldest backup to 5 mins ago.)
* We can set the automated backup retention period from 1 to 35 days! (set 0 to disable automated backup)

Manual DB snapshots

* User triggered & no retention period problems!

Trick → Suppose we need to use the RDS database for just 2 hrs in a month, instead of stopping the the DB (which will still cost money because storage is being used) we can take the snapshot of DB & delete the DB, whenever needed restore it!
* saving a snapshot costs much lesser than storing the entire DB!

Aurora Backups

Automated Backups

* 1 - 35 days retention { cannot be disabled }
* point in time recovery in that timeframe

Manual Backups

* same

Restoring RDS or Aurora

* restoration creates a new DB

Restoring MySQL RDS / Aurora cluster DB from S3

use Percona XtraBackup to create an Aurora cluster

1) Create a backup of the on-premises DB
2) Store it in an amazon S3 bucket
3) Restore the backup into a new RDS / Aurora cluster running mySQL.

Aurora Database Cloning

- Create a new Aurora DB Cluster from an existing one
- Faster than snapshot & restore
- Uses copy-on-write protocol
  1. Initially, the new DB cluster uses the same data volume as the original DB cluster (fast and efficient – no copying is needed)
  2. When updates are made to the new DB cluster data, then additional storage is allocated and data is copied to be separated
- Very fast & cost-effective
- Useful to create a "staging" database from a "production" database without impacting the production database

→ suppose we need to run some tests on the Aurora DB so we stage the DB away from the production env.

→ its a quick & cost effective feature to copy a database whereas replicas are read only copy of DB's.

RDS & Aurora Security

→ Postgre SQL & MySQL support IAM authentication but oracle does not

- At-rest encryption:
  - Database master & replicas encryption using AWS KMS – must be defined as launch time
  - If the master is not encrypted, the read replicas cannot be encrypted
  - To encrypt an un-encrypted database, go through a DB snapshot & restore as encrypted
- In-flight encryption: TLS-ready by default, use the AWS TLS root certificates client-side
- IAM Authentication: IAM roles to connect to your database (instead of username/pw)
- Security Groups: Control Network access to your RDS / Aurora DB
- No SSH available except on RDS Custom
- Audit Logs can be enabled and sent to CloudWatch Logs for longer retention

IAM
KMS {rest}
Security grp
SSL {in flight}

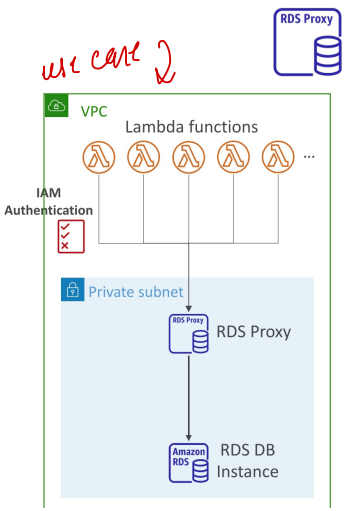
Amazon RDS Proxy

think of it like a load balancer for RDS

- Fully managed database proxy for RDS
- Allows apps to pool and share DB connections established with the database
- Improving database efficiency by reducing the stress on database resources (e.g., CPU, RAM) and minimize open connections (and timeouts)
- Serverless, autoscaling, highly available (multi-AZ)
- Reduced RDS & Aurora failover time by up 66%
- Supports RDS (MySQL, PostgreSQL, MariaDB, MS SQL Server) and Aurora (MySQL, PostgreSQL)
- No code changes required for most apps
- Enforce IAM Authentication for DB, and securely store credentials in AWS Secrets Manager
- RDS Proxy is never publicly accessible (must be accessed from VPC)

→ Reduces failovers time by 66%

use case



→ A func^n can multiply & they may have open connections or timeouts to the DB so connecting it to a proxy will prevent the overloading of DB instance. This may overload the proxy [but it is meant to be]

Amazon ElastiCache Overview

→ Results of queries are cached!

→ storing the user session make it stateless

- The same way RDS is to get managed Relational Databases…
- ElastiCache is to get managed Redis or Memcached
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Helps make your application stateless
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups
- Using ElastiCache involves heavy application code changes

→ unlike RDS proxy which does not require code changes

→ stateless application

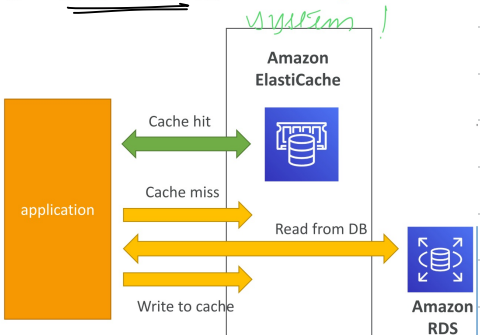
That each request as independent transaction that can be handled by any available servers.

→ it doesn't save client data in one session for use in next session.

ElastiCache Solution Architecture - DB Cache

First how the cache works in every other system!

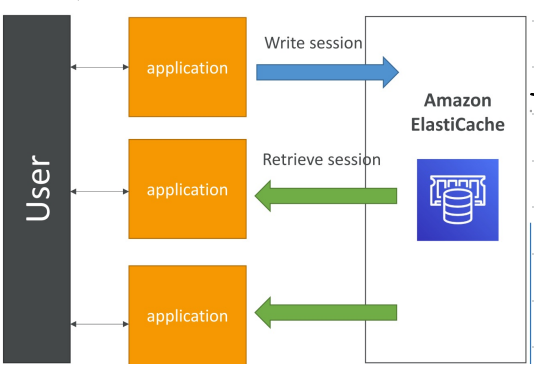
- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.



ElastiCache Solution Architecture – User Session Store

(Making the application stateless)

- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



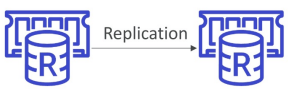
→ For session restoration!

→ stateless because user data is not stored in the EC2 servers.

ElastiCache – Redis vs Memcached

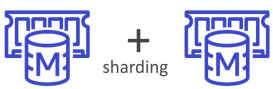
REDIS

- Multi AZ with Auto-Failover
- Read Replicas to scale reads and have high availability
- Data Durability using AOF persistence
- Backup and restore features
- Supports Sets and Sorted Sets



MEMCACHED

- Multi-node for partitioning of data (sharding)
- No high availability (replication)
- Non persistent
- No backup and restore
- Multi-threaded architecture



AOF persistence (Append only File)
→ logs every write operation received by the server in an AOF allows point-in-time recovery!

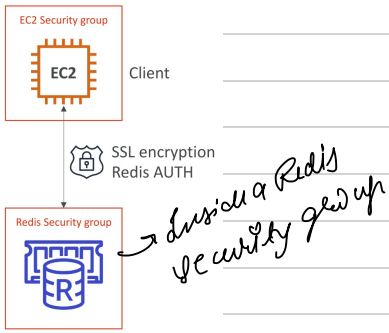
→ AOF provides better durability than RDB (Redis Database file) persistence, which takes the snapshot of the dataset at specific intervals!

→ AOF allows entire dataset recovery in the time of crash & also doesn't block current write operations.

ElastiCache – Cache Security

- ElastiCache supports IAM Authentication for Redis only
- IAM policies on ElastiCache are only used for AWS API-level security
- Redis AUTH → Redis already has this!
  - You can set a "password/token" when you create a Redis cluster
  - This is an extra level of security for your cache (on top of security groups)
  - Support SSL in flight encryption
- Memcached
  - Supports SASL-based authentication (advanced)

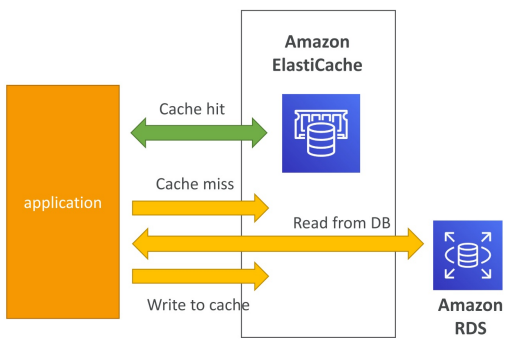
simple authentication & security layer.



Patterns for ElastiCache

1. Lazy Loading: all the read data is cached, data can become stale in cache
2. Write Through: Adds or update data in the cache when written to a DB (no stale data)
3. Session Store: store temporary session data in a cache (using TTL features)

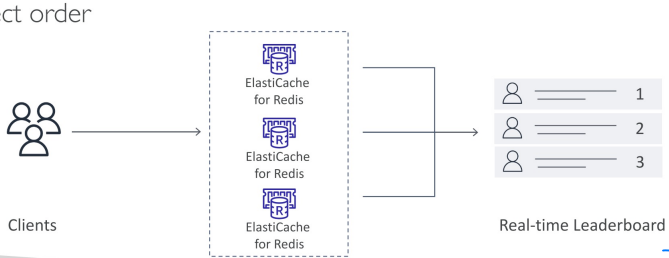
- *Quote: There are only two hard things in Computer Science: cache invalidation and naming things*



Lazy Loading illustrated

ElastiCache – Redis Use Case

- Gaming Leaderboards are computationally complex
- Redis Sorted sets guarantee both uniqueness and element ordering
- Each time a new element added, it's ranked in real time, then added in correct order



→ with a Redis cluster we can create a real-time leaderboard which use Redis sorted sets that rank & also order each time new element is added

→ We don't need to program the entire shit out instead leverage redis database features

Important ports vs RDS DB ports

FTP – 21
HTTP – 80
HTTPS – 443
SSH – 22
FSTP – 22

PostgreSQL – 5432
MySQL – 3306
Oracle RDS – 1521
MSSQL Server – 1433
Maria DB – 3306
Aurora – 5432 [if postgres]
or
3306 [if MySQL]

Database in AWS !

Database Types

online transaction Processing

- RDBMS (= SQL / OLTP): RDS, Aurora – great for joins
- NoSQL database – no joins, no SQL : DynamoDB (~JSON), ElastiCache (key / value pairs), Neptune (graphs), DocumentDB (for MongoDB), Keyspaces (for Apache Cassandra)
- Object Store: S3 (for big objects) / Glacier (for backups / archives)
- Data Warehouse (= SQL Analytics / BI): Redshift (OLAP), Athena, EMR
- Search: OpenSearch (JSON) – free text, unstructured searches
- Graphs: Amazon Neptune – displays relationships between data
- Ledger: Amazon Quantum Ledger Database
- Time series: Amazon Timestream

online analytic processing !

- Note: some databases are being discussed in the Data & Analytics section

→ data durability :- safeguarding data from loss or corruption in the events of failure or outages

Choosing the Right Database

- We have a lot of managed databases on AWS to choose from
- Questions to choose the right database based on your architecture:
  - Read-heavy, write-heavy, or balanced workload? Throughput needs? Will it change, does it need to scale or fluctuate during the day?
  - How much data to store and for how long? Will it grow? Average object size? How are they accessed?
  - Data durability? Source of truth for the data ?
  - Latency requirements? Concurrent users?
  - Data model? How will you query the data? Joins? Structured? Semi-Structured?
  - Strong schema? More flexibility? Reporting? Search? RDBMS / NoSQL?
  - License costs? Switch to Cloud Native DB such as Aurora?

Amazon RDS – Summary

- Managed PostgreSQL / MySQL / Oracle / SQL Server / DB2 / MariaDB / Custom
- Provisioned RDS Instance Size and EBS Volume Type & Size
- Auto-scaling capability for Storage
- Support for Read Replicas and Multi AZ
- Security through IAM, Security Groups, KMS , SSL in transit
- Automated Backup with Point in time restore feature (up to 35 days)
- Manual DB Snapshot for longer-term recovery
- Managed and Scheduled maintenance (with downtime)
- Support for IAM Authentication, integration with Secrets Manager
- RDS Custom for access to and customize the underlying instance (Oracle & SQL Server)

- **Use case:** Store relational datasets (RDBMS / OLTP), perform SQL queries, transactions

Amazon Aurora – Summary

- Compatible API for PostgreSQL / MySQL, separation of storage and compute
- Storage: data is stored in 6 replicas, across 3 AZ – highly available, self-healing, auto-scaling
- Compute: Cluster of DB Instance across multiple AZ, auto-scaling of Read Replicas
- Cluster: Custom endpoints for writer and reader DB instances
- Same security / monitoring / maintenance features as RDS
- Know the backup & restore options for Aurora
- **Aurora Serverless** – for unpredictable / intermittent workloads, no capacity planning
- **Aurora Global:** up to 16 DB Read Instances in each region, < 1 second storage replication
- **Aurora Machine Learning:** perform ML using SageMaker & Comprehend on Aurora
- **Aurora Database Cloning:** new cluster from existing one, faster than restoring a snapshot
- **Use case:** same as RDS, but with less maintenance / more flexibility / more performance / more features

Amazon ElastiCache – Summary

- Managed Redis / Memcached (similar offering as RDS, but for caches)
- In-memory data store, sub-millisecond latency
- Select an ElastiCache instance type (e.g., cache.m6g.large)
- 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 and Scheduled maintenance
- Requires some application code changes to be leveraged

- **Use Case:** Key/Value store, Frequent reads, less writes, cache results for DB queries, store session data for websites, cannot use SQL.

Fully managed, high available, NoSQL database with replication across 3 AZ. Key- value pair database.
Flagship AWS product which scales to massive workloads, distributed serverless database.
(with rds or elastic cache we need to provide a instance type) still servers at backend.
Millionns of request per seconds, trillions of rows 100s of TB of storage

Amazon DynamoDB – Summary

- AWS proprietary technology, managed serverless NoSQL database, millisecond latency
- Capacity modes: provisioned capacity with optional auto-scaling or on-demand capacity
- Can replace ElastiCache as a key/value store (storing session data for example, using TTL feature)
- Highly Available, Multi AZ by default, Read and Writes are decoupled, transaction capability
- DAX cluster for read cache, microsecond read latency
- Security, authentication and authorization is done through IAM
- Event Processing: DynamoDB Streams to integrate with AWS Lambda, or Kinesis Data Streams
- Global Table feature: active-active setup
- Automated backups up to 35 days with PITR (restore to new table), or on-demand backups
- Export to S3 without using RCU within the PITR window, import from S3 without using WCU
- Great to rapidly evolve schemas
- **Use Case:** Serverless applications development (small documents 100s KB), distributed serverless cache

Amazon S3 – Summary

- S3 is a… key / value store for objects
- Great for bigger objects, not so great for many small objects
- Serverless, scales infinitely, max object size is 5 TB, versioning capability
- **Tiers:** S3 Standard, S3 Infrequent Access, S3 Intelligent, S3 Glacier + lifecycle policy
- **Features:** Versioning, Encryption, Replication, MFA-Delete, Access Logs…
- **Security:** IAM, Bucket Policies, ACL, Access Points, Object Lambda, CORS, Object/Vault Lock
- **Encryption:** SSE-S3, SSE-KMS, SSE-C, client-side, TLS in transit, default encryption
- **Batch operations** on objects using S3 Batch, listing files using S3 Inventory
- **Performance:** Multi-part upload, S3 Transfer Acceleration, S3 Select
- **Automation:** S3 Event Notifications (SNS, SQS, Lambda, EventBridge)
- **Use Cases:** static files, key value store for big files, website hosting

DocumentDB

- Aurora is an "AWS-implementation" of PostgreSQL / MySQL …
- DocumentDB is the same for MongoDB (which is a NoSQL database)

- MongoDB is used to store, query, and index JSON data
- Similar "deployment concepts" as Aurora
- Fully Managed, highly available with replication across 3 AZ
- DocumentDB storage automatically grows in increments of 10GB

- Automatically scales to workloads with millions of requests per seconds

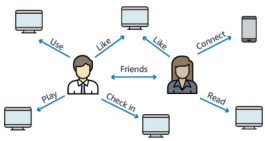
JSON

```
JSON
1  [
2      {
3          "year" : 2013,
4          "title" : "Turn It Down, Or Else!",
5          "info" : {
6              "directors" : [ "Alice Smith", "Bob Jones"],
7              "release_date" : "2013-01-18T00:00:00Z",
8              "rating" : 6.2,
9              "genres" : ["Comedy", "Drama"],
10             "image_url" : "http://ia.media-imdb.com/images/N/O9ERWA
11             "plot" : "A rock band plays their music at high volumes
12             "actors" : ["David Matthewman", "Jonathan G. Neff"]
13         }
14     },
15     {
16         "year": 2015,
17         "title": "The Big New Movie",
18         "info": {
19             "plot": "Nothing happens at all.",
20             "rating": 0
21         }
22     }
23 ]
```

Amazon Neptune { graph databases }

→ AWS managed graph database
(a NoSQL DB

- Fully managed **graph** database
- A popular **graph dataset** would be a **social network**
  - Users have friends
  - Posts have comments
  - Comments have likes from users
  - Users share and like posts…
- Highly available across 3 AZ, with up to 15 read replicas
- Build and run applications working with highly connected datasets – optimized for these complex and hard queries
- Can store up to billions of relations and query the graph with milliseconds latency
- Highly available with replications across multiple AZs
- Great for knowledge graphs (Wikipedia), fraud detection, recommendation engines, social networking

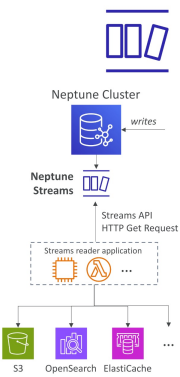


→ Any change to the DB is recorded & can be reviewed & enacted upon immediately, like sending notification if someone likes a post, etc.

Amazon Neptune – Streams

- **Real-time ordered** sequence of every change to your graph data
- Changes are available immediately after writing
- **No duplicates, strict order**
- Streams data is accessible in an **HTTP REST API**
- Use cases:
  - Send notifications when certain changes are made
  - Maintain your graph data synchronized in another data store (e.g., S3, OpenSearch, ElastiCache)
  - Replicate data across regions in Neptune

Application can query the stream data using REST API



Amazon Keyspaces (for Apache Cassandra)

Managed Apache Cassandra

- *Apache Cassandra is an open-source NoSQL distributed database*

- A managed Apache Cassandra-compatible database service
- Serverless, Scalable, highly available, fully managed by AWS
- Automatically scale tables up/down based on the application's traffic
- Tables are replicated 3 times across multiple AZ
- Using the Cassandra Query Language (CQL)
- Single-digit millisecond latency at any scale, 1000s of requests per second
- Capacity: On-demand mode or provisioned mode with auto-scaling (Similar to DynamoDB)
- Encryption, backup, Point-In-Time Recovery (PITR) up to 35 days

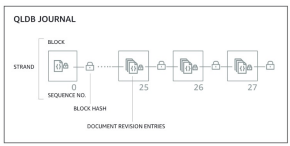
- Use cases: store IoT devices info, time-series data, …

Cassandra is a
- NoSQL
- Open Source
- distributed
- column store

(hybrid of key-value store & tabular DB)

Amazon QLDB

- QLDB stands for "Quantum Ledger Database"
- A ledger is a book recording financial transactions
- Fully Managed, Serverless, High available, Replication across 3 AZ
- Used to review history of all the changes made to your application data over time
- Immutable system: no entry can be removed or modified, cryptographically verifiable



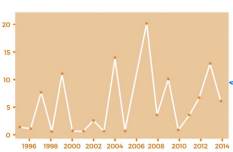
- 2-3x better performance than common ledger blockchain frameworks, manipulate data using SQL
- Difference with Amazon Managed Blockchain: no decentralization component, in accordance with financial regulation rules

→ Behind the scenes there is a journal which has a seq. of modi. , so anytime an modification is made there, is a cryptographic hash which is computed that guarantees that nothing has been deleted or changed.

Amazon Timestream {time series DB}

- Fully managed, fast, scalable, serverless time series database
- Automatically scales up/down to adjust capacity
- Store and analyze trillions of events per day
- 1000s times faster & 1/10th the cost of relational databases
- Scheduled queries, multi-measure records, SQL compatibility
- Data storage tiering: recent data kept in memory and historical data kept in a cost-optimized storage
- Built-in time series analytics functions (helps you identify patterns in your data in near real-time)
- Encryption in transit and at rest

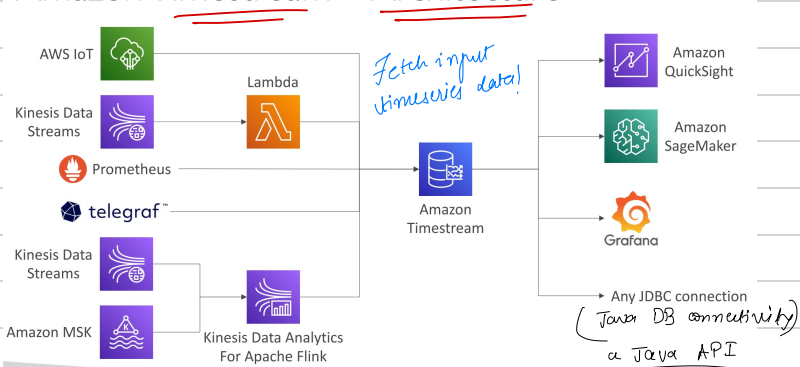
- Use cases: IoT apps, operational applications, real-time analytics, …



specific for time series data because using a relational db for it is very costly.

→ store the readings & do fast analytics.

Amazon Timestream – Architecture



Fetch input timeseries data!

(Java DB connectivity) a Java API