Relational Database Service (RDS)

Database Refresher & Models

DBs are systems which store and manage data: DB Types:

1. Relational Database Management System (RDBS, commonly SQL)

- RDBS has a structure in & between tables of data; rigid schema
- Fixed relationship between tables. These relationships are defined in advance

2. Non-relational (NoSQL); more relaxed schemas

- Common Examples of Non-relational DBs:
    - Key-value DB's: just key/value pairs and no schema otherwise. Highly scalable and fast. Good for in-memory caching
    - Wide Column Store: Every item in table has same key layout (Partition Key, + other key). Items in a table can have differing attributes. Eg. DynamoDB
    - Document Database. Good for DBs of orders or contact list. Key/Value but the value is the accessible document+data
    - Column Store (AKA OLTP, good for transactional processes like collections of orders) and Row Store (RedShift; good for analytics)
    - Graph. Data and the relationships of data stored in DB

RDS - ACID vs BASE

This lesson steps through the ACID and BASE Database transaction models and introduces the CAP Theorem

- transaction models AKA define a few things about transactions to and from a DB
- CAP Theorem - Consistency, Availability, Partition Tolerant (resilience). CAP Theorem says that any DB product only capable of delivering a max of TWO of these three factors
    - Consistency: Every read will return the most recent write
    - Availability: Every request will get a non error response (without the guarantee that is the most recent write!)
    - Partition Tolerant: The system continues to operate despite network partitions (eg. a network outage)
- ACID focuses on consistency, BASE focuses on Availability

ACID - atomic, consistent, isolated, durable

```
Exam: 'ACID' mentioned in exam? RDS based DB. Acid limits DB scalability
```

- Atomic: either all or NO components of a transaction succeed or fail
- Consistent: transactions move DB from one VALID state to another; nothing in-between is allowed
- Isolated: if multiple transactions occur at once, they don't interfere with each other
- Durable: once committed, transactions are durable; stored on non-volatile memory and are resilient to outages/crashes

BASE - Basically available, soft state, eventually consistent

- Basically Available: Read / Write ops are available 'as much as possible' but without any consistency guarantees
- Soft State: The DB doesn't enforce consistency, this is offloaded onto the application/user
- Eventually Consistent: If we wait long enough, reads from the system will be consistent BASE = highly scalable, Eg DynamoDB `Exam: BASE = noSQL DBs like DynamoDB Exam: But if ACID + BASE are mentioned, it often refers to DynamoDB Transactions which add ACID to the BASE model`

RDS - Databases on EC2

NOTE: Running DB on EC2 is generally bad practice as there is usually a better alternative Why you might run DB on EC2...

- If you need access to the DB instance OS
- Advanced DB Option tuning (DBROOT)
- Vendor/decision maker demands
- Need to run DB or DB Version AWS doesn't provide
- Need a specific OS/DB combo or architecture AWS doesn't provide

Why you SHOULD NOT run DB on EC2

- Admin overhead is high to manage on EC2 and DBHost
- Backups / disaster recovery management becomes more difficult
- EC2 is a single AZ. If zone fails, DB access fails
- Features. AWS DB products are great and you could be missing out
- EC2 is ON or OFF. No serverless easy scaling
- No Replication
- Performance. AWS invests time into optimisation / features that you're missing

Demo - RDS - Splitting Wordpress Monolith => APP & DB

- Moving DB to new AZ, separate from Webserver/App

Steps:

1. 1 click deploy > Create complete
2. Copy WP instance IP, paste to browser > Title "The Best Cats!!", user "admin", PW "an1m4ls4l1f3", email test@test.com > Install
3. A4L-WordPress, ec2 connect `mysqldump -u root -p a4lwordpress > a4lwordpress.sql` > PW an1m4ls4l1f3
4. Inject a4lwordpress.sql into new DB `mysql -h privateipof_a4l-mariadb -u a4lwordpress -p a4lwordpress < a4lwordpress.sql` 5.... Error at step 4.
5. Clean up > CFN > Delete Stack

RDS - Architecture

- Known as "DBaaS", but more accurately RDS is a DBSaaS "DB Server as a Service"
- RDS provides multiple databases on one DB Server (instance)

- RDS is a managed service, so no access to OS or SSH Access
- RDS operates within a VPC
- RDS Instance can have more than 1 DB in it, and each RDS Instance has its own Dedicated EBS-provided storage NOTE: Amazon Aurora is DIFFERENT from RDS Cost: Billed for resource allocation -- Cost #1: instance size/type -- Cost #2: Multi-AZ or not -- Cost #3: Storage type/amount -- Cost #4: Data transferred between AZs (inside VPC is free) -- Cost #5: Backups & Snapshots -- Cost #6: Licensing

RDS - DEMO - Migrating EC2 DB into RDS

Create a MySQL RDS instance and migrate the Wordpress Database from the self-managed MariaDB server running on EC2 into this RDS instance.

Steps:

1. 1 click deploy
2. Set up website: EC2 > A4L-Wordpress instance, copy IPv4 address, paste into browser > title "The Best Cats!!", username "admin", PW "4n1m4ls4L1f3", email "test@example.com" > install WP
3. Set Up data to move: Blog Post > Posts > delete Hello World > Add New > H1 "The Best Cats Ever!!" > add Gallery widget and upload zipped file, Publish
4. Set up RDS DB: RDS > Subnet Groups > Create a DB Subnet Group "a4lsngroup" for title/descrip > VPL a4l > Add subnets, select us-east-1a/b/c > Pick your subnets > go to VPC, Subnets and look for full names of sn-db-A/B/C to select those ones > Create
5. Provision DB: RDS > DB > Create DB > Standard Create > select MySQL > engine version (currently 8.0.32 in demo video) > Template, Free Tier > DB instance identifier "a4lwordpress" > admin "a4lwordpress" > PW "4n1m4ls4L1f3" > Storage, uncheck "Enable storage autoscaling" > Connectivity, select a4l-vpc1 > VPC security group (firewall), Create New > name "a4lvpc-rds-sg" > Add'l Configuration, Initial DB Name "a4lwordpress" > Create DB
6. Config Security Group so RDS DB is connected: RDS > DB's > a4lwordpress > Connecitivity & Security tab, click the VPC SG link and open in new tab > find your SG > Inboud Rules > Edit > Add Rule, Type MYSQL/Aurora, Source (magnifying glass icon) "MIGRATE2RDS-Instance..." > Save Rule
7. Migrate Data to new RDS DB, create backup file to move: EC2 Connect with A4L-Wordpress > `mysqldump –h PRIVATEIPOFMARIADBINSTANCE –u a4lwordpress –p a4lwordpress > a4lwordpress.sql` -> replace private IP with Private IPv4 of a4l DB instance > Enter > use PW above > Enter
8. Move backup SQL DB instance into RDS: `mysql –h CNAMEOFRDSINSTANCE –u a4lwordpress –p a4lwordpress < a4lwordpress.sql` -> Replace CNAME with Endpoint name in RDS db instance > Enter
9. Point WP at RDS instance: `cd /var/www/html` > `sudo nano wp–config.php` -> Go to Database Hostname and replace the IPv4 pointing at the WB DB EC2 instance with the Endpoint name of the RDS instance... Looks like... /** Database hostname */ define('DB_HOST', 'a4lwordpress.cloy7bpdvh9v.us-east-1.rds.amazonaws.com'); > ctrl+o to Save, ctrl+x to Exit nano 10... Demo broke at this stage
10. Cleanup: RDS > Delete DB > VPC > Delete Security group "a4lvpc-rds-sg" > Cloufformation > Delete Stack

Relational Database Service (RDS) MultiAZ - Instance and Cluster Deployments

MultiAZ is a feature of RDS which provisions a HIGHLY AVAILABLE instance set.

- The product provides MultiAZ instance where a standby replica is kept in sync Synchronously with the primary instance.
    - The standby replica cannot be used for any performance scaling, only availability.
- MultiAZ cluster mode: where a write and two reader instances are kept in sync Synchronously. The reader instances can be used for read operations allowing for limited read scaling.
- Backups, software updates and restarts can take advantage of MultiAZ to reduce user disruption.

A. MultiAZ Instance Deployments

- Synchronous replication of the primary DB to the standby DB (data is written to both at the same time)
- Standby is only for backups and is never accessed unless a failover is required. Failover can take 60-120s
- Not free
- Only ONE standby replica due to architecture
- Same region only, but different AZs in region
- Backups taken from Standby to improve Primary performance

B. MultiAZ Cluster Deployments

NOTE: Keep in mind the differences between MultiAZ Cluster Deployments and Amazon Aurora

- ONE Writer replicates to TWO reader instances, all in different AZs. With Aurora, you can have more than 2 reader instances
- These Readers (called Standby in Instance Deployment) can be used for Read operations, allowing for some read scaling
- Faster Hardware than Instance Deployment
- Faster failover, ~35s
- Writes considered "committed" when 1 of the readers has confirmed it

RDS Automatic Backup, RDS Snapshots and Restore

- RDS is capable of performing Manual Snapshots and Automatic backups -- Manual snapshots are performed manually and live past the termination of an RDS instance -- Automatic backups can be taken of an RDS instance with a 0 (Disabled) to 35 Day retention.
- Automatic backups also use S3 for storing transaction logs every 5 minutes - allowing for point in time recovery.
- Snapshots can be restored .. but create a new RDS instance. EXAM: RDS Backups live in S3 but are AWS Managed, so you never seem them

RDS Backup - Snapshots (Manual)

- First snapshot is full copy, after that it's incremental copies
- Snapshots DON'T expire, they live beyond the termination of an RDS instance -- you have to delete them manually
- Technically, you could reach a 5 minute Recovery Point Objective

RDS Backup - Automated Backups

- Occur once per day
- Think of them as automated snapshots
- If using single AZ, plan for a small IO pause and do it during downtime
- Every 5 mins, Transaction Logs are recorded to S3. With these plus snapshots; you have a 5 Minute Recovery Point Objective
- Retention Period. 0 - 35 days before AWS deletes the automated backups. If you select 35 days, you can restore to any point in time over that 35 day period
- To avoid losing data beyond 35 days, you need to do a final manual snapshot

RDS Backups - Cross-Region Replication

- RDS can replicate backups to another region (both snapshots and transaction logs)
- Charges apply for the cross-region data copy and for the storage in the destination region
- This feature must be enabled

RDS Backups - Restores

- Creates a new RDS instance when you restore an automated backup or manual snapshot
- Snapshot restore is a single point in time, at the time of creation of the snapshot
- Automated backups are restorable to any 5 minute point in time

RDS Read Replicas

RDS Read Replicas can be added to an RDS Instance - 5 direct per primary instance.

- READ-ONLY
- They can be in the same region, or cross-region replicas.
- They provide read performance scaling for the instance, but also offer low RTO recovery for any instance failure issues
- They don't help with data corruption as the corruption will be replicated to the RR
- Asynchronous replication EXAM: Synchronous is MultiAZ, asynchronous is Read Replicas

Read Replicas - Why do they matter?

- Read performance and read scaling.
    - You get 5 direct read-replicas per DB instance, each providing add'l instance of read performance
    - Read-replicas can have read-replicas, but then lag starts to be a problem
    - Global performance improvements
- Recovery Point and Recovery Time Objective benefits
    - Snapshots/Backups improve RPO, with RR's offering a near 0 RPO (little potential for data loss)
    - RR's have low RTO. NOTE: For Failure only, not corruption. The corruption is also likely replicated
- Read-replicas are Read-only until promoted (until activated), then they become a normal RDS instance
- Since you can cross-region replicate, you get Global Resilience

Demo - RDS - MultiAZ & Snapshot Restore with RDS

Working with RDS Multi AZ mode and snapshot restores

Steps:

1. 1 click deploy
2. Set up WP (The Best Cats!!, admin, 4n1m4ls4L1f3, test@test.com), create blog post
3. Create snapshot: RDS > select DB > Actions, Take snapshot > name "a4lwordpress-with-cat-post-mysql-8032" > Take snapshot
4. Enable MultiAZ: RDS > select database > Modify > Enable MultiAZ (this part costs money so I'm not doing it)

Demo 2 - Restore RDS in the case of data corruption Steps:

1. Change WP blog post text to simulate 'corrupted' data
2. RDS > Snapshots > select snap, Actions, Restore Snapshot > name "a4lwordpress-restore" > select burstable somethings > select RDSsecurity for SG > Create/Restore NOTE: Restoring RDS from snapshot creates a NEW instance
3. Point to new DB: EC2 connect > cd /var/www/html > `sudo nano wp-config.php` > Get new restored RDS endpoint and prepare to paste it into DB hostname in nano > paste into correct spot > ctrl+o to save/write > ctrl+x to exit
4. Cleanup > Delete restore snapshot > CFN delete stack

RDS - Security

- In transit encryption (SSL/TLS) is available for RDS and can be set to mandatory on a per user basis
- At Rest encrpytion available via KMS and EBS encryption:
    - encrypted at the storage level by the Data Encryption Key (DEK) that is generated by KMS using either a customer managed key (CMK) or an AWS managed key
    - host can see the encrypted data: encryption is done by the host using DEK when writing to the EBS volume
    - DEK is used to encrypt: storage, logs, snapshots and replicas
- Transparent Data Encryption (TDE): Encryption is done within the database engine.
    - Data is already encrypted when host writes data to EBS volume
    - RDS Oracle supports TDE using CloudHSM (has much stronger key controls, you remove AWS from the chain of trust)
- Encryption can't be removed once enabled

RDS - Security - IAM Authentication

RDS uses local database username/password authentication, a user is created when the RDS instance is created. They're not IAM users and are outside of the control of AWS. You can configure RDS to use IAM user authentication against a DB

- Attach policy to IAM user or IAM role to generate a token that can be used to AUTHENTICATE against the DB
- Important: AUTHORIZATION and permissions is still handled internally by DB engine, authentication can be done using IAM if enabled on the RDS instance

Amazon RDS Custom

Amazon RDS Custom is a managed database service for applications that require customization of the underlying operating system and database environment.

- fills the gap between RDS and EC2 running a DB engine
- RDS is fully managed, so OS/engine access is limited. DB on EC2 is self-managed, but has overhead. RDS Custom bridges this gap
- RDS Custom works only for MS SQL and Oracle
- You can connect using SSH, RDP, Session Manager and get access to the OS and DB engine
- Customizing RDS Custom? Make sure to look at RDS DB Automation settings to ensure no disruptions (pause automation, customize, restart automation)

Aurora Architecture

Aurora [Provisioned] is a AWS designed relational database engine officially part of RDS. Aurora implements a number of radical design changes which offer significant performance and feature improvements over other RDS database engines.

- Uses a "cluster", which other RDS engines don't have. Cluster contains a primary instance + 0 or more replicas which have read capability during normal operation
- For storage, Aurora uses a Cluster Volume, not local storage; faster provisioning, higher availability, better performance
- Max cluster volume space of 128 TiB
- Can have up to 15 Replicas
- Storage is SSD based; high IOPS, low latency

Aurora - Billing

- "High watermark billing" -- If you scale up to 50 TiB and then scale down to 40, you'll still be paying for the full 50. Freed up storage can be re-used
- No free-tier option
- Aurora does not support micro-instances. Beyond RDS SingleAZ (micro), Aurora offers better value
- 100% Database size in backups are included
- Backup restores create a new cluster
- Backtrack: A rollback feature that lets you roll back to your existing cluster but to a previous point in time. "in place rewinds"

Aurora Serverless

Serverless VS Provisioned Aurora Serverless provides a version of the Aurora DB product where you don't need to statically provision DB instances of a certain size or worry about managing DB instances

- Removes the overhead of managing individual DB instances
- Uses concept of Arurora Capacity Units (ACUs)
- Serverless cluster has a min and max ACU tht you choose, can go to 0 and be paused, cluster adjusts based on load
- Same resilience as Provisioned (6 copies across AZs)

Aurora Serverless - Uses

- For infrequently used applications
- New applications (where you're unsure of load levels)
- For variable workloads / unpredictable workloads
- For dev and test databases (since it can pause itself)
- Multi-tenant applications (like subscribing to app), where load is directed related to customer use/revenue

DEMO - Migrating to Aurora Serverless. Advised to not actually do per tutorial, just watch

Aurora Global Database

Aurora global databases are a feature of Aurora Provisioned clusters which allow data to be replicated globally providing significant RPO and RTO improvements for business continuity and disaster recovery planning. Additionally, global databases can provide performance improvements for customers .. with data being located closer to them, in a read-only form.

- Global level Aurora, primary region and 5 secondary region. Secondary clusters are read-only. Each secondary region has up to 16 replicas EXAM: Great for Cross-Region Disaster Recovery and Business Continuity. ~1s replication times EXAM: Global Read Scaling: Low latency performance improvements to global customers EXAM: ~1s replication or less between regions. Recovery Point Objective (RPO) of 1 second and a Recovery Time Objective (RTO) of less than 1 minute

Aurora Provisioned - Multi-master writes

Allows Aurora cluster to have multiple instances that are all capable of both reads and writes

- No lengthy failovers since replicas already have R/W capability

RDS Proxy

Amazon RDS Proxy is a fully managed, highly available database proxy for Amazon Relational Database Service (RDS) that makes applications more scalable, more resilient to database failures, and more secure.

- Why do you want RDS Proxy? Opening/Closing connections consumes resources, takes time which creates latency. Esp true with serverless. Handling failure of DB is difficult
- RDS proxy changes your architecture by creating long term connection pools; instead of your app connecting to a DB every time its used, its connected to a proxy which are already open to a pool of DB connections
- Abstracts client away from DB failure

RDS Proxy - When to use?

- too many connection errors esp. if using small/burst instances
- when using AWS Lambda: time saved, connection reuse, IAM auth
- Long running connections (SaaS apps) needing low latency
- Where resilience to DB failure is a priority
- To further reduce failover time

- Make DB transparent to application

RDS Proxy - Key Facts (EXAM)

- Fully managed DB proxy for RDS/Aurora
- Auto-scaling, highly available by default
- Provides connection pooling to reduce DB load
- RDS proxy only accessibly from a VPC
- Accessed via a proxy endpoint; no app changes
- can enfore SSL/TLS
- Can reduce failover time by over 60%
- Abstracts failure away from your applications

RDS - Database Migration Service (DMS)

- The Database Migration Service (DMS) is a managed service which allows for 0 data loss, low or 0 downtime migrations between 2 database endpoints.
- The service is capable of moving databases INTO or OUT of AWS.
- Runs using a replication instance via EC2, requiring Source/Destination endpoints and Source/Target Databases
- ONE of the endpoints must be running on AWS. Endpoints store connection info for source/target DBs
- Replication Instance runs one or more Replication Tasks
- Jobs can be 1 of 3 types:

1. Full Load (one-off full-data migration),
2. Full Load + CDC (change data capture, for ongoing replication),
3. or CDC only (to replicate only data changes, using native tools to bulk move initial data outside of DMS)

- Schema Conversion Tool assists with schema conversion EXAM: DB Migration question? Default to DMS as answer, especially if talking about "no-downtime migration"

DMS - Schema Conversion Tool

- Used when converting one DB engine to another
- NOT USED for moving between COMPATIBLE DB ENGINES

DMS and Snowball devices

- A physical device for moving data physically
- Larger migrations (multi-TB in size) over networks takes times and consumes capacity, so DMS can utilize Snowball