RDS - Relational Data Source

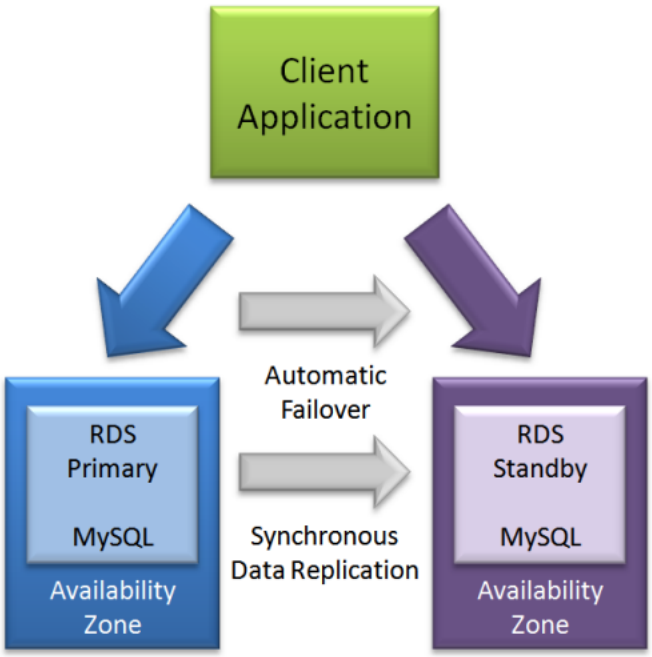
https://aws.amazon.com/rds/details/multi-az/
https://aws.amazon.com/rds/faqs/
https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

With AWS Service-Based databases in the RDS service, you do not need to be concerned with operating system installation and configuration, but you will still need to create the databases, manage security, and perform backup procedures.

- Modify at run-time and allocate more size to database
- Dynamically Convert the instance to Single AZ to Multi AZ
- **NOTE:**
  - For 'Scale-out' use READ-REPLICAs
  - For 'High-Availability' use Multi AZ

HOW FAIL-OVER is HANDLED

In Amazon RDS, failover is automatically handled so that you can resume database operations as quickly as possible without administrative intervention in the event that your primary database instance went down. When failing over, Amazon RDS simply flips the canonical name record (CNAME) for your DB instance to point at the standby, which is in turn promoted to become the new primary.



https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html

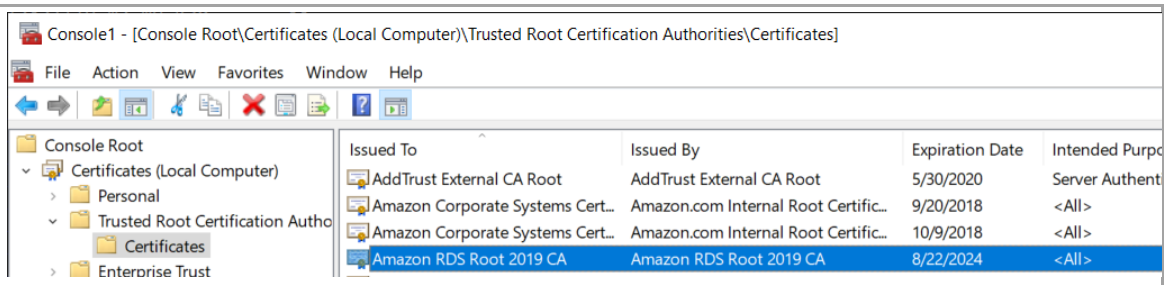
- Amazon RDS Multi-AZ deployments provide enhanced availability and durability for Database (DB) Instances, making them a natural fit for production database workloads.
- When you provision a Multi-AZ DB Instance, Amazon RDS automatically creates a primary DB Instance and synchronously replicates the data to a standby instance in a different Availability Zone (AZ).
- Each AZ runs on its own physically distinct, independent infrastructure, and is engineered to be highly reliable.

- In case of an infrastructure failure, Amazon RDS performs an automatic failover to the standby (or to a read replica in the case of Amazon Aurora), so that you can resume database operations as soon as the failover is complete.
- Since the endpoint for your DB Instance remains the same after a failover, your application can resume database operation without the need for manual administrative intervention.

- The chief benefits of running your DB instance as a Multi-AZ deployment are enhanced database durability and availability.
- The increased availability and fault tolerance offered by Multi-AZ deployments make them a natural fit for production environments.
- **- Increased database availability in the case of system upgrades like OS patching or DB Instance scaling.**
- **- Provides enhanced database durability in the event of a DB instance component failure or an Availability Zone outage.**

SSL with RDS

- You can use Secure Sockets Layer (SSL) to encrypt connections between your client applications and your Amazon RDS DB instances running Microsoft SQL Server.
- SSL support is available in all AWS regions for all supported SQL Server editions.
- When you create an SQL Server DB instance, Amazon RDS creates an SSL certificate for it.
- The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.
- There are 2 ways to use SSL to connect to your SQL Server DB instance:
  - **Force SSL for all connections** — this happens transparently to the client, and the client doesn't have to do any work to use SSL.
  - **Encrypt specific connections** — this sets up an SSL connection from a specific client computer, and you must do work on the client to encrypt connections.



- You can force all connections to your DB instance to use SSL, or you can encrypt connections from specific client computers only. To use SSL from a specific client, you must obtain certificates for the client computer, import certificates on the client computer, and then encrypt the connections from the client computer.
  - If you want to force SSL, use the rds.force_ssl parameter.
  - By default, the rds.force_ssl parameter is set to false.
  - Set the rds.force_ssl parameter to true to force connections to use SSL.
  - The rds.force_ssl parameter is static, so after you change the value, you must reboot your DB instance for the change to take effect.

References:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/SQLServer.Concepts.General.SSL.Using.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.Options.TDE.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAMDBAuth.html

Check out this Amazon RDS Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

BEST PRACTICES
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html#CHAP_BestPractices.MySQLStorage

- Database monitoring includes real-time monitoring and log review
- You can see performance statistics related to databases
- The monitoring of a database should be used to improve the overall performance and availability of the database

RDS

Fast disaster recovery

Scalable

Ease of management

Backups, patch management, security management

RDS: How?

Create using API or web console

Specify desired database software

MySql, Postgres, Aurora, MariaDB, Oracle, SQL Server

Specify performance size

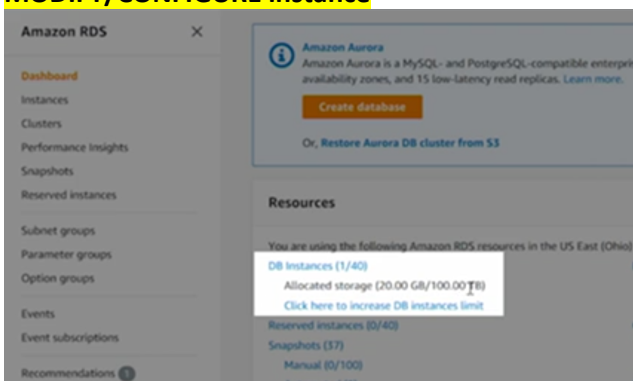
Monitor with CloudWatch

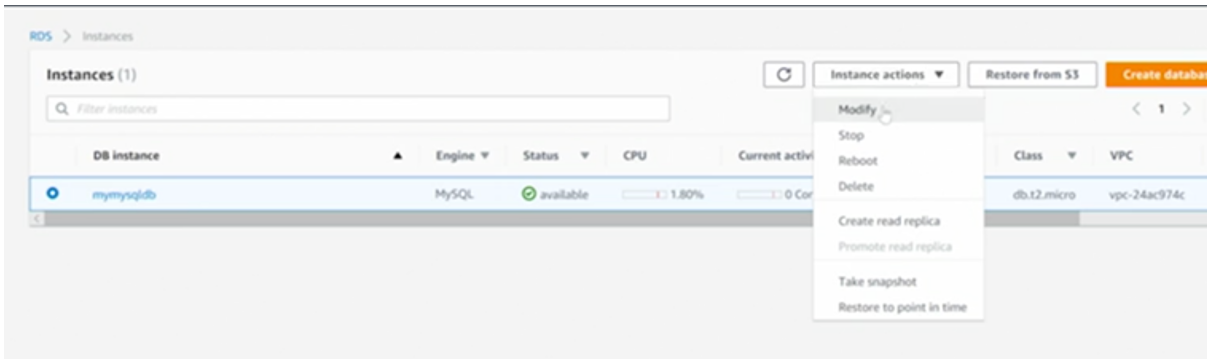
RDS: Managed for You

- No SSH or root access
- Updates applied for you
- Automatic backups
- High availability/scalability/fault tolerance
- Read replicas

- MySQL is a database option that is available in the free tier
- When you implement databses, it is important to consider the cost of database implementations

MODIFY/CONFIGURE instance





AWS - RDS IAM authentication

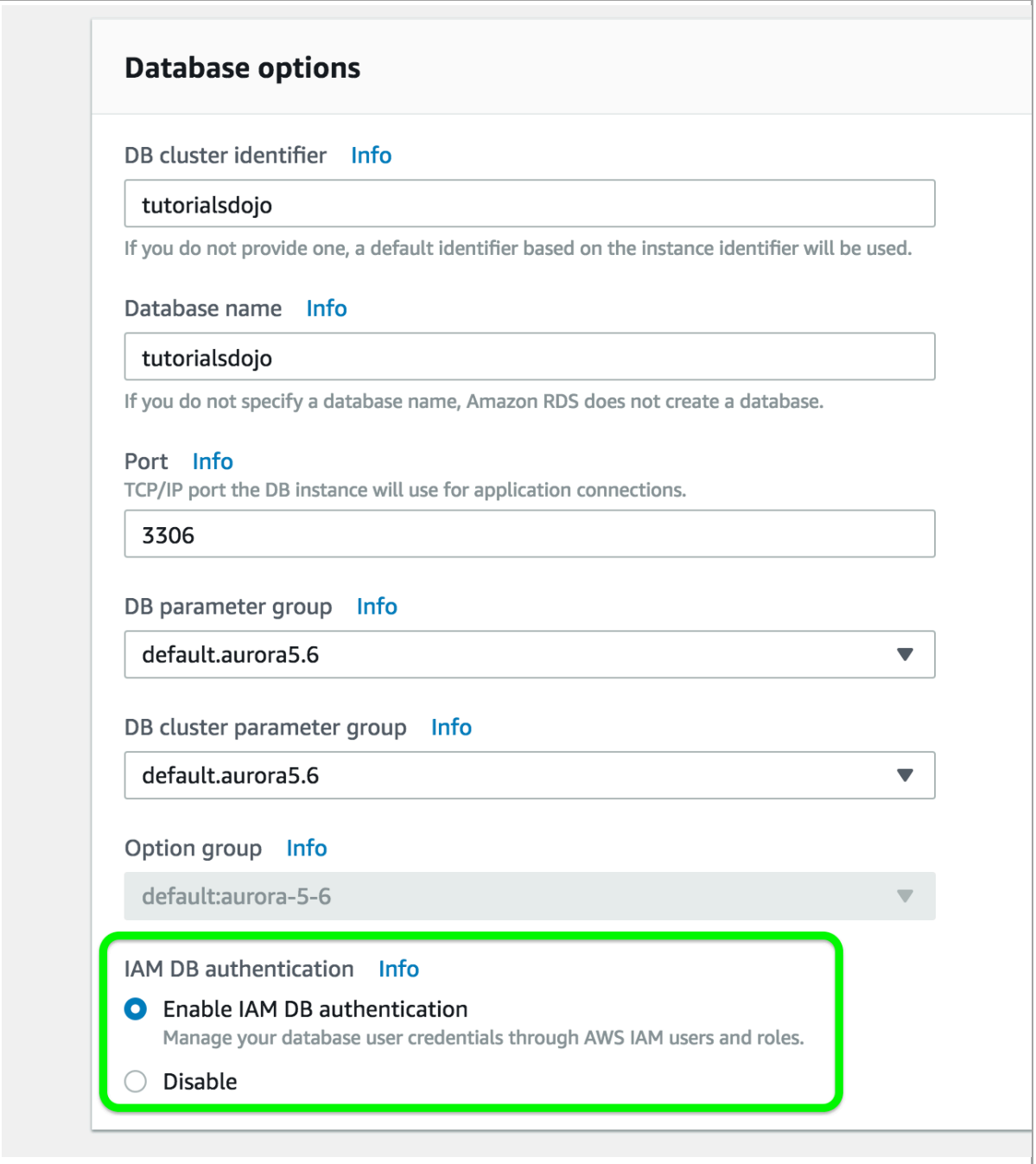
IAM DB Authentication

- You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication.
- IAM database authentication works with MySQL and PostgreSQL.

- With this authentication method, you don't need to use a password when you connect to a DB instance.
- Instead, you use an authentication token.

Authentication Token

- An ***authentication token*** is a unique string of characters that Amazon RDS generates on request.
- Authentication tokens are generated using AWS Signature Version 4.
- Each token has a lifetime of 15 minutes.
- You don't need to store user credentials in the database, because authentication is managed externally using IAM. You can also still use standard database authentication.



IAM database authentication provides the following benefits:

- Network traffic to and from the database is encrypted using Secure Sockets Layer (SSL).
- You can use IAM to centrally manage access to your database resources, instead of managing access individually on each DB instance.
- For applications running on Amazon EC2, you can use profile credentials specific to your EC2 instance to access your database instead of a password, for greater security

Reference:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAMDBAuth.html

Check out this Amazon RDS Cheat Sheet:

https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/

AWS - RDS - Read Replicas

Read Replicas

- Amazon RDS Read Replicas provide enhanced performance and durability for database (DB) instances.
- This feature makes it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- You can create one or more replicas of a given source DB Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput.
- Read replicas can also be promoted when needed to become standalone DB instances.
- For the MySQL, MariaDB, PostgreSQL, and Oracle database engines, Amazon RDS creates a second DB instance using a snapshot of the source DB instance.
- It then uses the engines' native asynchronous replication to update the read replica whenever there is a change to the source DB instance.
- The read replica operates as a DB instance that allows only read-only connections; applications can connect to a read replica just as they would to any DB instance.
- Amazon RDS replicates all databases in the source DB instance.

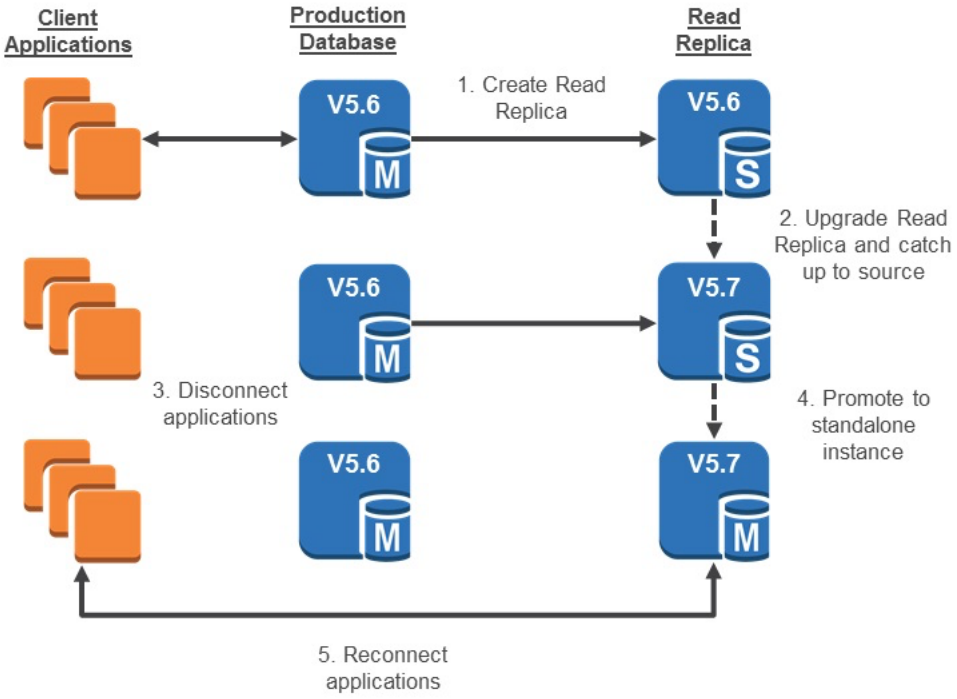
Multi-AZ deployments	Multi-Region deployments	Read replicas
Main purpose is high availability	Main purpose is disaster recovery and local performance	Main purpose is scalability
Non-Aurora: synchronous replication; Aurora: asynchronous replication	Asynchronous replication	Asynchronous replication
Non-Aurora: only the primary instance is active; Aurora: all instances are active	All regions are accessible and can be used for reads	All read replicas are accessible and can be used for readscaling
Non-Aurora: automated backups are taken from standby; Aurora: automated backups are taken from shared storage layer	Automated backups can be taken in each region	No backups configured by default
Always span at least two Availability Zones within a single region	Each region can have a Multi-AZ deployment	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Non-Aurora: database engine version upgrades happen on primary; Aurora: all instances are updated together	Non-Aurora: database engine version upgrade is independent in each region; Aurora: all instances are updated together	Non-Aurora: database engine version upgrade is independent from source instance; Aurora: all instances are updated together
Automatic failover to standby (non-Aurora) or read replica (Aurora) when a problem is detected	Aurora allows promotion of a secondary region to be the master	Can be manually promoted to a standalone database instance (non-Aurora) or to be the primary instance (Aurora)

Multi-AZ Deployments	Read Replicas
Synchronous replication – highly durable	Asynchronous replication – highly scalable
Only database engine on primary instance is active	All read replicas are accessible and can be used for read scaling
Automated backups are taken from standby	No backups configured by default
Always span two Availability Zones within a single Region	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Database engine version upgrades happen on primary	Database engine version upgrade is independent from source instance
Automatic failover to standby when a problem is detected	Can be manually promoted to a standalone database instance

- When you create a read replica for Amazon RDS for MySQL, MariaDB, PostgreSQL, and Oracle, Amazon RDS sets up a secure communications channel using public-key encryption between the source DB instance and the read replica, even when replicating across regions.
- Amazon RDS establishes any AWS security configurations such as adding security group entries needed to enable the secure channel.
- You can also create read replicas within a Region or between Regions for your Amazon RDS for MySQL, MariaDB, PostgreSQL, and Oracle database instances encrypted at rest with AWS Key Management Service (KMS).

In this scenario, the use of Amazon RDS Read Replicas is the best option. It provides enhanced performance and durability for database (DB) instances. This feature makes it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. You can create one or more replicas of a given source DB

Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput. Read replicas can also be promoted when needed to become standalone DB instances. Read replicas are available in Amazon RDS for MySQL, MariaDB, Oracle, and PostgreSQL as well as Amazon Aurora.



References:

https://aws.amazon.com/rds/details/read-replicas/

https://aws.amazon.com/rds/features/multi-az/

Check out this Amazon RDS Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

Additional tutorial - How do I make my RDS MySQL read replica writable?

https://youtu.be/j5da6d2TIPc

References:

https://aws.amazon.com/caching/database-caching/

https://aws.amazon.com/rds/details/read-replicas/

https://aws.amazon.com/elasticache/

Check out this Amazon RDS Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

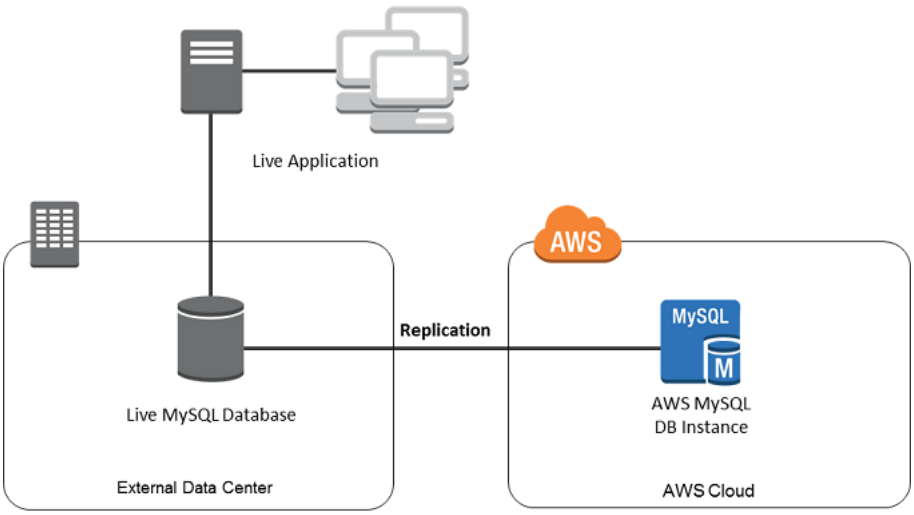
AWS - RDS to OnPrem Replicas

Create an IPSec VPN connection using either OpenVPN or VPN/VGW through the Virtual Private Cloud service. Prepare an instance of MySQL running external to Amazon RDS. Configure the MySQL DB instance to be the replication source. Use mysqldump to transfer the database from the Amazon RDS instance to the on-premises MySQL instance and start the replication from the Amazon RDS Read Replica.

Explanation

Amazon supports Internet Protocol security (IPsec) VPN connections. IPsec is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a data stream. Data transferred between your VPC and datacenter routes over an encrypted VPN connection to help maintain the confidentiality and integrity of data in transit. An Internet gateway is not required to establish a hardware VPN connection.

You can set up replication between an Amazon RDS MySQL (or MariaDB DB instance) that is running in AWS and a MySQL (or MariaDB instance) to your on-premises data center. Replication to an instance of MySQL running external to Amazon RDS is only supported during the time it takes to export a database from a MySQL DB instance.



To allow communication between RDS and to your on-premises network,

- you must first set up a VPN or an AWS *Direct Connect* connection.
- Once that is done, just follow the below the steps to perform the replication:
    - Prepare an instance of MySQL running external to Amazon RDS.

- Configure the MySQL DB instance to be the replication source.
- Use **mysqldump** to transfer the database from the Amazon RDS instance to the instance external to Amazon RDS (e.g. on-premises server)

- Start replication to the instance running external to Amazon RDS.

References:

http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_VPN.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/MySQL.Procedural.Exporting.NonRDSRepl.html

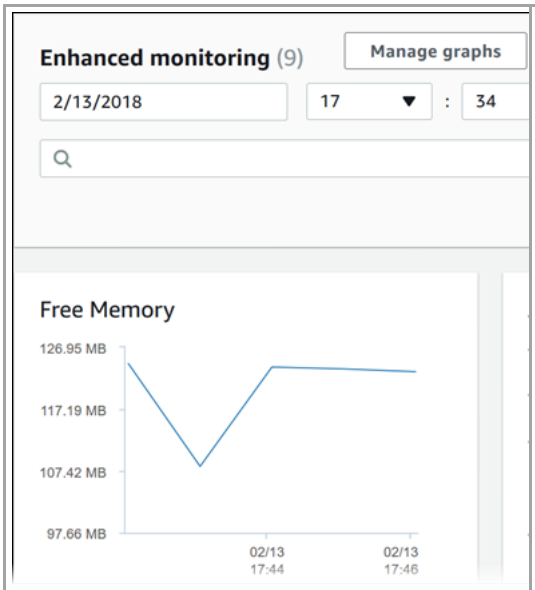
Tutorials Dojo's AWS Certified Solutions Architect Professional Exam Study Guide:

https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-solutions-architect-professional/

AWS - RDS Monitoring

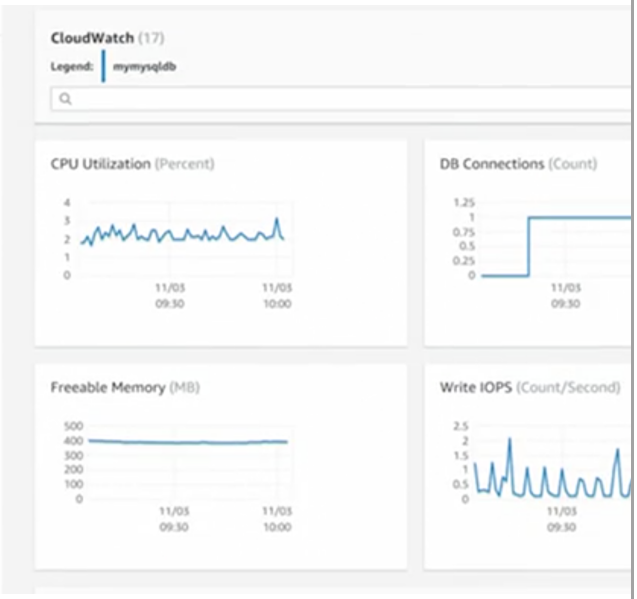
- **Amazon RDS** provides metrics in real time for the operating system (OS) that your DB instance runs on.
- You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output from CloudWatch Logs in a monitoring system of your choice.

- **CloudWatch** gathers metrics about CPU utilization from the hypervisor for a DB instance, and **Enhanced Monitoring** gathers its metrics from an agent on the instance.
- As a result, you might find differences between the measurements, because the hypervisor layer performs a small amount of work.
- The differences can be greater if your DB instances use smaller instance classes, because then there are likely more virtual machines (VMs) that are managed by the hypervisor layer on a single physical instance.
- Enhanced Monitoring metrics are useful when you want to see how different processes or threads on a DB instance use the CPU.



CloudWatch MONITORING & metrics

- CPU utilization
- DB connections
- Free Storage Space
- Free-able Memory
- Write IOPS
- Read IOPS



In RDS, the Enhanced Monitoring metrics shown in the Process List view are organized as follows:

- RDS child processes
- RDS processes
- OS processes

RDS child processes –

- Shows a summary of the RDS processes that support the DB instance, for example aurora for Amazon Aurora DB clusters and mysqld for MySQL DB instances.
- Process threads appear nested beneath the parent process.
- Process threads show CPU utilization only as other metrics are the same for all threads for the process.
- The console displays a maximum of 100 processes and threads.
- The results are a combination of the top CPU consuming and memory consuming processes and threads.

- If there are more than 50 processes and more than 50 threads, the console displays the top 50 consumers in each category. This display helps you identify which processes are having the greatest impact on performance.

RDS processes –

- Shows a summary of the resources used by the RDS management agent, diagnostics monitoring processes, and other AWS processes that are required to support RDS DB instances.

OS processes –

- Shows a summary of the kernel and system processes, which generally have minimal impact on performance.

References:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/rds-metricscollected.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.OS.html#USER_Monitoring.OS.CloudWatchLogs

Check out this Amazon CloudWatch Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/

Check out this Amazon RDS Cheat Sheet:

https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/

AWS - RDS Recommended Practices

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Recommendations.html

MYSQL - RDS recommendations:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MySQL.html

- Amazon RDS fully supports the InnoDB storage engine for MySQL DB instances
- You must be running an instance of MySQL 5.6 or later to use the InnoDB memcached interface
- Federated Storage Engine is currently not supported by Amazon RDS for MySQL
- For user-created schemas,
  - the MyISAM storage engine does not support reliable recovery and can result in lost or corrupt data when MySQL is restarted after a recovery, preventing Point-In-Time restore or snapshot restore from working as intended.
- However, if you still choose to use MyISAM with Amazon RDS, snapshots can be helpful under some conditions.