**Answer Sheet**

**Q1) A financial services company runs an Amazon RDS for SQL Server instance farm. The most critical RDS SQL Server DB instance recently experienced a 100% CPU spike, causing a significant incident and impacting the users. The Database Manager needs to determine the root cause, but could not succeed in determining it using the standard metrics collected.**
**Which steps should the Database Manager take to investigate the database processes running during an increase in CPU load? (Select TWO.)**

○ Look at Amazon RDS Recommendations to view and implement insights on the database resources and performance data.
**Explanation:-**This option is incorrect. Although RDS Recommendations will help implement best practices, it will not help identify the root cause of the incident.
○ Use Amazon QuickSight to perform ad-hoc analysis on the processes running.
**Explanation:-**This option is incorrect. QuickSight is designed for business intelligence requirements, not performance tuning.
✓ Use Amazon RDS Performance Insights to assess the database load and review waits and SQL statements.
**Explanation:-**Monitoring is an integral part of maintaining the reliability, availability, and performance of Amazon RDS and your AWS solutions. Two of the recommended RDS monitoring tools are:
Amazon RDS Enhanced Monitoring — provides metrics in real-time for the operating system (OS) that your DB instance runs on. CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance, and Enhanced Monitoring collects 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 more significant if your DB instances use lower 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.
Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. The central metric for Performance Insights is DB Load, representing the average number of active sessions for the DB engine. The DB Load metric is collected every second. An active session is a connection that has submitted work to the DB engine and is waiting for a response. For example, if you submit a SQL query to the DB engine, the database session is active while the DB engine is processing the query.
A wait event causes an SQL statement to wait for a specific event to happen before it can continue running. For example, a SQL statement might wait until a locked resource is unlocked. By combining DB Load with wait events, you can get a complete picture of the session state.
Given that standard metrics were insufficient, the question checks which tools will directly provide details on the CPU utilization of the RDS DB instance, including detailed data on database transactions, processes, SQL statements, and database wait events. Among the choices, the best answers are:
- Review the Enhanced Monitoring metrics to view CPU utilization at the DB instance level.
- Use Amazon RDS Performance Insights to assess the database load and review waits and SQL statements.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.OS.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.Overview.html
✓ Review the Enhanced Monitoring metrics to view CPU utilization at the DB instance level.
**Explanation:-**Monitoring is an integral part of maintaining the reliability, availability, and performance of Amazon RDS and your AWS solutions. Two of the recommended RDS monitoring tools are:
Amazon RDS Enhanced Monitoring — provides metrics in real-time for the operating system (OS) that your DB instance runs on. CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance, and Enhanced Monitoring collects 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 more significant if your DB instances use lower 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.
Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. The central metric for Performance Insights is DB Load, representing the average number of active sessions for the DB engine. The DB Load metric is collected every second. An active session is a connection that has submitted work to the DB engine and is waiting for a response. For example, if you submit a SQL query to the DB engine, the database session is active while the DB engine is processing the query.
A wait event causes an SQL statement to wait for a specific event to happen before it can continue running. For example, a SQL statement might wait until a locked resource is unlocked. By combining DB Load with wait events, you can get a complete picture of the session state.
Given that standard metrics were insufficient, the question checks which tools will directly provide details on the CPU utilization of the RDS DB instance, including detailed data on database transactions, processes, SQL statements, and database wait events. Among the choices, the best answers are:
- Review the Enhanced Monitoring metrics to view CPU utilization at the DB instance level.
- Use Amazon RDS Performance Insights to assess the database load and review waits and SQL statements.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Monitoring.OS.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.Overview.html
○ Check Amazon CloudWatch Events and review the CPU-intensive database transactions.
**Explanation:-**This option is incorrect. CloudWatch Events provides a high-level system and application events. Compared to Enhanced Monitoring, it will not provide more information on what is running in the SQL Server DB instance.

---

**Q2) A Database Specialist needs to deploy a new Amazon Aurora MySQL database cluster. The Application Manager wants to monitor the database performance and asks the Specialist if Amazon RDS Performance Insights can be enabled.**
**Which steps should the Database Specialist consider to enable Amazon RDS Performance Insights? (Select TWO).**

✓ The AWS account used to build the cluster is granted proper AWS KMS key and IAM policies.
**Explanation:-**Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. To use Performance Insights, you must enable it on your DB instance. Enabling and disabling Performance Insights does not cause downtime, a reboot, or a failover. You have the following options when you choose to Enable Performance Insights:
1) Retention – The amount of time to retain Performance Insights data. Choose either seven days (the default) or two years.
2) Master key – Specify your AWS Key Management Service (AWS KMS) key. Performance Insights encrypts all potentially sensitive data using your AWS KMS key. Data is encrypted in transit and at rest.
References:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.Enabling.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.UsingDashboard.htm

✅ Amazon RDS Performance Insights is configured on a per-instance level.

**Explanation:-**Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users. To use Performance Insights, you must enable it on your DB instance. Enabling and disabling Performance Insights does not cause downtime, a reboot, or a failover. You have the following options when you choose to Enable Performance Insights:

1) Retention – The amount of time to retain Performance Insights data. Choose either seven days (the default) or two years.

2) Master key – Specify your AWS Key Management Service (AWS KMS) key. Performance Insights encrypts all potentially sensitive data using your AWS KMS key. Data is encrypted in transit and at rest.

References:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.Enabling.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.UsingDashboard.htm

⚪ Performance Insights data retention can be set from a day to a year (365 days).

**Explanation:-**This option is incorrect because, currently, Performance Insights data retention can be set between default (7 days) and long-term retention (2 years).

⚪ Amazon RDS Performance Insights is configured on the cluster level.

**Explanation:-**This option is incorrect because this monitoring feature is configured on every instance.

⚪ Enhanced Monitoring should be enabled before Performance Insights.

**Explanation:-**This option is incorrect because Performance Insights can be activated without enabling Enhanced Monitoring.

---

**Q3) A Database Specialist creates an Amazon RDS DB instance from a snapshot. The new database will be used in the staging environment for developing new features for a web application. After updating the server configuration with the new DB instance endpoint, the web application still can't access the restored RDS instance. The initial investigation shows that the database credentials used are correct.**
**Which of the following is the most likely cause of this connection problem?**

✅ Amazon RDS initially configures the restored DB instance to use the default security group.

**Explanation:-**When you restore a DB instance from a DB snapshot, the default DB parameter and default security group are associated with the restored instance. That association means that the default security group does not allow access to the DB instance, and no custom parameter settings are available in the default parameter group. You need to retain the DB parameter group and security group associated with the DB instance that was used to create the DB snapshot.

Since the server configuration already uses the new DB instance's endpoint, it can be deduced that the connection problem is due to the security group's incorrect settings.

References:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.RestoringFromSnapshot.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html

⚪ The restored DB instance must be restarted first before it can be used.

**Explanation:-**This optoin is incorrect because the DB instance can be used as soon as its status becomes available.

⚪ The restored DB instance is corrupted.

**Explanation:-**This option is incorrect because this is highly unlikely to happen. Amazon RDS is a managed database service, and one of its selling points is its enhanced reliability for critical production databases, including automated backups, database snapshots, and automatic host replacement.

⚪ The Reboot with failover option is enabled.

**Explanation:-**This option is incorrect since this is only applicable for RDS Multi-AZ deployments. This option is primarily used to simulate a DB failure in the primary DB instance and to verify the failover process.

---

**Q4) A Database Specialist is adding new indexes and altering large tables of an Amazon Aurora PostgreSQL database that uses a medium instance type with a default configuration. The application suddenly crashes with the following error message when loading a large dataset:**
**ERROR: could not write block 18980612 of temporary file: No space left on device**
**Which of the following can the Database Specialist do to resolve this issue? (Select TWO.)**

✅ Modify the Aurora database to use a DB instance class with more local SSD storage.

**Explanation:-**Each DB instance in an Amazon Aurora DB cluster uses local solid-state drive (SSD) storage to store temporary tables for a session. This local storage for temporary tables doesn't automatically grow like the Aurora cluster volume. Instead, the amount of local storage is limited. The limit is based on the DB instance class for DB instances in your DB cluster.

Instances in Aurora clusters have two types of storage:

Storage for persistent data (called the cluster volume). This storage type increases automatically when more space is required.

Local storage for each Aurora instance in the cluster, based on the instance class. This storage type and size is bound to the instance class, and can be changed only by moving to a larger DB instance class. Aurora for MySQL uses local storage for storing error logs, general logs, slow query logs, audit logs, and non-InnoDB temporary tables.

To show the amount of storage available for temporary tables and logs, you can use the CloudWatch metric FreeLocalStorage. This metric is for per-instance temporary volumes, not the cluster volume. In some cases, you can't modify your workload to reduce the amount of temporary storage required. If so, you have to modify your DB instances to use a DB instance class that has more local SSD storage.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/postgresql-aurora-storage-issue/
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/CHAP_Troubleshooting.html#CHAP_Troubleshooting.Aurora.NoSpaceLeft
https://docs.amazonaws.cn/en_us/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.BestPractices.html#AuroraPostgreSQL.BestPractices.TroubleshootingStorage

✅ Lessen the database workload to reduce the amount of temporary storage required.

**Explanation:-**Each DB instance in an Amazon Aurora DB cluster uses local solid-state drive (SSD) storage to store temporary tables for a session. This local storage for temporary tables doesn't automatically grow like the Aurora cluster volume. Instead, the amount of local storage is limited. The limit is based on the DB instance class for DB instances in your DB cluster.

Instances in Aurora clusters have two types of storage:

Storage for persistent data (called the cluster volume). This storage type increases automatically when more space is required.

Local storage for each Aurora instance in the cluster, based on the instance class. This storage type and size is bound to the instance class, and can be changed only by moving to a larger DB instance class. Aurora for MySQL uses local storage for storing error logs, general logs, slow query logs, audit logs, and non-InnoDB temporary tables.

To show the amount of storage available for temporary tables and logs, you can use the CloudWatch metric FreeLocalStorage. This metric is for per-instance temporary volumes, not the cluster volume. In some cases, you can't modify your workload to reduce the amount of temporary storage

required. If so, you have to modify your DB instances to use a DB instance class that has more local SSD storage.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/postgresql-aurora-storage-issue/
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/CHAP_Troubleshooting.html#CHAP_Troubleshooting.Aurora.NoSpaceLeft
https://docs.amazonaws.cn/en_us/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.BestPractices.html#AuroraPostgreSQL.BestPractices.TroubleshootingStorage

- Add an Auto Scaling policy to the Aurora database which will automatically increase the local storage.

**Explanation:-**This option is incorrect because the root cause of this issue is the lack of free local storage of the Aurora database and not the lack of an Auto Scaling policy. By default, the Amazon Aurora storage automatically scales with the data in your cluster volume, which is why it doesn't need to have a new Auto Scaling policy. What it needs is a larger instance type to increase its local SSD storage.

- Wait for a few minutes until Amazon Aurora automatically scales out the cluster volume storage and then reload the datasets once again.

**Explanation:-**This option is incorrect because the error message pertains to the lack of available disk space in the local storage and not on the cluster volume. The cluster volume is a type of storage that increases automatically when more space is required.

- Enable local storage scaling on the Amazon Aurora database, which is disabled by default.

**Explanation:-**This option is incorrect because the amount of local storage is limited and can't be automatically scaled. Amazon Aurora also does not have a feature that you can enable to scale the local storage automatically.

---

**Q5) A company has an application that uses an Amazon RDS MySQL DB instance with a default DB parameter group. The database abruptly stopped working, affecting the production workloads. Upon investigation, the database is in STORAGE_FULL state and doesn't have enough space to perform basic operations.**
**What should the Database Specialist do first to quickly fix this issue?**

- ✅ Increase the allocated storage property of the RDS DB instance.

**Explanation:-**An Amazon RDS DB instance in the STORAGE_FULL state doesn't have enough available space to perform basic operations, such as connecting to or restarting the instance. To resolve this issue, follow these steps:
Confirm that the DB instance status is STORAGE_FULL.
Add more storage space to the instance.
Increase the allocated storage property of your DB instance.
If the DB instance is in a STORAGE_FULL state, the instance accepts only allocated storage modifications. Any modifications to other values are rejected.
When the DB instance is in a storage-optimization status, the instance is operational, but you can't make other storage modifications for six hours or until the DB instance's status is no longer storage-optimization. In most cases, a small increase to the Allocated Storage allows you to reconnect to the instance so that you can perform additional troubleshooting
If your workload is unpredictable, you can enable storage autoscaling for an Amazon RDS DB instance. To do so, you can use the Amazon RDS console, the Amazon RDS API, or the AWS CLI. With storage autoscaling enabled, when Amazon RDS detects that you are running out of free database space it automatically scales up your storage.
Take note that the RDS storage autoscaling feature can't completely prevent storage-full situations for large data loads because further storage modifications can't be made until six hours after storage optimization has completed on the instance. If you perform a large data load, and autoscaling doesn't provide enough space, the database might remain in the storage-full state for several hours.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/rds-out-of-storage/
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html#USER_PIOPS.Autoscaling

- Upgrade the value of the allocated storage property in the default DB parameter group.

**Explanation:-**This option is incorrect because you can't modify a default DB parameter group.

- Enable storage autoscaling for the Amazon RDS DB instance.

**Explanation:-**This option is incorrect because the DB instance only accepts allocated storage modifications if the database is in STORAGE_FULL state. In this scenario, you have to increase the allocated storage property of the RDS DB instance first before you can enable storage autoscaling. This feature can't completely prevent storage-full situations because further storage modifications can't be made until six hours after the storage optimization has completed.

- Restart the RDS DB Instance.

**Explanation:-**This option is incorrect because if the RDS database is in STORAGE_FULL state, you can neither connect to nor restart the database instance.

---

**Q6) A mobile game app uses Amazon Cognito for user authentication and stores the user data and game scores on an Amazon DynamDB table. The DynamoDB Streams is enabled on the table, which allows another application to capture the data updates from the stream to provide a near-real-time leaderboard. A database error occurred, which required the DynamoDB table to be restored from a recent backup. The DynamoDB Streams setting of the table was missing after restoring it, causing the leaderboard to contain outdated data.**
**What is the root cause of the issue?**

- The restoration process failed due to the lack of provisioned write capacity of the source table.

**Explanation:-**This option is incorrect because the restore operation is fully managed and controlled by AWS on the backend. Therefore, restoring from a backup does not require a specific write capacity.

- DynamoDB doesn't include the primary partition key and the sort key of the source table to the destination table.

**Explanation:-**This option is incorrect. When you restore a table from a backup, the values of the primary partition key and the sort key of the source table will be carried over to the destination table.

- ✅ DynamoDB doesn't include the existing Stream settings of the source table to the destination table.

**Explanation:-**When you create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes.
When you do a restore, you can change the following table settings:
Global secondary indexes (GSIs)
Local secondary indexes (LSIs)
Billing mode
Provisioned read and write capacity
Encryption settings
However, some settings are not carried over on the restored table and you must manually configure them after restoring.
You must manually set up the following on the restored table:
Auto scaling policies
AWS Identity and Access Management (IAM) policies
Amazon CloudWatch metrics and alarms

Tags
Stream settings
Time to Live (TTL) settings
Since the application uses DynamoDB Streams, the Stream settings of the source table must be configured on the restored table.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Backup.Tutorial.html

- The new DynamoDB table was restored from an On-demand backup instead of a Point-In-Time-Recovery backup.

**Explanation:-**This option is incorrect because both of these backup features can be used to restore a DynamoDB table from a backup. Whichever backup feature you use, you still need to configure the stream settings on the restored table.

---

**Q7) A Database Specialist needs to monitor its Amazon Aurora PostgreSQL database cluster activity in real-time to meet compliance and regulatory requirement using a third-party monitoring tool. The database cluster must push an audit log containing the activity event record to the monitoring tool every time a user or an application connects to the database. The audit logs should always be encrypted to ensure data security.**
**Which approach can satisfy the above requirements?**

- Configure the Amazon RDS Event notification to send the encrypted audit logs to an Amazon Kinesis stream. Integrate the third-party monitoring tool and the Kinesis stream to monitor the activity stream of the Aurora PostgreSQL database in real-time.

**Explanation:-**This option is incorrect because RDS Events only provide notification via Amazon SNS when a specific Amazon RDS event occurs. The information it collects doesn't include encrypted audit logs and its notifications are not in real-time.

- Enable Enhanced Monitoring in the Aurora PostgreSQL database cluster. Integrate the third-party monitoring tool and the Amazon Aurora to monitor the activity stream in real-time.

**Explanation:-**This option is incorrect because Enhanced Monitoring simply provides metrics in real-time for the operating system (OS) that your DB instance runs on. It is not capable of sending encrypted audit logs to an external tool. You have to use database activity streams instead.

- Create an Amazon RDS Proxy for the Aurora PostgreSQL cluster. Integrate the third-party monitoring tool with the RDS Proxy to monitor the activity stream in real-time.

**Explanation:-**This option is incorrect because RDS Proxy just allows applications to pool and share connections established with the database to improve database efficiency and application scalability. This feature can't be used to send the encrypted audit logs of an Aurora PostgreSQL cluster to an Amazon Kinesis stream that will be used by the third-party monitoring tool.

- ✅ Set up database activity streams in the Aurora PostgreSQL cluster that will automatically send the encrypted audit logs to an Amazon Kinesis stream. Integrate the third-party monitoring tool and the Kinesis stream to monitor the activity stream in real-time.

**Explanation:-**Beyond external security threats, managed databases need to provide protection against insider risks from database administrators (DBAs). Database Activity Streams, currently supported for Amazon Aurora, provides a real-time data stream of the database activity in your relational database. When integrated with 3rd party database activity monitoring tools, you can monitor and audit database activity to provide safeguards for your database and meet compliance and regulatory requirements.
Database Activity Streams protects your database from internal threats by implementing a protection model that controls DBA access to the database activity stream. Thus the collection, transmission, storage, and subsequent processing of the database activity stream is beyond the access of the DBAs that manage the database.
The stream is pushed to an Amazon Kinesis data stream that is created on behalf of your database. From Kinesis Data Firehose, the database activity stream can then be consumed by Amazon CloudWatch or by partner applications for compliance management such as McAfee's Data Center Security Suite, or IBM Security Guardium. These partner applications can use the database activity stream information to generate alerts and provide auditing of all activity on your Amazon Aurora database.
Database activity streams require the use of AWS Key Management Service (AWS KMS). AWS KMS is required because the activity streams are always encrypted.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/DBActivityStreams.html
https://aws.amazon.com/about-aws/whats-new/2019/05/amazon-aurora-with-postgresql-compatibility-supports-database-activity-streams/
https://aws.amazon.com/rds/features/security/#Database_Activity_Streams

---

**Q8) A serverless online bidding service uses a DynamoDB table as its database. The application has predictable and consistent traffic that ramps gradually. It heavily relies on the UpdateItem operation to immediately refresh the current item's price. The application's workload peaks every weekday, one hour before the bid closes. During this time, some of the requests encounter a ProvisionedThroughputExceededException error that causes the item to close at a lower price.**
**Which of the following solutions should the Database Specialist do to resolve the issue?**

- Set the DynamoDB table to use the On-Demand mode with Reserved Capacity.

**Explanation:-**This option is incorrect because the On-demand Mode is used for unpredictable traffic with unknown workloads. Since the peak load time is given in the question, using the Provisioned Mode is the better option.

- Use DynamoDB Accelerator (DAX).

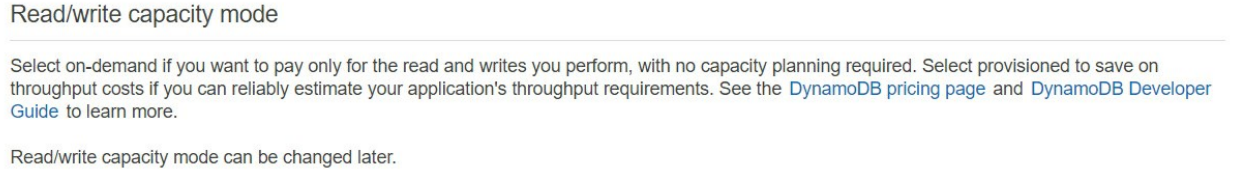
**Explanation:-**This option is incorrect. Although DAX can improve your database performance, it is mainly used for read-heavy workloads that need millisecond latency. In the question, it is given that the application is write-intensive since it heavily relies on the UpdateItem API, which is a write operation.

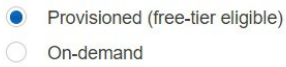
- Use DynamoDB Global table.

**Explanation:-**This option is incorrect because a Global table is primarily used to facilitate low network latency and disaster recovery for your application. You might improve your network throughput but it doesn't mean that you'll also increase the write throughput of your database.

- ✅ Modify the DynamoDB table to use provisioned mode. Increase the allocated write capacity units (WCUs).

**Explanation:-**
Amazon DynamoDB has two read/write capacity modes for processing reads and writes on your tables:
On-demand
Provisioned (default, free-tier eligible)
Amazon DynamoDB on-demand is a flexible billing option capable of serving thousands of requests per second without capacity planning. DynamoDB on-demand offers pay-per-request pricing for read and write requests so that you pay only for what you use.






On-demand mode is a good option if any of the following are true:
You create new tables with unknown workloads.
You have unpredictable application traffic.
You prefer the ease of paying for only what you use.
For provisioned mode tables, you specify throughput capacity in terms of read capacity units (RCUs) and write capacity units (WCUs):
The provisioned mode is a good option if any of the following are true:
You have predictable application traffic.
You run applications whose traffic is consistent or ramps gradually.
You can forecast capacity requirements to control costs.
A ProvisionedThroughputExceededException error means that you've exceeded your maximum allowed provisioned throughput for a table or for one or more global secondary indexes. You can resolve this issue by increasing the throughput of your DynamoDB table.
Since the peak load time is known in the given scenario, using the Provisioned mode is a better choice than On-demand mode. You can manually specify the minimum and maximum provisioned capacity for reads and writes on the AWS DynamoDB Console.


References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Programming.Errors.html

---

**Q9) A Database Specialist needs to migrate an on-premises Microsoft SQL Server database to AWS. Various applications in the production environment use the database so the migration must not cause any downtime. The Database Specialist must perform a full data load to the target database while also capturing the recent changes from the source via change data capture (CDC).**
**In this setup, how does AWS DMS reads the ongoing changes from the source database?**

✅ AWS DMS uses the fn_dblog() or fn_dump_dblog() function in SQL Server to read the changes in the transaction log based on the log sequence number (LSN).
**Explanation:-**You can create an AWS DMS task that captures ongoing changes to the source data store. You can do this capture while you are migrating your data. You can also create a task that captures ongoing changes after you complete your initial (full-load) migration to a supported target data store. This process is called ongoing replication or change data capture (CDC). AWS DMS uses this process when replicating ongoing changes from a source data store. This process works by collecting changes to the database logs using the database engine's native API.
Each source engine has specific configuration requirements for exposing this change stream to a given user account. Most engines require some additional configuration to make it possible for the capture process to consume the change data in a meaningful way, without data loss. For example, Oracle requires the addition of supplemental logging, and MySQL requires row-level binary logging (bin logging). To read ongoing changes from the source database, AWS DMS uses engine-specific API actions to read changes from the source engine's transaction logs.
To read ongoing changes from the source database, AWS DMS uses engine-specific API actions to read changes from the source engine's transaction logs. Following are some examples of how AWS DMS does that:
For Oracle - AWS DMS uses either the Oracle LogMiner API or binary reader API (bfile API) to read ongoing changes. AWS DMS reads ongoing changes from the online or archive redo logs based on the system change number (SCN).
For Microsoft SQL Server - AWS DMS uses MS-Replication or MS-CDC to write information to the SQL Server transaction log. It then uses the fn_dblog() or fn_dump_dblog() function in SQL Server to read the changes in the transaction log based on the log sequence number (LSN).
For MySQL - AWS DMS reads changes from the row-based binary logs (binlogs) and migrates those changes to the target.
For PostgreSQL - AWS DMS sets up logical replication slots and uses the test_decoding plugin to read changes from the source and migrate them to the target.
For Amazon RDS as a source - AWS recommends ensuring that backups are enabled to set up CDC. It is also recommended to ensure that the source database is configured to retain change logs for a sufficient time—24 hours is usually enough.
AWS DMS uses a replication instance to connect to your source data store, read the source data, and format the data for consumption by the target data store. A replication instance also loads the data into the target data store. Most of this processing happens in memory. However, large transactions might require some buffering on disk. Cached transactions and log files are also written to disk.
AWS DMS reads changes from the row-based binary logs (binlogs) and migrates those changes to the target.
**Explanation:-**This option is incorrect because this is the behavior of AWS DMS for reading changes from a MySQL database.
AWS DMS reads ongoing changes from the binary reader API (bfile API) based on the system change number (SCN).
**Explanation:-**This option is incorrect because this is only true for an Oracle database. AWS DMS uses either the Oracle LogMiner API or binary reader API (bfile API) to read ongoing changes. It also reads ongoing changes from the online or archived redo logs based on the system change number (SCN).
AWS DMS sets up logical replication slots and uses the test_decoding plugin to read changes from the source and migrate them to the target.
**Explanation:-**This option is incorrect because this is the behavior of AWS DMS for reading changes from a PostgreSQL database. Microsoft SQL Server doesn't use the test_decoding plugin for change data capture.

---

**Q10) A Database Specialist needs to integrate an on-premises corporate Active Directory (AD) to an Amazon RDS for SQL Server DB instance. Employees must use their existing AD credentials to connect to the RDS SQL Server database.**
**What should the Database Specialist implement to satisfy this requirement?**

Launch an Active Directory Connector and directly integrate it with the on-premises Active Directory. Set up Windows Authentication to the Amazon RDS for SQL Server DB instance using the Active Directory Connector.

**Explanation:-**This option is incorrect because using the Active Directory Connector alone is not enough to set up Windows Authentication to your SQL Server database. You have to launch an AWS Managed Microsoft AD with a trust relationship with the on-premises Active Directory to meet the specified requirement.

Launch an AWS Managed Microsoft AD and set up Windows Authentication by enabling Transparent Data Encryption (TDE) in the Amazon RDS for SQL Server DB instance. Modify the default security group of the domain controllers to restrict unauthorized access.

**Explanation:-**This option is incorrect because enabling TDE will just encrypt the data and connection in transit. This approach will not enable the employees to use their existing AD credentials to connect to the SQL Server database.

Launch an AWS Managed Microsoft AD and set up Windows Authentication by establishing a trust relationship with the on-premises Active Directory via a forest trust. Configure the security group of the domain controllers to restrict unauthorized access.

**Explanation:-**You can use Microsoft Windows Authentication to authenticate users when they connect to your Amazon RDS for Microsoft SQL Server DB instance. The DB instance works with AWS Directory Service for Microsoft Active Directory, also called AWS Managed Microsoft AD, to enable Windows Authentication. When users authenticate with a SQL Server DB instance joined to the trusting domain, authentication requests are forwarded to the domain directory that you create with AWS Directory Service.

Amazon RDS uses mixed mode for Windows Authentication. This approach means that the master user (the name and password used to create your SQL Server DB instance) uses SQL Authentication. Because the master user account is a privileged credential, you should restrict access to this account. To get Windows Authentication using an on-premises or self-hosted Microsoft Active Directory, create a forest trust.

You can configure one and two-way external and forest trust relationships between your AWS Directory Service for Microsoft Active Directory and on-premises directories, as well as between multiple AWS Managed Microsoft AD directories in the AWS cloud. AWS Managed Microsoft AD supports all three trust relationship directions: Incoming, Outgoing, and Two-way (Bi-directional).

AWS Directory Service creates a fully managed, Microsoft Active Directory in the AWS Cloud. When you create an AWS Managed Microsoft AD directory, AWS Directory Service creates two domain controllers and Domain Name Service (DNS) servers on your behalf. The directory servers are created in two subnets in two different Availability Zones within a VPC. This redundancy helps ensure that your directory remains accessible even if a failure occurs.

Launch an AWS Managed Microsoft AD and set up Windows Authentication where the directory and the DB instance are in the same VPC. Enable cross-VPC traffic to establish a trust relationship with the on-premises Active Directory.

**Explanation:-**This option is incorrect because enabling cross-VPC traffic is not applicable if the directory and the DB instance are in the same VPC. Furthermore, establishing a trust relationship is done via Active Directory and not through cross-VPC traffic configuration.

---

**Q11) A Database Specialist is planning to migrate an on-premises 200-GB Oracle database to an AWS RDS DB instance. The application requires a database environment that can accommodate OLTP workload requirements with high volumes of read operations. The specialist wants to ensure that the instance used will always have enough bandwidth to support the IO throughput requirement.**
**What is the most cost-effective configuration that meets these requirements? (SELECT THREE)**

Configure the Amazon RDS DB instance to use Magnetic Storage.

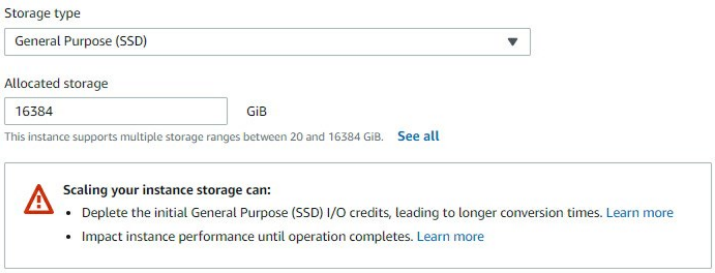
**Explanation:-**This option is incorrect because it is used for backward compatibility and does not configure the solution for future IO needs.

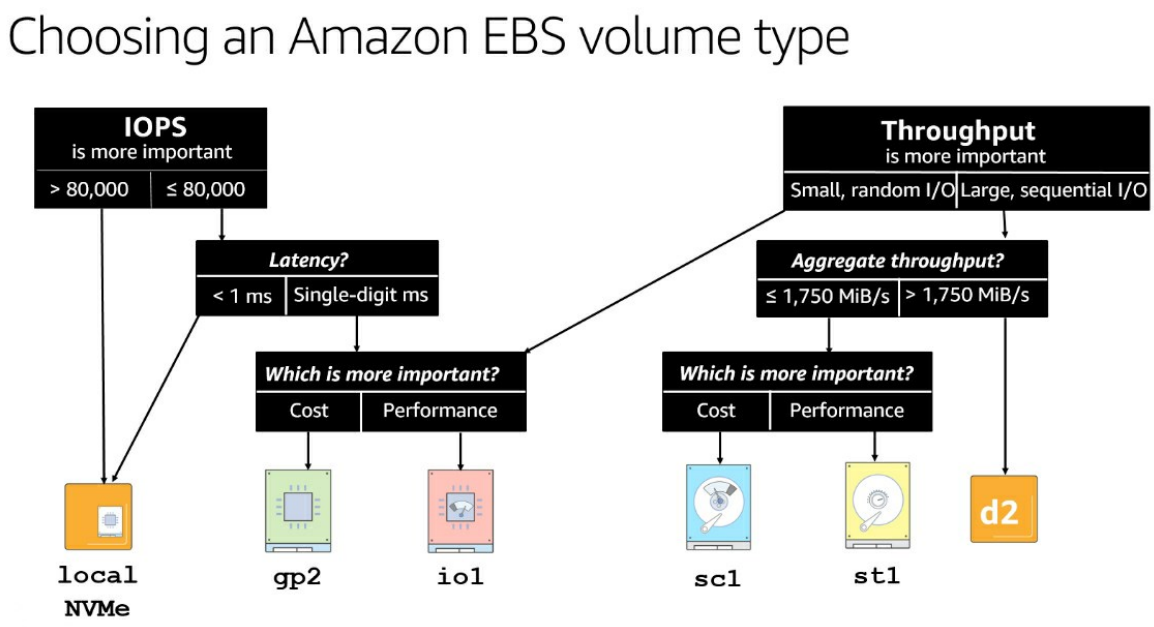
Configure the Amazon RDS DB instance to use gp2 SSD.

**Explanation:-**

Amazon RDS provides three storage types: General Purpose SSD (also known as gp2), Provisioned IOPS SSD (also known as io1), and magnetic (also known as standard). They differ in performance characteristics and price.

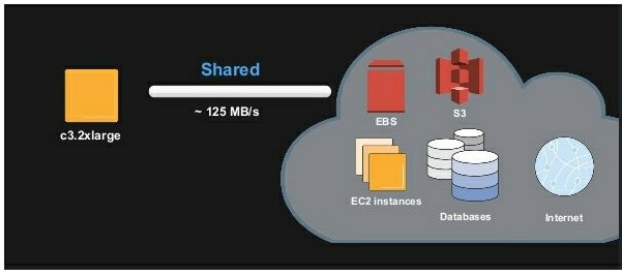
General Purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods. Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads that require low I/O latency and consistent I/O throughput. Amazon RDS also supports magnetic storage for backward compatibility.




An Amazon EBS–optimized instance uses an optimized configuration stack and provides additional, dedicated capacity for Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance. EBS–optimized instances deliver dedicated bandwidth to Amazon EBS. When attached to an EBS–optimized instance, General Purpose

SSD (gp2) volumes are designed to deliver baseline and burst performance 99% of the time, and Provisioned IOPS SSD (io1) volumes are designed to deliver their provisioned performance 99.9% of the time.




Amazon RDS uses the Oracle DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. The source DB instance becomes the primary DB instance. Updates made to the primary DB instance are asynchronously copied to the read replica. You can reduce the load on your primary DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
In this scenario, the specialist needs to meet three Amazon RDS requirements: 1) most cost-effective configuration, 2) OLTP application with high volumes of read operations, and 3) enough bandwidth to support the IO throughput requirement. Since no information was specified on IO latency requirements, it is safe to assume that the cost-effective general purpose SSD configuration is sufficient.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html

✅ Configure an Amazon RDS Read replica.
**Explanation:-**Amazon RDS provides three storage types: General Purpose SSD (also known as gp2), Provisioned IOPS SSD (also known as io1), and magnetic (also known as standard). They differ in performance characteristics and price.
General Purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods. Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads that require low I/O latency and consistent I/O throughput. Amazon RDS also supports magnetic storage for backward compatibility.
An Amazon EBS–optimized instance uses an optimized configuration stack and provides additional, dedicated capacity for Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance. EBS–optimized instances deliver dedicated bandwidth to Amazon EBS. When attached to an EBS–optimized instance, General Purpose SSD (gp2) volumes are designed to deliver baseline and burst performance 99% of the time, and Provisioned IOPS SSD (io1) volumes are designed to deliver their provisioned performance 99.9% of the time.
Amazon RDS uses the Oracle DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. The source DB instance becomes the primary DB instance. Updates made to the primary DB instance are asynchronously copied to the read replica. You can reduce the load on your primary DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
In this scenario, the specialist needs to meet three Amazon RDS requirements: 1) most cost-effective configuration, 2) OLTP application with high volumes of read operations, and 3) enough bandwidth to support the IO throughput requirement. Since no information was specified on IO latency requirements, it is safe to assume that the cost-effective general purpose SSD configuration is sufficient.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html
⚪ Configure the application to connect to Multi-AZ standby replica for operations.
**Explanation:-**This option is incorrect because the standby replica created, when Multi-AZ configuration is enabled, is inaccessible for users while the primary DB instance is available. It is designed for high availability and not read scalability.
✅ Use an EBS-Optimized instance for the Amazon RDS DB instance.
**Explanation:-**Amazon RDS provides three storage types: General Purpose SSD (also known as gp2), Provisioned IOPS SSD (also known as io1), and magnetic (also known as standard). They differ in performance characteristics and price.
General Purpose SSD volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods. Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads that require low I/O latency and consistent I/O throughput. Amazon RDS also supports magnetic storage for backward compatibility.
An Amazon EBS–optimized instance uses an optimized configuration stack and provides additional, dedicated capacity for Amazon EBS I/O. This optimization provides the best performance for your EBS volumes by minimizing contention between Amazon EBS I/O and other traffic from your instance. EBS–optimized instances deliver dedicated bandwidth to Amazon EBS. When attached to an EBS–optimized instance, General Purpose SSD (gp2) volumes are designed to deliver baseline and burst performance 99% of the time, and Provisioned IOPS SSD (io1) volumes are designed to deliver their provisioned performance 99.9% of the time.
Amazon RDS uses the Oracle DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. The source DB instance becomes the primary DB instance. Updates made to the primary DB instance are asynchronously copied to the read replica. You can reduce the load on your primary DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
In this scenario, the specialist needs to meet three Amazon RDS requirements: 1) most cost-effective configuration, 2) OLTP application with high volumes of read operations, and 3) enough bandwidth to support the IO throughput requirement. Since no information was specified on IO latency requirements, it is safe to assume that the cost-effective general purpose SSD configuration is sufficient.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html
⚪ Configure the Amazon RDS DB instance to use io1 SSD.

---

**Q12) A Database Specialist is planning to migrate the on-premises data warehouse to an Amazon Redshift cluster. A Reporting Manager wants to ensure that the group's critical queries, especially the short-running ones, get prioritized over less critical and more complex queries. However, the Database Specialist wants to minimize query management and avoid any additional costs.**
**Which steps should the Database Specialist take to meet the requirements?**

Enable automatic workload management in the Redshift cluster and assign the group's queries to a queue with the HIGHEST priority. Enable short query acceleration and concurrency scaling.

**Explanation:-**This option is incorrect because concurrency scaling will incur additional costs if utilized.

Enable automatic workload management in the Redshift cluster and assign less critical and more complex queries to a queue with the LOWEST priority. Enable concurrency scaling.

**Explanation:-**This option is incorrect because this solution does not ensure the execution priority of the critical and short-running queries. Furthermore, concurrency scaling will incur additional costs if utilized.

Enable manual workload management in the Redshift cluster and assign both the critical and the short-running queries to a queue with a higher slot count than other queues.

**Explanation:-**This option is incorrect because manual workload management could negatively impact the performance if it is not managed properly. This requires some effort to finetune.

Enable automatic workload management in the Redshift cluster and assign the group's queries to a queue with the HIGHEST priority. Enable short query acceleration.

**Explanation:-**Amazon Redshift now makes it easy to maximize query throughput and get consistent performance for your most demanding analytics workloads. Automatic workload management (WLM) uses machine learning to dynamically manage memory and concurrency helping maximize query throughput. In addition, you can now easily set the priority of your most important queries, even when hundreds of queries are being submitted.

By setting query priorities, you can now ensure that higher priority workloads get preferential treatment in Redshift including more resources during busy times for consistent query performance. Automatic WLM uses intelligent algorithms to make sure that lower priority queries don't stall, but continue to make progress.

Short query acceleration (SQA) prioritizes selected short-running queries ahead of longer-running queries. SQA runs short-running queries in a dedicated space so that SQA queries aren't forced to wait in queues behind longer queries. SQA only prioritizes queries that are short-running and are in a user-defined queue. With SQA, short-running queries begin running more quickly and users see results sooner.

With the Concurrency Scaling feature, you can support virtually unlimited concurrent users and concurrent queries, with consistently fast query performance. However, you are charged for concurrency scaling clusters for the time they are in use.

---

**Q13) A popular multiplayer online game is using an Amazon DynamoDB table named GameScore to track users' scores. The table is configured with a partition key UserId and a sort key GameTitle as shown in the diagram below:**
**A Database Specialist has been tasked to work with a developer to add a leaderboard feature to display the top scores for each game as well as the top scorers. The feature will query data over the entire table and across all partitions.**
**What must be done to meet this requirement?**

Add a Global Secondary Index (GSI) for the new feature.

**Explanation:-**DynamoDB supports two types of secondary indexes:

Global secondary index — an index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions.

Local secondary index — an index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.

To speed up queries on non-key attributes, you can create a global secondary index. A global secondary index contains a selection of attributes from the base table, but they are organized by a primary key that is different from that of the table. The index key does not need to have any of the key attributes from the table; it doesn't even need to have the same key schema as a table.

In this scenario, you could create a global secondary index named GameTitleIndex, with a partition key of GameTitle and a sort key of TopScore. Since the base table's primary key attributes are always projected into an index, the UserId attribute is also present. The diagram above shows what GameTitleIndex index would look like.

References:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html

Add a sparse index for the new feature.

**Explanation:-**This option is incorrect because parse indexes are only useful for queries over a small subsection of a table. For any item in a table, DynamoDB writes a corresponding index entry only if the index sort key value is present in the item. If the sort key doesn't appear in every table item, the index is said to be "sparse".

Launch a DynamoDB global table to query data across all partitions.

**Explanation:-**This option is incorrect because this is not a suitable use case for DynamoDB global table. You can use a global secondary index to query data across all partitions. Amazon DynamoDB global tables provide a fully managed solution for deploying a multi-region, multi-master database, without having to build and maintain your own replication solution.

Create a Local Secondary Index (LSI) in the existing table for the new feature.

**Explanation:-**This option is incorrect because you can't add this index to an already existing table. A local secondary index has the same partition key as the base table, but has a different sort key. It is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.

---

**Q14) A multimedia company based in Singapore wants to build a highly available database solution in AWS. It will be used for a content management application that expects high-volume requests and hence, can scale depending on the read throughput. The fully-managed database service will store data in JSON-like documents with attribute value sizes as large as 16 MB.**
**Which database service would best fit the company's needs?**

Amazon DocumentDB

**Explanation:-**To effectively manage a content management system, you must be able to collect and aggregate content from a variety of sources and then deliver it to the customer. Due to their flexible schema, document databases are perfect for collecting and storing any type of data. In AWS, Amazon DynamoDB and Amazon DocumentDB are designed to store documents. The concern with using DynamoDB in this situation is that the items in the table can have attribute value sizes only up to 400KB.

Amazon DocumentDB is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads. As a document database, Amazon DocumentDB makes it easy to store, query, and index JSON data. To distribute read workloads, it can support up to 15 replica instances that accept read requests.

Amazon DynamoDB

**Explanation:-**This option is incorrect. As mentioned above, the item size of a DynamoDB table cannot store anything larger than 400KB. Amazon DynamoDB could also potentially encounter capacity unit management concerns (e.g. RCU, WCU) due to the high read requirement.

Amazon RDS

**Explanation:-**This option is incorrect because the application needs a noSQL-compatible database rather than a traditional relational database.

Amazon Elasticache with Redis cluster

**Explanation:-**This option is incorrect because Elasticache is not designed to accommodate the heavy write workload of the application. Elasticache

is more popular for storing small amounts of data such as user-session information.

---

**Q15) A gaming company is planning to add a gaming leaderboard feature that would list down the top players for a popular mobile application in the Philippines. They predict it would introduce a huge volume of Query operations on the Amazon DynamoDB, and they don't want it to cause slow performance. The Database Team is working on a low-cost solution that will not require a lot of implementation and coding effort.**
**What should the Database team do to achieve these requirements?**

○ Create an Amazon ElastiCache for Redis and use Redis sorted sets to boost Query operations.
**Explanation:-**This option is incorrect because it will require more implementation and coding effort to achieve potentially similar results.
○ Use Contributor Insights for DynamoDB to address the most frequently accessed and throttled keys.
**Explanation:-**This option is incorrect because Contributor Insights is used to provide information about the most accessed and throttled items in a table or global secondary index. DynamoDB delivers this information to you via CloudWatch Contributor Insights rules, reports, and graphs of report data.
✓ Configure a DynamoDB Accelerator to cache data.
**Explanation:-**Cache memory is important because it improves the efficiency of data retrieval. When you are designing a database for performance, especially heavy read workloads, a cache solution will go a long way. AWS primarily offers Amazon ElastiCache as a web service that makes it easy to set up, manage, and scale a distributed in-memory data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution. At the same time, it helps remove the complexity associated with deploying and managing a distributed cache environment. It is ideal for gaming leaderboards.
A few years ago, AWS launched Amazon DynamoDB Accelerator (DAX), a highly available, in-memory cache for Amazon DynamoDB. If you're currently using DynamoDB or considering DynamoDB, DAX can offer you response times in microseconds and millions of requests per second. Instead of learning another database system with a new set of APIs and data types—and then rewriting applications to do the two-step dance needed for cache look-ups, population, and invalidation—you can simply point your existing DynamoDB application at the DAX endpoint. What used to take weeks and months now takes only moments with DAX.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DAX.html
https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.html
○ Convert the DynamoDB tables to global tables.
**Explanation:-**This option is incorrect because global tables are designed for multi-regional and multi-master databases. This does not address the concerns that were presented in the scenario.

---

**Q16) An application uses GetItem and PutItem operations respectively to read and write data to its DynamoDB table. It processes thousands of requests per hour that stores multiple related items in the table, where each item is processed individually. The network overhead of the workload affects the application's performance. A Database Specialist needs to work with a developer to refactor the application without implementing concurrency management. Failed operations are acceptable as long as the requests get reprocessed.**
**Which of the following will improve the application performance in the most cost-effective way?**

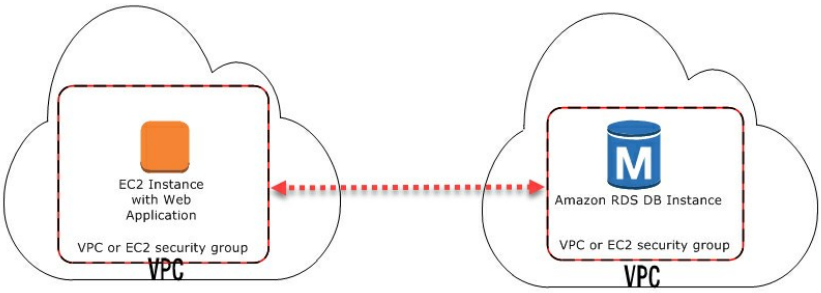
○ Refactor the application to use the TransactGetItems and TransactWriteItems operations in processing related items. Use the UnprocessedKeys map from the response if a partial result is returned to reprocess the failed items.
**Explanation:-**This option is incorrect because these operations are part of the DynamoDB Transactions feature which provides atomicity, consistency, isolation, and durability (ACID) in DynamoDB to maintain data integrity in your applications. Take note that every transactional read and write API call consumes high RCU and WCUs, unlike eventual or strong consistency requests. This entails a significant increase in costs which contradicts the requirements of the scenario. Using DynamoDB Batch Operations is valid since the scenario explicitly said that failed operations are acceptable as long as the items get reprocessed. Therefore, the scenario doesn't warrant the use of DynamoDB Transactions.
✓ Refactor the application to use the BatchGetItem and BatchWriteItem operations in processing related items. If a partial result is returned, use the UnprocessedKeys map from the response to reprocess the failed items.
**Explanation:-**For applications that need to read or write multiple items, DynamoDB provides the BatchGetItem and BatchWriteItem operations. Using these operations can reduce the number of network round trips from your application to DynamoDB. In addition, DynamoDB performs the individual read or write operations in parallel. Your applications benefit from this parallelism without having to manage concurrency or threading.
The batch operations are essentially wrappers around multiple read or write requests. For example, if a BatchGetItem request contains five items, DynamoDB performs five GetItem operations on your behalf. Similarly, if a BatchWriteItem request contains two put requests and four delete requests, DynamoDB performs two PutItem and four DeleteItem requests.
In general, a batch operation does not fail unless all of the requests in the batch fail. For example, suppose you perform a BatchGetItemoperation but one of the individual GetItem requests in the batch fails. In this case, BatchGetItem returns the keys and data from the GetItemrequest that failed. The other GetItem requests in the batch are not affected.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WorkingWithItems.html#WorkingWithItems.ConditionalUpdate
https://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_BatchGetItem.html
○ Increase the current RCU and WCU of the DynamoDB table. If a partial result is returned, use the UnprocessedKeys map from the response to reprocess the failed items.
**Explanation:-**This option is incorrect because increasing the RCU and WCU in DynamoDB will incur an additional cost. In addition, the UnprocessedKeys map is only available in BatchGetItem and BatchWriteItem operations.
○ Implement in-memory acceleration with DynamoDB Accelerator (DAX). Set the ConsistentRead to true for the read requests.
**Explanation:-**This option is incorrect because the application is not just read-intensive as it also heavily processes a lot of write operations. This option doesn't provide a holistic solution to the problem.

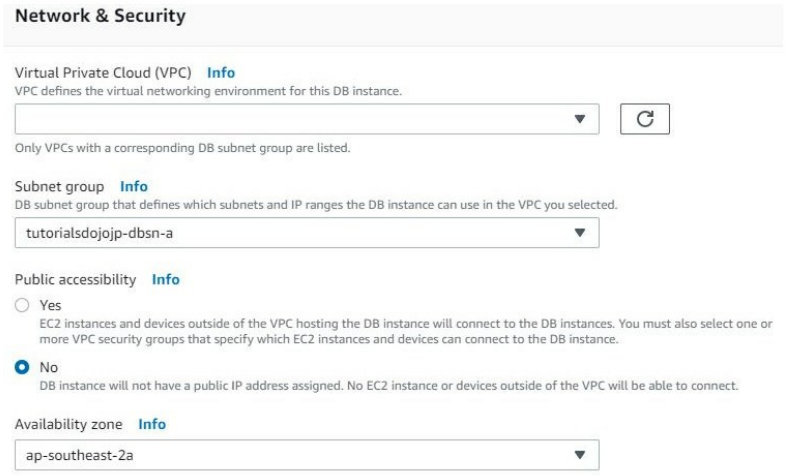
---

**Q17) A Database Specialist manages an Amazon Aurora PostgreSQL DB cluster currently residing in the default Amazon VPC with access to the Internet. The company wants to secure the database and associate it with a different VPC. The Application Manager reminded the Database Specialist that the application server resides in the default VPC and should successfully connect to the database once the change is completed.**
**What should the Database Specialist do to meet this requirement?**

○ Modify the existing Aurora DB cluster and associate it with the new VPC. Use VPC Peering to connect the default VPC to the new VPC.
**Explanation:-**This option is incorrect because it is not allowed to modify the VPC of an existing cluster.
✓ Restore the Aurora DB cluster snapshot in the new VPC. Use VPC Peering to allow the connection between the default VPC and the new VPC.
**Explanation:-**
One cannot modify the VPC of an Aurora cluster or instance. However, you can change the VPC of an Aurora cluster by using one of the following methods: 1) Create a clone in a different VPC, 2) Take a snapshot and then restore the snapshot in a different VPC, 3) Set up replication using

binary logging (MySQL only).
When your DB instance is in a different VPC from the EC2 instance you are using to access it; you can use VPC peering to access the DB instance. A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IP addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your VPCs, a VPC in another AWS account, or a VPC in a different AWS Region.




References:
https://aws.amazon.com/premiumsupport/knowledge-center/rds-vpc-aurora-cluster/
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_VPC.Scenarios.html

Restore the Aurora DB cluster snapshot in the new VPC using the AWS root account. It will automatically allow the connection between the default VPC and the new VPC.
**Explanation:-**This option is incorrect. The VPCs will not automatically allow the connection.
Create a new DB subnet group and associate it with the new VPC. Modify the Aurora DB cluster and associate it with the new DB Subnet Group. It will automatically allow the connection between the default VPC and the new VPC.
**Explanation:-**This option is incorrect. DB Subnet groups are associated with the VPC, and thus, you cannot change the subnet groups of the existing cluster independently.

---

**Q18) A database administrator is managing a newly-built Amazon RDS Oracle db.m5.large instance in a single Availability Zone. Lately, the increase in the volume of read requests has negatively impacted the database performance. Furthermore, the Application Manager is worried that the system is not built for high availability. If the database fails, the system should be able to serve its users within an hour. Data loss is acceptable.**
**What is the most cost-effective solution to meet these requirements?**

Create an AWS Lambda function that restores a snapshot of the Amazon RDS instance to a new instance. Create an Amazon RDS event notification that triggers this function whenever the primary instance is down.
**Explanation:-**This option is incorrect because this will not resolve the performance issue caused by the increased volume of read requests. Furthermore, it may not meet the recovery time objective of an hour when the database snapshot grows bigger.
Configure the RDS instance class to a higher class.
**Explanation:-**This option is incorrect. Even though this may resolve the performance impact of the increased volume of read requests, it does not meet the high availability requirement. The RDS instance is still a single point of contact.
Create a Read Replica and configure the application to connect to the replica for read requests.
**Explanation:-**Amazon RDS uses the MariaDB, MySQL, Oracle, PostgreSQL, and Microsoft SQL Server DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the read replica. You can reduce the load on your source DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. Furthermore, you can promote a read replica to a standalone instance as a disaster recovery solution if the source DB instance fails. The question has two requirements: 1) resolve the performance issue caused by the increased read traffic, and 2) create a high-availability solution with a recovery time objective of an hour. This problem can be resolved by creating a Read Replica for the RDS instance.
Configure Multi-AZ deployment to build a standby replica and configure the application to connect to the replica for read requests.
**Explanation:-**This option is incorrect because a standby replica cannot be used for scaling read traffic. Although it may meet the high availability requirement, the performance impact of the increased read traffic remains unresolved.

---

**Q19) A company is developing an application that will be used to visualize complex hybrid network architectures, process highly connected datasets, and detect security issues. It requires a database that quickly finds specific connection paths between the hosts and traces a malicious file to the original host that downloaded it. The database will also be used to determine the fastest route of sending data from one host to another. A bulk loader API must also be provided for loading data from external files directly to the DB instance.**
**Which of the following is the MOST suitable database engine that needs to be used in this scenario?**

Amazon QLDB

**Explanation:-**This option is incorrect because this is simply a fully managed ledger database that provides a transparent, immutable, and cryptographically verifiable transaction log owned by a central trusted authority. Amazon QLDB tracks each and every application data change and maintains a complete and verifiable history of changes over time.

Amazon Neptune

**Explanation:-**Amazon Neptune is a fast, reliable, fully-managed graph database service that makes it easy to build and run applications that work with highly connected datasets. The core of Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds latency. Amazon Neptune supports popular graph models Property Graph and W3C's RDF, and their respective query languages Apache TinkerPop Gremlin and SPARQL, allowing you to easily build queries that efficiently navigate highly connected datasets. Neptune powers graph use cases such as recommendation engines, fraud detection, knowledge graphs, drug discovery, and network security.

Amazon Neptune is highly available, with read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across Availability Zones. Neptune is secure with support for HTTPS encrypted client connections and encryption at rest. Neptune is fully managed, so you no longer need to worry about database management tasks such as hardware provisioning, software patching, setup, configuration, or backups.

You can use Amazon Neptune to store a graph of your network and use graph queries to answer questions like how many hosts are running a specific application. Neptune can store and process billions of events to manage and secure your network. If you detect an event that is an anomaly, you can use Neptune to quickly understand how it might affect your network by querying for a graph pattern using the attributes of the event. You can query Neptune to find other hosts or devices that may be compromised. For example, if you detect a malicious file on a host, Neptune can help you to find the connections between the hosts that spread the malicious file, and enable you to trace it to the original host that downloaded it.

Amazon Neptune provides a bulk Loader command for loading data from external files directly into a Neptune DB instance. You can use this command instead of executing a large number of INSERT statements, addVertex, and addEdge steps, or other API calls. The Neptune Loader command is faster, has less overhead, is optimized for large datasets, and supports both Gremlin data and the RDF (Resource Description Framework) data used by SPARQL.

References:

https://aws.amazon.com/neptune

https://aws.amazon.com/blogs/database/let-me-graph-that-for-you-part-1-air-routes/

https://docs.aws.amazon.com/neptune/latest/userguide/bulk-load.html

Amazon Aurora

**Explanation:-**This option is incorrect because it doesn't have a bulk loader API that can be used in loading data from external files directly to the DB instance. Moreover, using SQL queries for highly connected data is quite complex and hard to tune for performance, unlike Amazon Neptune.

Amazon DocumentDB

**Explanation:-**This option is incorrect because this is just a fully managed document database service that supports MongoDB workloads. Amazon DocumentDB is primarily used to store, query, and index JSON data. This database type is not suitable for processing highly connected datasets and for finding specific connection paths. You have to use Amazon Neptune instead.

---

**Q20) A financial services company based in Australia uses Amazon Redshift for its data-warehousing solutions. They have expanded recently in Singapore and is designing a solution that would integrate queries between data from an Amazon RDS for PostgreSQL database in Singapore with data from the Redshift cluster in Sydney.**
**Which solution will best simplify the integration between these data sources?**

Configure cross-regional snapshots with Redshift cluster and restore a new cluster with the latest snapshot in Singapore. Export the PostgreSQL tables to an S3 bucket and load it to the new Amazon Redshift cluster.

Configure cross-regional snapshots with Redshift cluster and restore a new cluster with the latest snapshot in Singapore. Export the PostgreSQL tables to an S3 bucket. Create an external schema and external tables from the S3 files and use Redshift Spectrum to query from S3 and the new Redshift cluster.

Export the RDS PostgreSQL tables to an S3 bucket. Create an external schema and external tables from the S3 files and use Redshift Spectrum to query from S3 and the new Redshift cluster.

Set up connectivity from your Amazon Redshift cluster to your Amazon RDS PostgreSQL cluster. Create an external schema from the PostgreSQL database and use federated queries to access both sources.

**Explanation:-**Previously, you need to extract data from your PostgreSQL database to Amazon Simple Storage Service (Amazon S3) and load it to Amazon Redshift using COPY, or query it from Amazon S3 with Amazon Redshift Spectrum. Now, Amazon Redshift Federated Query enables you to use the analytic power of Amazon Redshift to directly query data stored in Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL databases. With Federated queries, you can query and analyze data across operational databases, data warehouses, and data lakes. This feature can integrate queries from Amazon Redshift on live data in external databases with queries across your Amazon Redshift and Amazon S3 environments.

In some cases, you might access an Amazon RDS or Aurora database in a different AWS region than Amazon Redshift. In these cases, you typically incur network latency and billing charges for transferring data across AWS regions. However, the scenario asks for a solution that simplifies the integration between the two sources.

References:

https://docs.aws.amazon.com/redshift/latest/dg/federated-overview.html

https://aws.amazon.com/blogs/big-data/amazon-redshift-federated-query-best-practices-and-performance-considerations/

---

**Q21) A company is developing an application that will use an Amazon RDS MySQL database to process and store online transactions worldwide. It will have an unpredictable workload that can potentially exceed the initially allocated storage of the database. The Database Specialist needs to design a cost-effective solution to meet the demand.**
**What should the Database Specialist do to accomplish this?**

Launch the Amazon RDS MySQL database with default settings. RDS will automatically scale the database storage by default.

**Explanation:-**This option is incorrect because RDS does not automatically scale the database storage by default. You have to enable storage autoscaling to meet the requirement.

Enable Performance Insights for the new RDS DB instance.

**Explanation:-**This option is incorrect because this is just a database performance tuning and monitoring feature that helps you quickly assess the load on your database and determine when and where to take action.

Enable Enhanced Monitoring for the new RDS DB instance.

**Explanation:-**This option is incorrect because this simply provides real-time metrics of the underlying operating system (OS) that your RDS DB instance runs on.

Enable storage autoscaling for the new RDS DB instance.

**Explanation:-**If your workload is unpredictable, you can enable storage autoscaling for an Amazon RDS DB instance. To do so, you can use the Amazon RDS console, the Amazon RDS API, or the AWS CLI.

For example, you might use this feature for a new mobile gaming application that users are adopting rapidly. In this case, a rapidly increasing

workload might exceed the available database storage. To avoid having to manually scale up database storage, you can use Amazon RDS storage autoscaling.
With storage autoscaling enabled, Amazon RDS automatically scales up your storage when it detects that you are running out of free database space. Amazon RDS starts a storage modification for an autoscaling-enabled DB instance when these factors apply:
Free available space is less than 10 percent of the allocated storage.
The low-storage condition lasts at least five minutes.
At least six hours have passed since the last storage modification.
The additional storage is in increments of whichever of the following is greater:
5 GiB
10 percent of currently allocated storage
Storage growth prediction for 7 hours based on the FreeStorageSpace metrics change in the past hour.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html#USER_PIOPS.Autoscaling
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html

---

**Q22) A Database Specialist currently manages an Amazon Aurora with PostgreSQL compatibility cluster. Every afternoon, performance is degraded due to huge spikes of read operations, maximizing the CPU utilization of the Aurora Replica. The Product Owner asked for a solution that could dynamically meet the application's connectivity and workload requirements.**
**What is the most cost-effective solution that can address the requirement?**

- Migrate to Aurora Global Database.

**Explanation:-**This option is incorrect because it is designed for multi-regional applications. Aurora replicates data to the secondary AWS Regions with a typical latency of under a second. You issue write operations directly to the primary DB instance in the primary AWS Region.

- Create a new Aurora cluster with at least one replica and Aurora Auto Scaling enabled.

**Explanation:-**This option is incorrect because Aurora Auto Scaling can only be enabled in an existing cluster. Therefore, it is not necessary to create a new cluster.

- ✅ Enable Aurora Auto Scaling in the cluster.

**Explanation:-**One can achieve read scaling for your Aurora DB cluster by creating up to 15 Aurora Replicas in a DB cluster that uses single-master replication. Each Aurora Replica returns the same data from the cluster volume with minimal replica lag—usually considerably less than 100 milliseconds after the primary instance has written an update. As your read traffic increases, you can create additional Aurora Replicas and connect to them directly to distribute the read load for your DB cluster. Aurora Replicas don't have to be of the same DB instance class as the primary instance.
To meet your connectivity and workload requirements, Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas provisioned for an Aurora DB cluster using single-master replication. Aurora Auto Scaling is available for both Aurora MySQL and Aurora PostgreSQL. Aurora Auto Scaling enables your Aurora DB cluster to handle sudden increases in connectivity or workload. When the connectivity or workload decreases, Aurora Auto Scaling removes unnecessary Aurora Replicas so that you don't pay for unused provisioned DB instances.
References:
https://docs.amazonaws.cn/en_us/AmazonRDS/latest/AuroraUserGuide/Aurora.Integrating.AutoScaling.html
https://aws.amazon.com/getting-started/hands-on/aurora-autoscaling-with-readreplicas/

- Upgrade the instance class of the Aurora Replica.

**Explanation:-**This option is incorrect because this solution does not meet the requirement - i.e. it does not dynamically meet the connectivity and workload requirements of the application.

---

**Q23) An enterprise application that uses an Amazon RDS MySQL database experienced a 3-hour outage that affected thousands of customers around the globe. Upon investigation, the root cause was due to a recent change in the DB security group.**
**What should the Database Specialist do to get a notification whenever the DB security group is modified?**

- Enable Database Activity Streams that will provide real-time notifications for any changes in the RDS database, including its associated security group.

**Explanation:-**This option is incorrect because this feature is only available in Amazon Aurora. More importantly, it can't monitor or track the changes to your security group. It only provides a near real-time data stream of the database activity in your database.

- Enable Enhanced Monitoring that will provide real-time metrics and notifications to any RDS database changes.

**Explanation:-**This option is incorrect because Enhanced Monitoring doesn't send notifications by default nor track security group changes.

- ✅ Set up Amazon RDS Event Notification and subscribe to the configuration change category for the DB security group.

**Explanation:-**Amazon RDS uses the Amazon Simple Notification Service (Amazon SNS) to provide notification when an Amazon RDS event occurs. These notifications can be in any notification form supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint.
Amazon RDS groups these events into categories that you can subscribe to so that you can be notified when an event in that category occurs. You can subscribe to an event category for a DB instance, DB snapshot, DB parameter group, or DB security group. For example, if you subscribe to the Backup category for a given DB instance, you are notified whenever a backup-related event occurs that affects the DB instance. If you subscribe to a configuration change category for a DB security group, you are notified when the DB security group is changed. You also receive a notification when an event notification subscription changes.
Event notifications are sent to the addresses that you provide when you create the subscription. You might want to create several different subscriptions, such as one subscription receiving all event notifications and another subscription that includes only critical events for your production DB instances. You can easily turn off notification without deleting a subscription by choosing No for Enabled in the Amazon RDS console or by setting the Enabled parameter to false using the AWS CLI or Amazon RDS API.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Events.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/MonitoringOverview.html

- Set up Amazon RDS Performance Insights and subscribe to the configuration change category for the DB security group.

**Explanation:-**This option is incorrect because RDS Performance Insights is primarily used for visualizing and filtering the database load by waits, SQL statements, hosts, or users. This feature can't be used to track changes to the associated security group of the database.

---

**Q24) The Database Specialist needs to add a multithreaded cache layer to an application to improve its performance. The application will cache small arbitrary data from common database query results and external API calls. The cache layer must utilize multiple processing cores to easily scale up compute capacity and handle more operations.**
**What is the most suitable service that the Database Specialist should use in this scenario?**

- Amazon Elasticsearch

**Explanation:-**This option is incorrect because this is just a fully managed service that makes it easy for you to deploy, secure, and run Elasticsearch cost-effectively at scale. Elasticsearch is an open-source search engine and not an in-memory data store, unlike Redis or Memcached.

✅ Amazon ElastiCache for Memcached

**Explanation:-**Redis and Memcached are popular, open-source, in-memory data stores. Although they are both easy to use and offer high performance, there are important differences to consider when choosing an engine. Memcached is designed for simplicity while Redis offers a rich set of features that make it effective for a wide range of use cases.

In this scenario, Redis can provide a much more durable and powerful cache layer to the prototype distributed system, however, you should take note of one keyword in the requirement: multithreaded. In terms of commands execution, Redis is mostly a single-threaded server. It is not designed to benefit from multiple CPU cores unlike Memcached, however, you can launch several Redis instances to scale out on several cores if needed. Memcached is a more suitable choice since the scenario specifies that the system will run large nodes with multiple cores or threads and in addition, the prototype only needs simple data structures that Memcached can adequately provide.

The Auto Discovery feature of Amazon ElastiCache for Memcached has the ability to enable the client programs to automatically identify all of the nodes in a cache cluster.

You can choose Memcached over Redis if you have the following requirements:

You need the simplest model possible.

You need to run large nodes with multiple cores or threads.

You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases.

You need to cache objects, such as a database.

References:

https://aws.amazon.com/elasticache/redis-vs-memcached

https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/SelectEngine.html

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

⚪ Amazon ElastiCache for Redis

**Explanation:-**This option is incorrect because Redis does not entirely support a multithreaded architecture. Although Redis has more features compared with Memcached, the scenario only requires a cache layer which is multithreaded and can store simple data model. This is why Memcached is a more suitable cache engine to choose from instead of Redis.

⚪ Amazon CloudFront

**Explanation:-**This option is incorrect because this is a content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency. It is not a suitable service to use in caching database queries and external API calls.

---

**Q25) An organization requires a cost-effective data warehouse solution that stores 120 TB of data in a durable, consistent, and highly structured format. The solution requires low-latency responses for data queries for the current year. It must automatically scale to support the fluctuating number of incoming queries. The users and the IT Compliance team should also have access to the entire 10-year dataset and other historical data.**
**Which of the following options meets the above requirements?**

✅ Use Amazon Redshift as a data warehouse solution to store the most recent data of the current year. Enable the Concurrency Scaling feature to provide consistently fast performance during periods of fluctuating analytical demand. Store the data from the previous years and other historical data on an S3 bucket. Use Redshift Spectrum to query the historical data.

**Explanation:-**Amazon Redshift is a fast, fully managed cloud data warehouse that makes it simple and cost-effective to analyze all your data using standard SQL and your existing Business Intelligence (BI) tools. It allows you to run complex analytic queries against terabytes to petabytes of structured data, using sophisticated query optimization, columnar storage on high-performance storage, and massively parallel query execution. Amazon Redshift also includes Amazon Redshift Spectrum, allowing you to run SQL queries directly against exabytes of unstructured data in Amazon S3 data lakes. No loading or transformation is required, and you can use open data formats, including Avro, CSV, Grok, Amazon Ion, JSON, ORC, Parquet, RCFile, RegexSerDe, Sequence, Text, and TSV. Redshift Spectrum automatically scales query compute capacity based on the data retrieved, so queries against Amazon S3 run fast, regardless of data set size.

Concurrency Scaling is a feature in Amazon Redshift that provides consistently fast query performance, even with thousands of concurrent queries. With this feature, Amazon Redshift automatically adds transient capacity when needed to handle heavy demand. Amazon Redshift automatically routes queries to scaling clusters, which are provisioned in seconds and begin processing queries immediately.

This feature is free for most customers. Each Amazon Redshift cluster earns up to one hour of free Concurrency Scaling credits per day. This gives you predictability in your month-to-month cost, even during periods of fluctuating analytical demand.

Elastic Resize adds or removes nodes from a single Redshift cluster within minutes to manage its query throughput. For example, an ETL workload for certain hours in a day or month-end reporting may need additional Redshift resources to complete on time. Concurrency Scaling automatically adds additional cluster resources to increase the overall query concurrency.

References:

https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-clusters.html

https://aws.amazon.com/redshift/faqs/

⚪ Store both the recent and historical data of the data warehouse solution to Amazon Redshift. Deploy a number of large, dense storage instances.

**Explanation:-**This option is incorrect because this option doesn't scale automatically based on the incoming user requests. A better solution is to enable the Concurrency Scaling feature. Moreover, storing the historical data to Redshift entails higher operating costs as compared with storing the data to S3 and fetching it using Amazon Redshift Spectrum.

⚪ Store both the recent and historical data of the data warehouse solution to Amazon Redshift. Enable the Auto Scaling feature to provide consistently fast performance during periods of fluctuating analytical demand. Deploy a number of large, dense storage instances.

**Explanation:-**This option is incorrect because storing the historical data to Redshift is not a cost-effective solution and could result in higher operating costs. A better way is to store the old data to S3 and fetch it using Amazon Redshift Spectrum. Deploying a number of large storage instances also drives up your expenses. You have to use the Concurrency Scaling feature to automatically scale in or scale out your Redshift cluster.

⚪ Set up the data warehouse solution using Amazon Redshift to store the most recent data of the current year. Store the data from the previous years and other historical data on an S3 bucket. Use Redshift Spectrum to query the historical data. Use the Elastic Resize feature to provide consistently fast performance during periods of fluctuating analytical demand.

**Explanation:-**This option is incorrect because the Elastic Resize feature is usually implemented manually. Using the Concurrency Scaling feature is a more suitable option since it automatically scales the Redshift cluster to support the fluctuating number of incoming queries.

---

**Q26) A global retail company runs an e-commerce application supporting two sites - London and Singapore. The application has two web servers in the eu-west-2 and ap-southeast-1 Regions while accessing a shared database running on Amazon RDS for MySQL DB instance in a Multi-AZ deployment hosted in the eu-west-2 Region. The Application Manager complains to the Database team that the Singapore team has been facing slow performance when viewing the reports dashboard. A Database Specialist is assigned to create a solution that offers acceptable performance for users across both locations.**
**What is the FASTEST method to resolve the performance issue with minimal downtime?**

✅ Create an RDS read replica in the ap-southeast-1 Region from the primary RDS DB instance in the eu-west-2 Region. Reconfigure the ap-

southeast-1 webserver to access this replica.
**Explanation:-**With Amazon RDS, you can create a MariaDB, MySQL, Oracle, or PostgreSQL read replica in a different AWS Region from the source DB instance. Creating a cross-Region read replica isn't supported for SQL Server on Amazon RDS. You create a read replica in a different AWS Region to do the following:
- Improve your disaster recovery capabilities.
- Scale read operations into an AWS Region closer to your users.
- It is easier to migrate from a data center in one AWS Region to a data center in another AWS Region.
Creating a read replica in a different AWS Region from the source instance is similar to creating a replica in the same AWS Region. However, you can only create a cross-Region Amazon RDS read replica from a source Amazon RDS DB instance that is not a read replica of another Amazon RDS DB instance.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html
- Migrate the RDS MySQL DB instance to Aurora Global Databases and set the environment in the eu-west-2 and ap-southeast-1 Regions. Reconfigure the ap-southeast-1 webserver to access replica in the same region.
**Explanation:-**This option is incorrect. Although this solution can work, it is not the fastest method to implement.
- Create an RDS read replica in the eu-west-2 Region where the primary instance resides. Create a read replica in the ap-southeast-1 Region from the read replica located on the eu-west-2 Region. Reconfigure the ap-southeast-1 webserver to access this replica.
**Explanation:-**This option is incorrect. You can only create a cross-Region Amazon RDS read replica from a source Amazon RDS DB instance that is not a read replica of another Amazon RDS DB instance.
- Create a new instance from a copied RDS DB instance snapshot in the ap-southeast-1 Region. Use AWS DMS and change data capture (CDC) to update the new instance in the ap-southeast-1 Region. Reconfigure the ap-southeast-1 webserver to access this instance.
**Explanation:-**This option is incorrect. Although this solution could potentially work, it is neither the fastest nor the most cost-effective.

---

**Q27) A Database Specialist is configuring the connection management for an Amazon Aurora for MySQL cluster with four Aurora replicas - two db.r5.large and two db.r5.2xlarge instances. While planning for the read workload management, the application owner wanted to dedicate the r5.large instances to the analysis team and leave the remaining instances, including the newly created replicas, to other users. In case of a failover, the analysis team replicas should be the least likely candidates for promotion.**
**What should the Database Specialist do to meet these requirements? (Select THREE.)**

- ✅ Create an ANY custom endpoint with an exclusion list. Exclude the db.r5.large replicas.

**Explanation:-**When you connect to an Aurora cluster, the hostname and port that you specify points to an intermediate handler called an endpoint. You can map each connection to the appropriate instance or group of instances based on your use case.
Amazon Aurora has four types of endpoints available:
cluster endpoint (or writer endpoint),
reader endpoint,
instance endpoint
custom endpoint
Each Aurora cluster has a single built-in reader and cluster endpoint, whose name and other attributes are managed by Aurora. You cannot create, delete, or modify both kinds of endpoints. An instance endpoint connects to a specific DB instance within an Aurora cluster.
Meanwhile, each custom endpoint has an associated type that determines which DB instances are eligible to be associated with that endpoint. Currently, the type can be READER, WRITER, or ANY. Only DB instances that are read-only Aurora Replicas can be part of a READER custom endpoint. Both read-only Aurora Replicas and the read/write primary instance can be part of an ANY custom endpoint. Aurora directs connections to cluster endpoints with type ANY to any associated DB instance with equal probability. The WRITER type applies only to multi-master clusters because those clusters can include multiple read/write DB instances.
Furthermore, you can define a list of DB instances to include in, or exclude from, a custom endpoint. We refer to these lists as static and exclusion lists, respectively. Each custom endpoint can contain only one of these list types. While the static list enforces that the endpoint connects to the specified instances, the exclusion list enforces the endpoint to represent all DB instances in the cluster, including any that you add in the future, except the ones specified for exclusion.
You can customize the order in which your Aurora Replicas are promoted to the primary instance after a failure by assigning each replica a priority. Priorities range from 0 for the first priority to 15 for the last priority. If the primary instance fails, Amazon RDS promotes the Aurora Replica with a better priority to the new primary instance. You can assign lower priority tiers to replicas that you do not want to be promoted to the primary instance. However, if the higher priority replicas on the cluster are unhealthy or unavailable for some reason then Amazon RDS will promote the lower priority replica.
In this scenario, the Database Specialist needs to create a minimum of two custom endpoints to meet the requirements. The analysis team can use a custom endpoint that is configured to use a static list, which connects to either one of the two db.r5.large replicas. At the same time, the specialist needs to create another custom endpoint which excludes those two replicas via an exclusion list. The other users will connect to the endpoint. To demote the promotion priority of the replicas for the analysis teams, the specialist needs to configure the priority tier parameter for the two db.r5.large replicas to tier-15.
- Create a cluster endpoint and configure it to use the db.r5.large replicas.

**Explanation:-**This option is incorrect because you cannot create, delete, or modify a cluster endpoint. The built-in cluster endpoint points to the primary instance.
- ✅ Create a custom endpoint and include the db.r5.large replicas in a static list.

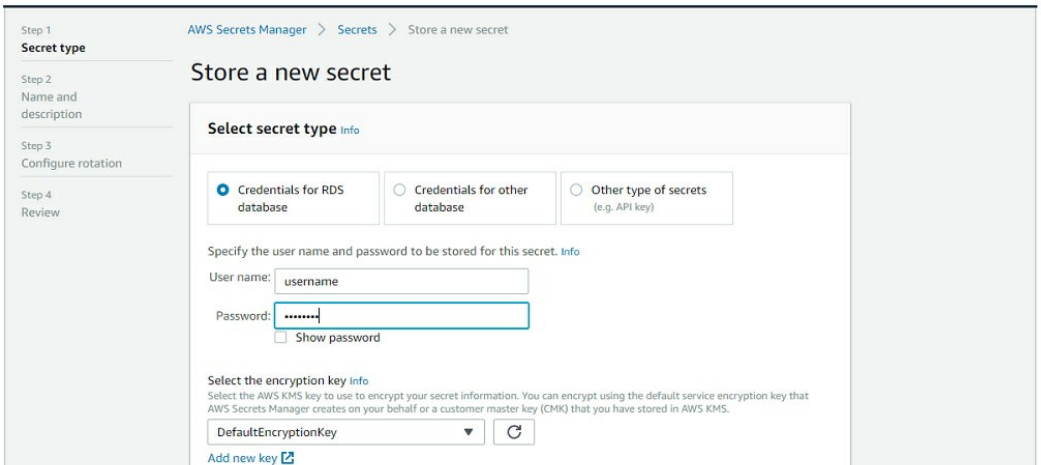
**Explanation:-**When you connect to an Aurora cluster, the hostname and port that you specify points to an intermediate handler called an endpoint. You can map each connection to the appropriate instance or group of instances based on your use case.
Amazon Aurora has four types of endpoints available:
cluster endpoint (or writer endpoint),
reader endpoint,
instance endpoint
custom endpoint
Each Aurora cluster has a single built-in reader and cluster endpoint, whose name and other attributes are managed by Aurora. You cannot create, delete, or modify both kinds of endpoints. An instance endpoint connects to a specific DB instance within an Aurora cluster.
Meanwhile, each custom endpoint has an associated type that determines which DB instances are eligible to be associated with that endpoint. Currently, the type can be READER, WRITER, or ANY. Only DB instances that are read-only Aurora Replicas can be part of a READER custom endpoint. Both read-only Aurora Replicas and the read/write primary instance can be part of an ANY custom endpoint. Aurora directs connections to cluster endpoints with type ANY to any associated DB instance with equal probability. The WRITER type applies only to multi-master clusters because those clusters can include multiple read/write DB instances.
Furthermore, you can define a list of DB instances to include in, or exclude from, a custom endpoint. We refer to these lists as static and exclusion lists, respectively. Each custom endpoint can contain only one of these list types. While the static list enforces that the endpoint connects to the

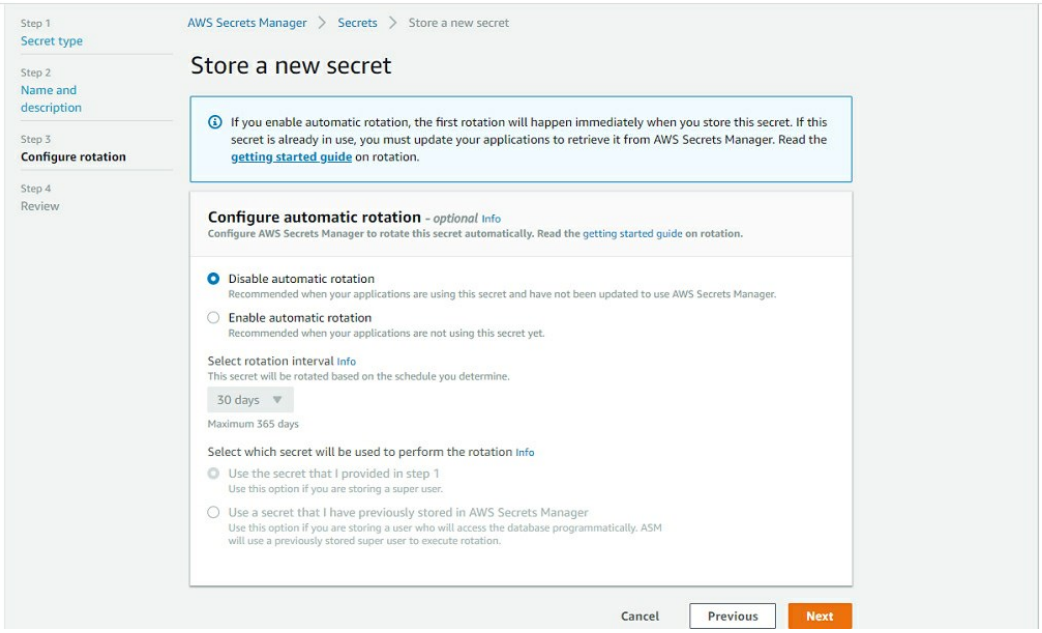
specified instances, the exclusion list enforces the endpoint to represent all DB instances in the cluster, including any that you add in the future, except the ones specified for exclusion.
You can customize the order in which your Aurora Replicas are promoted to the primary instance after a failure by assigning each replica a priority. Priorities range from 0 for the first priority to 15 for the last priority. If the primary instance fails, Amazon RDS promotes the Aurora Replica with a better priority to the new primary instance. You can assign lower priority tiers to replicas that you do not want to be promoted to the primary instance. However, if the higher priority replicas on the cluster are unhealthy or unavailable for some reason then Amazon RDS will promote the lower priority replica.
In this scenario, the Database Specialist needs to create a minimum of two custom endpoints to meet the requirements. The analysis team can use a custom endpoint that is configured to use a static list, which connects to either one of the two db.r5.large replicas. At the same time, the specialist needs to create another custom endpoint which excludes those two replicas via an exclusion list. The other users will connect to the endpoint. To demote the promotion priority of the replicas for the analysis teams, the specialist needs to configure the priority tier parameter for the two db.r5.large replicas to tier-15.
✅ Set the priority tier of the chosen analysis team replicas to tier-15.
**Explanation:-**When you connect to an Aurora cluster, the hostname and port that you specify points to an intermediate handler called an endpoint. You can map each connection to the appropriate instance or group of instances based on your use case.
Amazon Aurora has four types of endpoints available:
cluster endpoint (or writer endpoint),
reader endpoint,
instance endpoint
custom endpoint
Each Aurora cluster has a single built-in reader and cluster endpoint, whose name and other attributes are managed by Aurora. You cannot create, delete, or modify both kinds of endpoints. An instance endpoint connects to a specific DB instance within an Aurora cluster.
Meanwhile, each custom endpoint has an associated type that determines which DB instances are eligible to be associated with that endpoint. Currently, the type can be READER, WRITER, or ANY. Only DB instances that are read-only Aurora Replicas can be part of a READER custom endpoint. Both read-only Aurora Replicas and the read/write primary instance can be part of an ANY custom endpoint. Aurora directs connections to cluster endpoints with type ANY to any associated DB instance with equal probability. The WRITER type applies only to multi-master clusters because those clusters can include multiple read/write DB instances.
Furthermore, you can define a list of DB instances to include in, or exclude from, a custom endpoint. We refer to these lists as static and exclusion lists, respectively. Each custom endpoint can contain only one of these list types. While the static list enforces that the endpoint connects to the specified instances, the exclusion list enforces the endpoint to represent all DB instances in the cluster, including any that you add in the future, except the ones specified for exclusion.
You can customize the order in which your Aurora Replicas are promoted to the primary instance after a failure by assigning each replica a priority. Priorities range from 0 for the first priority to 15 for the last priority. If the primary instance fails, Amazon RDS promotes the Aurora Replica with a better priority to the new primary instance. You can assign lower priority tiers to replicas that you do not want to be promoted to the primary instance. However, if the higher priority replicas on the cluster are unhealthy or unavailable for some reason then Amazon RDS will promote the lower priority replica.
In this scenario, the Database Specialist needs to create a minimum of two custom endpoints to meet the requirements. The analysis team can use a custom endpoint that is configured to use a static list, which connects to either one of the two db.r5.large replicas. At the same time, the specialist needs to create another custom endpoint which excludes those two replicas via an exclusion list. The other users will connect to the endpoint. To demote the promotion priority of the replicas for the analysis teams, the specialist needs to configure the priority tier parameter for the two db.r5.large replicas to tier-15.
⚪ Create an ANY custom endpoint and include all replicas, except for the db.r5.large replicas, in a static list.
**Explanation:-**This option is incorrect because a custom endpoint using a static list will not automatically include the new instances. Hence, it will not meet the requirements.
⚪ Set the priority tier of the chosen analysis team replicas to tier-0.
**Explanation:-**This option is incorrect because setting the priority tier to tier-0 will put the replicas to the highest promotion priority.

---

**Q28) A CRM application in an Amazon EC2 instance uses an Amazon RDS for PostgreSQL instance for the database inside the same VPC. Worried about potential threats, the Application Manager wants the database username and password credentials configured for the Application to change every 35 days. The Database Specialist needs to enforce the rule without breaking the Application.**
**Which steps should the Database Specialist take to manage the database credentials securely?**

✅ Use AWS Secrets Manager to store a new secret that connects to the PostgreSQL database. Update the Application to retrieve the secret. Enable Automatic Rotation in the Secrets Manager for the secret.
**Explanation:-**
You can configure AWS Secrets Manager to rotate secrets automatically, which can help you meet your security and compliance needs. It is a service that makes it easier to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.
Secrets Manager offers built-in integrations for MySQL, PostgreSQL, and Amazon Aurora on Amazon RDS, and can rotate credentials for these databases natively. To retrieve secrets, employees replace plaintext secrets with a call to Secrets Manager APIs, eliminating the need to hard-code secrets in source code or update configuration files and redeploy code when secrets are rotated.
The scenario asks for a solution that fixes a potential vulnerability of an application that uses static passwords and avoid a potential break due to a mistake in implementing that fix. A break usually happens when there is a requirement for manual intervention, normally executed by humans. Automation becomes critical.






● Use AWS Systems Manager Parameter Store to store a new secret that connects to the PostgreSQL database. Update the Application to retrieve the secret. Enable Automatic Rotation in the AWS SSM for the secret.
**Explanation:-**This option is incorrect because AWS Systems Manager does not include a feature that rotates passwords automatically.
● Configure IAM database authentication for the database. Create an IAM user and map it to a database user for the application credential. Require the Application Manager to update the password every 35 days.
**Explanation:-**This option is incorrect because it would require human intervention and can lead to an application break.
● Store the username and password in a text file and upload it in an Amazon S3 bucket. Restrict permissions on the bucket to the proper IAM role. Modify the Application to retrieve the credentials during its startup. Rotate the credentials every 35 days.
**Explanation:-**This option is incorrect because it would require more human intervention and can lead to an application break.

---

**Q29) The Audit team flagged an unencrypted Amazon Aurora with MySQL Compatibility DB Cluster and required the Financial Services company to secure the data at rest via database encryption. The Application Manager agreed to allocate the whole weekend to perform maintenance on the database to comply with the Audit team.**
**What is the fastest way to encrypt the database?**

● Take a manual snapshot of the unencrypted Aurora MySQL DB Writer Instance. Create an encrypted copy of that snapshot. Restore a DB instance from the encrypted snapshot.
**Explanation:-**This option is incorrect. One cannot create an encrypted copy of an unencrypted snapshot. However, this works for Amazon RDS.
● Use AWS Database Migration Service to migrate the database to an encrypted Aurora DB cluster.
**Explanation:-**This option is incorrect because it is not the fastest solution.
✅ Restore the unencrypted Aurora DB cluster snapshot to an encrypted Aurora DB cluster.
**Explanation:-**You can encrypt your Amazon Aurora DB clusters and snapshots at rest by enabling the encryption option for your Amazon Aurora DB clusters. Data that is encrypted at rest includes the underlying storage for DB clusters, automated backups, read replicas, and snapshots.
Amazon Aurora encrypted DB clusters use the industry-standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon Aurora DB clusters. After your data is encrypted, Amazon Aurora handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.
DB clusters that are encrypted can't be modified to disable encryption. You can't convert an unencrypted DB cluster to an encrypted one. However, you can restore an unencrypted Aurora DB cluster snapshot to an encrypted Aurora DB cluster. To do this, specify a KMS encryption key when you restore from the unencrypted DB cluster snapshot.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Overview.Encryption.html#Overview.Encryption.Limitations
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_CopySnapshot.html#USER_CopySnapshot.Encryption
● Take a manual snapshot of the unencrypted Aurora MySQL DB Cluster. Create an encrypted copy of that snapshot. Create a DB Cluster from the encrypted snapshot.
**Explanation:-**This option is incorrect. One cannot create an encrypted copy of an unencrypted snapshot.

---

**Q30) The Security team requested the Database Specialist to reinforce auditing in an Amazon RDS for Microsoft SQL Server instance. They wanted to audit the SQL Server error logs and agent logs for investigation purposes and keep them in reliable storage for a year. The Database Specialist wants a solution that does not require a lot of coding development and day-to-day management.**
**Which approach will meet these requirements?**

● Publish the logs to Amazon CloudWatch Logs. Create an export task in CloudWatch Logs to move the logs to an Amazon S3 bucket with a 1-year lifecycle rule.
**Explanation:-**This option is incorrect. You do not need to create an export task to an S3 bucket to meet the requirement.
● Use AWS CloudTrail to store the SQL Server error logs and agent logs and keep them for a year.
**Explanation:-**This optoin is incorrect. AWS CloudTrail keeps a record of actions taken by a user, role, or an AWS service in Amazon RDS. CloudTrail captures all API calls for Amazon RDS as events.
● Use RDS Performance Insights to store the SQL Server error logs and agent logs and keep them for a year.
**Explanation:-**This option is incorrect. RDS Performance Insights helps assess the load on your database, and determine when and where to take action.
✅ Publish the logs to Amazon CloudWatch Logs and set them to expire after a year.
**Explanation:-**You can access Microsoft SQL Server error logs, agent logs, trace files, and dump files using the Amazon RDS console, AWS CLI, or

RDS API. In addition to viewing and downloading DB instance logs, you can publish logs to Amazon CloudWatch Logs. With CloudWatch Logs, you can perform real-time analysis of the log data, store the data in highly durable storage, and manage the data with the CloudWatch Logs Agent. AWS retains log data published to CloudWatch Logs for an indefinite period unless you specify a retention period.
With CloudWatch Logs, you can do the following:
- Store logs in highly durable storage space with a retention period that you define.
- Search and filter log data.
- Share log data between accounts.
- Export logs to Amazon S3.
- Stream data to Amazon Elasticsearch Service.
- Process log data in real-time with Amazon Kinesis Data Streams.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_LogAccess.html
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_LogAccess.Concepts.SQLServer.html

---

**Q31) A Database Specialist recently created an Amazon RDS for PostgreSQL DB instance in a Multi-AZ deployment and has enforced the Application team to connect to the database using IAM Database Authentication. After enabling IAM Database Authentication in the DB instance and creating policies that grant the IAM users rds-db:connect action, a corresponding user was created inside the PostgreSQL DB instance. However, the Application team complained that the attempt to connect was unsuccessful.**
**What should the Database Specialist consider to resolve the issue?**

- The database user needs to be created with the AWSAuthenticationPlugin AS 'RDS' clause.

**Explanation:-**This option is incorrect because this is only applicable for Amazon RDS for MySQL and not for PostgreSQL.

- ✅ The rds_iam role needs to be granted in the PostgreSQL database user.

**Explanation:-**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. By default, IAM database authentication is disabled on DB instances. You can enable IAM database authentication (or disable it again) using the AWS Management Console, AWS CLI, or the API.
To allow an IAM user or role to connect to your DB instance, you must create an IAM policy that allows rds-db:connect action. After that, you attach the policy to an IAM user or role. To use IAM authentication with PostgreSQL, connect to the DB instance, create database users, and then grant them the rds_iam role.

References:

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

- The database password the Application teams used for the user was incorrect.

**Explanation:-**This option is incorrect. IAM Database Authentication allows you to connect to the database without a database password.

- IAM Database Authentication works only for IAM roles.

**Explanation:-**This option is incorrect. IAM Database Authentication can be enabled for IAM users.

---

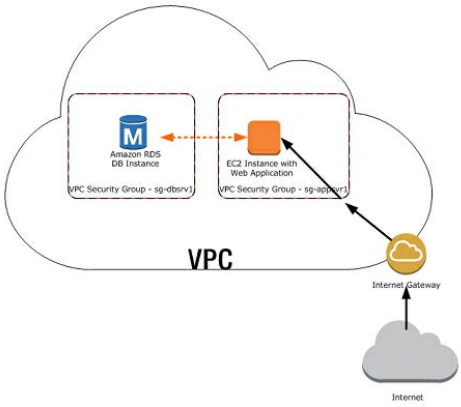
**Q32) A company plans to create a new enterprise application. The company's Database Specialist intends to deploy a new Amazon RDS for MariaDB database environment. For ease of management, the DB instance will reside on the same VPC as the application running in an EC2 instance, which will be accessible from the Internet. The Database Specialist needs to prevent the database from being accessed outside the VPC.**
**Which steps should the Database Specialist take to set up the database? (Select THREE.)**

- Create a VPC security group for the EC2 instance that allows traffic from 0.0.0.0/0.

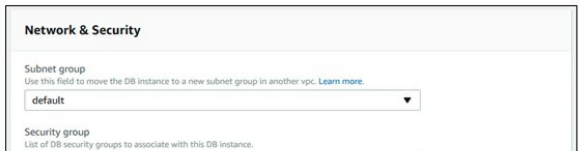
**Explanation:-**This option is incorrect as this will allow the database to be accessed outside the VPC. Although this may be an option moving forward, this does not directly meet the requirements in the scenario.

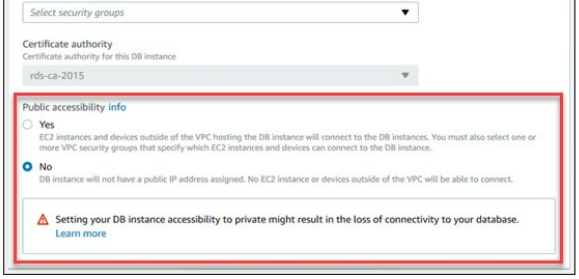
- ✅ Create a DB subnet group with two private subnets associated with the VPC.

**Explanation:-**
A typical scenario includes a DB instance in an Amazon VPC, that shares data with a web server running in the same VPC.




Because your DB instance only needs to be available to your web server and not to the public Internet, you create a VPC with both public and private subnets. The web server is hosted in the public subnet so that it can reach the public Internet. The DB instance is hosted in a private subnet. The web server can connect to the DB instance because it is hosted within the same VPC. Still, the DB instance is not available to the public Internet, providing greater security.






Your VPC must have at least two subnets. These subnets must be in two different Availability Zones in the AWS Region where you want to deploy your DB instance. A subnet is a segment of a VPC's IP address range that you can specify, and that lets you group instances based on your security and operational needs. Your VPC must have a DB subnet group that you create. Each DB subnet group should have subnets in at least two Availability Zones in a given AWS Region. The subnets in a DB subnet group are either public or private. They can't be a mix of both public and private subnets. Your VPC must have a VPC security group that allows access to the DB instance.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_VPC.Scenarios.html

✅ Create a DB instance in the VPC with Public accessibility option disabled.
**Explanation:-**A typical scenario includes a DB instance in an Amazon VPC, that shares data with a web server running in the same VPC. Because your DB instance only needs to be available to your web server and not to the public Internet, you create a VPC with both public and private subnets. The web server is hosted in the public subnet so that it can reach the public Internet. The DB instance is hosted in a private subnet. The web server can connect to the DB instance because it is hosted within the same VPC. Still, the DB instance is not available to the public Internet, providing greater security.
Your VPC must have at least two subnets. These subnets must be in two different Availability Zones in the AWS Region where you want to deploy your DB instance. A subnet is a segment of a VPC's IP address range that you can specify, and that lets you group instances based on your security and operational needs. Your VPC must have a DB subnet group that you create. Each DB subnet group should have subnets in at least two Availability Zones in a given AWS Region. The subnets in a DB subnet group are either public or private. They can't be a mix of both public and private subnets. Your VPC must have a VPC security group that allows access to the DB instance.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_VPC.Scenarios.html
✅ Create a VPC security group for the MariaDB instance that allows inbound traffic on port 3306 from the EC2 instance.
**Explanation:-**A typical scenario includes a DB instance in an Amazon VPC, that shares data with a web server running in the same VPC. Because your DB instance only needs to be available to your web server and not to the public Internet, you create a VPC with both public and private subnets. The web server is hosted in the public subnet so that it can reach the public Internet. The DB instance is hosted in a private subnet. The web server can connect to the DB instance because it is hosted within the same VPC. Still, the DB instance is not available to the public Internet, providing greater security.
Your VPC must have at least two subnets. These subnets must be in two different Availability Zones in the AWS Region where you want to deploy your DB instance. A subnet is a segment of a VPC's IP address range that you can specify, and that lets you group instances based on your security and operational needs. Your VPC must have a DB subnet group that you create. Each DB subnet group should have subnets in at least two Availability Zones in a given AWS Region. The subnets in a DB subnet group are either public or private. They can't be a mix of both public and private subnets. Your VPC must have a VPC security group that allows access to the DB instance.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_VPC.Scenarios.html
⚪ Create a DB subnet group with one private subnet and one public subnet associated with the VPC.
**Explanation:-**This optoin is incorrect. A DB subnet group cannot be a mix of both public and private subnets.
⚪ Create a DB subnet group with two public subnets associated with the VPC.
**Explanation:-**This option is incorrect. Although this is possible, it does not meet the requirements of the scenario. The database has to be configured for private access.

---

**Q33) A company wants to secure its CRM application, which uses an Amazon RDS for Oracle DB instance for database needs. The Database Specialist is in charge of enforcing the users to use SSL/TLS to connect to the Oracle database. After a successful set up to enable SSL encryption in the instance and testing the Oracle database queries from the workstation, the Database Specialist asks some users to validate the change themselves. Despite using the same database login credentials, the validation test failed and could not even connect to the database successfully. The Database Specialist investigates and confirms that the security group configuration allows network traffic to and from the application. The username and password used in the test are valid and correct.**
**What should the Database Specialist most likely consider to fix the connection failure?**

⚪ The Database Specialist needs to restart the Oracle DB instance.
**Explanation:-**This optoin is incorrect. Although restarting the instance may confirm the changes have taken effect, the test queries ran by the Specialist confirms that the database is ready to accept connections before the application test.
✅ The Database Specialist needs to configure the application to use the latest available CA root certificate and use the SSL certificate on the connection string.
**Explanation:-**Secure Sockets Layer (SSL) is an industry-standard protocol for securing network connections between client and server. After SSL version 3.0, the name was changed to Transport Layer Security (TLS), but it is still often referred to as SSL, and we refer to the protocol as SSL. Amazon RDS supports SSL encryption for Oracle DB instances. You enable SSL encryption for an Oracle DB instance by adding the Oracle SSL option to the option group associated with the DB instance. Amazon RDS uses a second port, as required by Oracle, for SSL connections. Doing this allows both clear text and SSL-encrypted communication to occur at the same time between a DB instance and an Oracle client. For example, you can use the port with clear text communication to communicate with other resources inside a VPC while using the port with SSL-encrypted communication to communicate with resources outside the VPC.
Note that you can't use both SSL and Oracle native network encryption (NNE) on the same DB instance. Before you can use SSL encryption, you must disable any other connection encryption. If you use or plan to use Secure Sockets Layer (SSL) or Transport Layer Security (TLS) with certificate verification to connect to your RDS DB instances, you require Amazon RDS CA-2019 certificates, which are enabled by default for new DB instances.
In the scenario, the Database Specialist has succeeded in testing the connection to the SSL-enabled database from the workstation. However, it does not confirm any tests from the application. The certificates may have been implemented in the Database Specialist's work tools, but not in the

application.
Reference:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html

- The Database Specialist needs to encrypt the database using Oracle TDE.

**Explanation:-**This option is incorrect. Although the company may want this eventually, the question asks how to fix the connection failure

- After enabling SSL encryption in the Oracle DB instance, the Database Specialist needs to configure the firewall using AWS WAF and allow the application to access the instance.

**Explanation:-**This option is incorrect. In this scenario, the Database Specialist already confirmed that the traffic from the application to the database works.

---

**Q34) Due to regulatory compliance, a financial services company has to archive three years of monthly database backups of a 4 TB Amazon RDS for Oracle DB instance and present it to auditors within the day the request was made.**
**Which is the most operationally efficient solution that will meet these requirements?**

✅ Write a Lambda function that takes a manual RDS snapshot every first of the month.

**Explanation:-**Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. RDS Automated snapshots can be set to max retention of 35 days, which means AWS deletes your snapshot when it is 35 days old. Most companies require backup routines that are executed in certain conditions and kept for more than a month. You can create a DB snapshot using the AWS Management Console, the AWS CLI, or the RDS API. Unlike automated backups, manual snapshots aren't subject to the backup retention period. Snapshots do not expire. For very long-term backups of MariaDB, MySQL, and PostgreSQL data, we recommend exporting snapshot data to Amazon S3.
With AWS SDK, developers can take a manual snapshot programmatically. It means developers can write a script to do this and run it in an AWS Lambda function. You can set up a rule to run an AWS Lambda function on a schedule.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/RunLambdaSchedule.html

- Write a Lambda function that takes an automated RDS snapshot every first of the month.

**Explanation:-**This option is incorrect. Automated backups are configured by modifying the RDS DB instance and can be kept only for as long as 35 days.

- Write a Lambda function that takes a manual RDS snapshot every first of the month. Move the snapshot to an Amazon S3 bucket.

**Explanation:-**This option is incorrect. At the moment, only certain database types (MySQL, MariaDB, PostgreSQL) can be exported to an Amazon S3 bucket.

- Using the RDS console, create a snapshot schedule to take a snapshot every 30 days.

**Explanation:-**This option is incorrect. Automated backups run every day. You cannot create a snapshot schedule using the RDS console. More importantly, not all months have 30 days, so this is programmatically incorrect.

---

**Q35) One of the database team members has disabled the Point-In-Time Recovery (PITR) of a DynamoDB table to remove additional charges. After three days, the member re-enabled the Point-In-Time recovery to restore the table that holds data from 15 days ago and found out that the backup made on that day no longer exists.**
**How can the database team recover the backup data?**

✅ The backup table can no longer be recovered because re-enabling the PITR after disabling it will reset the time for which you can recover that table.

**Explanation:-**Amazon DynamoDB Point-In-Time recovery (PITR) provides automatic backups of your DynamoDB table data. You can enable point-in-time recovery using the AWS Management Console, AWS Command Line Interface (AWS CLI), or the DynamoDB API. When enabled, point-in-time recovery provides continuous backups until you explicitly turn it off.
After you enable point-in-time recovery, you can restore to any point in time within EarliestRestorableDateTime and LatestRestorableDateTime. LatestRestorableDateTime is typically 5 minutes before the current time.
For LatestRestorableDateTime, you can restore your table to any point in time during the last 35 days. The retention period is a fixed 35 days (5 calendar weeks) and can't be modified. Any number of users can execute up to four concurrent restores (any type of restore) in a given account.
If you disable point-in-time recovery and later re-enable it on a table, you reset the start time for which you can recover that table. As a result, you can only immediately restore that table using the LatestRestorableDateTime.
References:
https://aws.amazon.com/dynamodb/backup-restore/
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/PointInTimeRecovery_Howitworks.html

- Restore the table using the EarliestRestorableDateTime.

**Explanation:-**This option is incorrect because once you enable the PITR, the time at which you can recover the table will be reset. All the backed-up data on the previously enabled PITR will be lost. Therefore, you will not be able to recover the data from 15 days ago.

- Identify the exact date from 15 days ago. Specify that date upon restoring the table.

**Explanation:-**This option is incorrect because all the backed up data from the previously enabled PITR is already lost when the member disabled it. Re-enabling the PITR will reset the time at which you can recover the table.

- Restore the table using the LatestRestorableDateTime.

**Explanation:-**This option is incorrect because the backed-up data needed is from 15 days ago and not the latest one. Moreover, there is no way to recover the old backups since the PITR was already disabled 3 days ago, which deleted the previous backups from that date.

---

**Q36) A company that uses a MySQL database on-premises has encountered a damaging loss after a database corruption due to a disk failure. The incident has prompted the database team to build a proof of concept for their database operations using Amazon Aurora MySQL. The team needs to demonstrate the effect of a disk failure event on the application's performance.**
**Which of the following will help the team simulate such an event?**

- Use Aurora Backtrack.

**Explanation:-**This option is incorrect because this feature is just used for moving your database to a prior point in time without needing to restore from a backup. It cannot be used to simulate a disk failure event.

✅ Use fault injection queries.

**Explanation:-**You can test the fault tolerance of your Amazon Aurora DB cluster by using fault injection queries to force a crash of an Amazon Aurora instance or an Aurora Replica. Fault injection queries are issued as SQL commands to an Amazon Aurora instance and they enable you to schedule a simulated occurrence of one of the following events:
A crash of a writer or reader DB instance
A failure of an Aurora Replica

A disk failure
Disk congestion
You can simulate a disk failure for an Aurora DB cluster using this fault injection query:
ALTER SYSTEM SIMULATE DISK FAILURE
During a disk failure simulation, the Aurora DB cluster randomly marks disk segments as faulting. Requests to those segments will be blocked for the duration of the simulation.
For the statement, you specify the index value of a specific logical block of data or storage node. However, if you specify an index value greater than the number of logical blocks of data or storage nodes used by the DB cluster volume, the statement returns an error. You can avoid that error by using this statement:
SHOW VOLUME STATUS
The above statement returns two server status variables, Disks and Nodes. These variables represent the total number of logical blocks of data and storage nodes, respectively, for the DB cluster volume.
You can also simulate the failure of an Aurora Replica using this fault injection query:
ALTER SYSTEM SIMULATE READ REPLICA FAILURE
An Aurora Replica failure will block all requests to an Aurora Replica or all Aurora Replicas in the DB cluster for a specified time interval. When the time interval completes, the affected Aurora Replicas will be automatically synced up with the master instance.

- Reboot the Aurora DB cluster and enable the Reboot with failover option.

**Explanation:-**This option is incorrect because this option is only applicable in Amazon RDS and not in Amazon Aurora. Rebooting with failover is beneficial when you want to simulate a failure of a DB instance for testing, or restore operations to the original AZ after a failover occurs.

- Stop the Aurora DB Cluster.

**Explanation:-**This option is incorrect because this will entirely stop the operation of your database, including its failover mechanism. Although you can simulate a database outage by stopping the Aurora DB cluster, it is still wrong because the simulation should be done in a database that has an available status.

---

**Q37) A company uses an Amazon RDS for MariaDB instance in a Multi-AZ deployment for its e-commerce application. To meet the compliance audit, the company's Database Specialist was required to move the primary DB instance out of its current Availability Zone (ap-southeast-2a) and restrict it to start only in Availability Zones (ap-southeast-2b) and (ap-southeast-2c) in the same Amazon VPC provided by the Network Specialist. The Database Specialist wants to avoid any database data modification and accomplish the task with minimum downtime.**
**What should the Database Specialist do to meet this requirement?**

- Create a new DB subnet group with appropriate Availability Zones. Modify the MariaDB instance and assign it to a new DB subnet group.

**Explanation:-**This option is incorrect. You cannot move an existing instance to a new DB subnet group.

- Reboot the DB instance with failover to the other AZ. Modify the existing DB subnet group to remove the subnet associated with ap-southeast-2a and add the subnet for the preferred Availability Zone.

**Explanation:-**This option is incorrect. The steps are incomplete, and the DB instance should not use the subnet related to the Availability Zone ap-southeast-2a (even flagged as part of the Multi-AZ deployment).

- ✅ Reboot the DB instance with failover to the other AZ. Modify the MariaDB instance to be a Single-AZ deployment. Modify the existing DB subnet group to remove the subnet associated with ap-southeast-2a and add the subnet for the preferred Availability Zone. Modify the DB instance back to a Multi-AZ deployment.

**Explanation:-**To launch an Amazon RDS DB instance, an RDS DB subnet group must contain at least two subnets. These subnets must be in different Availability Zones in the same AWS Region. You can remove or delete a subnet from the DB subnet group only if there are no DB instances associated with the subnet group and launched in the subnet that you're trying to delete. If you launch a DB instance with a DB subnet group that contains two subnets in two Availability Zones, then you can't delete any subnet from the DB subnet group.
If you have a Multi-AZ deployment that has two or more subnets in a subnet group, then you can launch the DB instance in any of the subnets of the two Availability Zones. If you have a Single-AZ deployment that has two or more subnets in the subnet group, then you can specify the Availability Zone when you create a DB instance. If you didn't specify the Availability Zone when you created the DB instance, then the DB instance is launched in any of the subnets of the two Availability Zones.
To delete a subnet from a DB subnet group, isolate the subnet by moving the DB instance to another subnet. Then, remove the subnet from the DB subnet group. The following steps are applicable for Amazon RDS for Oracle, PostgreSQL, MySQL, MariaDB, or SQL Server.
Reference:
https://aws.amazon.com/premiumsupport/knowledge-center/rds-db-subnet-group/

- Migrate the MariaDB instance to a new DB instance using an appropriate DB subnet group.

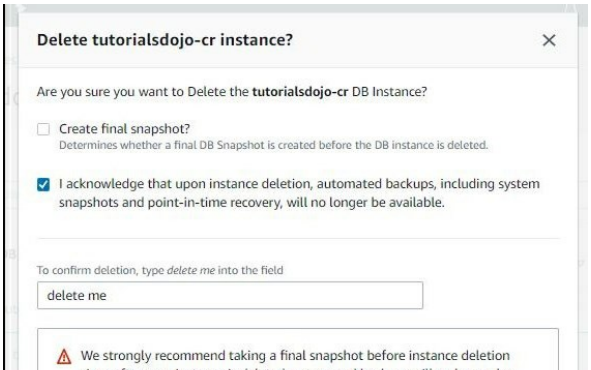
**Explanation:-**This option is incorrect. Although this works, this does not meet the scenario's requirements.

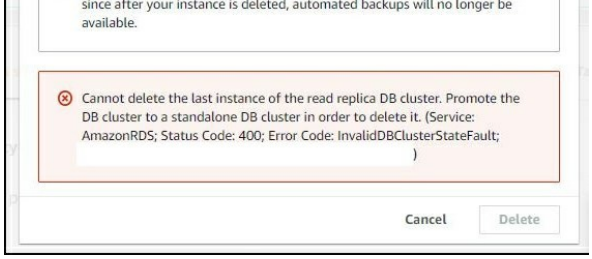
---

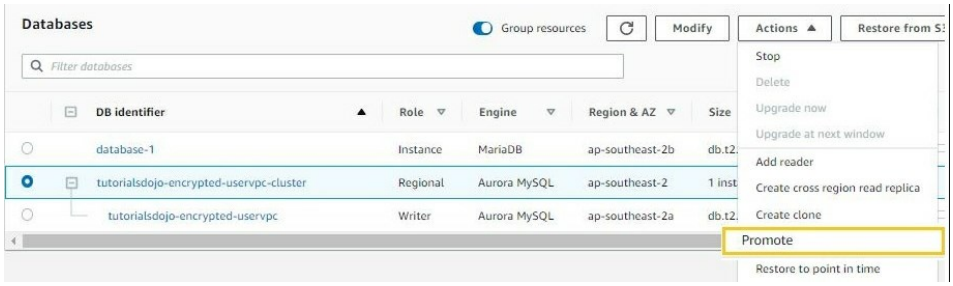
**Q38) A Database Specialist manages the costs incurred by an Amazon Aurora with MySQL compatibility cluster and has decided to delete a cross-Region read replica cluster. The attempt to remove the only instance in the read replica cluster using the RDS console fails.**
**What is the first step that the Database Specialist needs to take to delete the replica cluster successfully?**

- ✅ Promote the cross-region read replica cluster to a standalone DB cluster.

**Explanation:-**
For Aurora MySQL, you can't delete a DB instance in a DB cluster if both of the following conditions are true:
The DB cluster is a read replica of another Aurora DB cluster
The DB instance is the only instance in the DB cluster.
To delete a DB instance in this case, first promote the DB cluster so that it's no longer a read replica. After the promotion completes, you can delete the final DB instance in the DB cluster.






References:
https://aws.amazon.com/premiumsupport/knowledge-center/rds-error-delete-aurora-cluster/
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_DeleteInstance.html
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html

- Disable the deletion protection option of the source Aurora MySQL DB cluster.

**Explanation:-**This option is incorrect. The deletion protection option of the source Aurora MySQL DB cluster is irrelevant to the intention of the Database Specialist to delete the read replica cluster in the other region.

- Disable the deletion protection option of the read replica cluster.

**Explanation:-**This option is incorrect. It may be necessary once the read replica cluster becomes a standalone Aurora DB cluster. However, this will not resolve the error.

- Disable binary logging replication on the source Aurora MySQL DB cluster.

**Explanation:-**This option is incorrect because this will not resolve the issue. It will just break the synchronization process of the database environment.

---

**Q39) A company has an application that uses an Amazon DynamoDB table with a global secondary index. The Database Specialist must back up the data stored in the table to protect the system from database service disruptions.**
**Which of the following is true when the table is backed up? (Select TWO.)**

- ✅ The backup is created asynchronously.

**Explanation:-**When you create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes. All backups in DynamoDB work without consuming any provisioned throughput on the table.
DynamoDB backups do not guarantee causal consistency across items; however, the skew between updates in a backup is usually much less than a second.
While a backup is in progress, you cannot do the following:
Pause or cancel the backup operation.
Delete the source table of the backup.
Disable backups on a table if a backup for that table is in progress.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Backup.Tutorial.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html

- ✅ DynamoDB backups do not guarantee causal consistency across items.

**Explanation:-**When you create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes. All backups in DynamoDB work without consuming any provisioned throughput on the table.
DynamoDB backups do not guarantee causal consistency across items; however, the skew between updates in a backup is usually much less than a second.
While a backup is in progress, you cannot do the following:
Pause or cancel the backup operation.
Delete the source table of the backup.
Disable backups on a table if a backup for that table is in progress.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Backup.Tutorial.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html

- The backup is created synchronously.
- You can pause or cancel the backup operation.
- DynamoDB consumes the provisioned throughput of the table.

---

**Q40) A company is running an application using DynamoDB in provisioned mode. The manager instructed you to gather and analyze the usage and cost of the different AWS services for the past three months. You noticed that DynamoDB is responsible for almost half of the total cost of your account.**
**How can you lower the operating cost of the DynamoDB database?**

- Modify the Adaptive capacity

**Explanation:-**This option is incorrect. These are built-in features in DynamoDB and are fully managed by AWS, which means you don't have any control over them.

- Enable Burst capacity

**Explanation:-**This option is incorrect. These are built-in features in DynamoDB and are fully managed by AWS, which means you don't have any

control over them.
Use Spot capacity
**Explanation:-**This option is incorrect because this is only available in Amazon EC2 instances.
Use Reserved capacity
**Explanation:-**For provisioned mode tables, you specify throughput capacity in terms of read capacity units (RCUs) and write capacity units (WCUs):
The provisioned mode is a good option if any of the following are true:
You have predictable application traffic.
You run applications whose traffic is consistent or ramps gradually.
You can forecast capacity requirements to control costs.
You can reduce the cost of your provisioned DynamoDB by purchasing reserved capacity in advance. By reserving your read and write capacity units ahead of time, you will get significant cost savings compared to on-demand provisioned throughput settings. Any capacity that you provision in excess of your reserved capacity is billed at standard provisioned capacity rates.
Provisioned throughput is the maximum amount of capacity that an application can consume from a table or index. If your application exceeds your provisioned throughput capacity on a table or index, it is subject to request throttling.
Throttling prevents your application from consuming too many capacity units. When a request is throttled, it fails with an HTTP 400 code (Bad Request) and a ProvisionedThroughputExceededException. The AWS SDKs have built-in support for retrying throttled requests, so you do not need to write this logic yourself.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-design.html#bp-partition-key-partitions-adaptive-boost

---

**Q41) A Database Specialist needs to decommission a production Amazon Aurora PostgreSQL database cluster. Using the Amazon RDS console, the Specialist attempts to delete the last DB instance in the cluster. However, the action throws out an error and fails.**
**What is most likely causing the issue?**

The Aurora Cluster has deletion protection option enabled.
**Explanation:-**One can delete a DB instance in a DB cluster, including removing the primary DB instance in a DB cluster or an Amazon Aurora Replica. You can use the Amazon Relational Database Service (Amazon RDS) console or the AWS Command Line Interface (AWS CLI) to delete an Aurora cluster. To delete a DB instance, you must specify the name of the instance.
To delete an Aurora cluster using the AWS CLI, you must first delete all instances inside the cluster. After you delete all instances inside a cluster, you can then delete the cluster by using delete-db-cluster. If you delete the last instance in the cluster using the Amazon RDS console, the empty cluster is also automatically deleted. If you have a cluster with only one instance and delete that instance using the Amazon RDS console, then both that instance and the cluster are deleted.
You can enable deletion protection so that users can't delete a DB cluster. Aurora enforces deletion protection for a DB cluster, whether you perform the operation from the console, the CLI, or the API. If you try to delete a DB cluster that has deletion protection enabled, you can't do so. To be certain that you can delete the cluster, modify the cluster, and disable deletion protection.
Since there is only one remaining instance in the cluster, and the Database Specialist attempted to delete it using the console, it will try to delete the cluster, too.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/rds-error-delete-aurora-cluster/
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_DeleteInstance.html#USER_DeleteInstance.NoSnapshot
The last instance can only be removed by deleting the whole Aurora cluster using the delete-db-cluster command.
**Explanation:-**This option is incorrect because the Aurora cluster can only be removed once all instances are removed.
Deleting the last instance of the Aurora cluster can only be done using AWS CLI.
**Explanation:-**This option is incorrect. One can use the Amazon RDS console to delete an instance of the cluster, provided it can also delete the cluster.
The Aurora Cluster needs to be stopped first before deleting it.
**Explanation:-**This option is incorrect. The instance has to be in available status before it can be deleted.

---

**Q42) A company has a suite of applications that consist of Amazon Aurora database, Lambda functions, DynamoDB tables, and CloudWatch Alarms that are deployed to various environments. The Application team needs an automated deployment and configuration method that will standardize its core components and minimize rework due to manual configuration errors. The team should be able to manage the environment-specific settings separately.**
**What is the MOST suitable and secure solution that the Database Specialist should implement to meet these requirements?**

Upload all of the environment-specific parameters and security credentials in an Amazon S3 bucket. Design the AWS CloudFormation template to reference the values from the S3 bucket dynamically. Set the environment name as a parameter when deploying the CloudFormation stack to populate its related parameters.
**Explanation:-**This option is incorrect. Although this solution could work, storing sensitive credentials on Amazon S3 is a security risk. A more secure way to implement this is to use AWS Systems Manager Parameter Store instead.
Set up a parameterized AWS CloudFormation template with a Mapping property that sets the values of the required AWS resources dynamically. Log in to the AWS Management Console and import the template to CloudFormation as a stack. Manually modify the parameters before deploying the stack to a specific environment.
**Explanation:-**This option is incorrect because this solution entails a lot of manual tasks. Take note that the scenario asked for an automated deployment and configuration method.
Use AWS CodeCommit to store the environment-specific parameters. Design the AWS CloudFormation template to reference the values from the CodeCommit dynamically. Use the environment name as a parameter when deploying the CloudFormation stack to populate its related parameters.
**Explanation:-**This option is incorrect because AWS CodeCommit is simply a fully-managed source control service that makes it easy for companies to host secure and highly scalable private Git repositories. This service is not suitable to be used to store parameters for your CloudFormation stacks.
Store the environment-specific parameters in the AWS Systems Manager Parameter Store. Design the AWS CloudFormation template to reference the values from the Parameter Store dynamically. Use the environment name as a parameter when deploying the CloudFormation stack to populate its related parameters.
**Explanation:-**AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, Amazon Machine Image (AMI) IDs, and license codes as parameter values. You can store values as plain text or encrypted data. You can reference Systems Manager parameters in your scripts, commands, SSM documents, and configuration and automation workflows by using the unique name that you specified when you created the parameter.
Parameter Store offers the following benefits and features.
Use a secure, scalable, hosted secrets management service with no servers to manage.

Improve your security posture by separating your data from your code.
Store configuration data and encrypted strings in hierarchies and track versions.
Control and audit access at granular levels.
Configure change notifications and trigger automated actions for both parameters and parameter policies.
Tag parameters individually, and then secure access from different levels, including operational, parameter, Amazon EC2 tag, and path levels.
Validation of ID format when you specify an Amazon Machine Image (AMI) ID as a parameter value.
Reference AWS Secrets Manager secrets by using Parameter Store parameters.
Use Parameter Store parameters with other Systems Manager capabilities and AWS services to retrieve secrets and configuration data from a central store.
Parameters work with Systems Manager capabilities such as Run Command, State Manager, and Automation. You can also reference parameters in a number of other AWS services, including the following:
Amazon Elastic Compute Cloud (Amazon EC2)
Amazon Elastic Container Service (Amazon ECS)
AWS Secrets Manager
AWS Lambda
AWS CloudFormation
AWS CodeBuild
AWS CodePipeline
AWS CodeDeploy
References:
https://aws.amazon.com/blogs/mt/integrating-aws-cloudformation-with-aws-systems-manager-parameter-store
https://aws.amazon.com/premiumsupport/knowledge-center/cloudformation-systems-manager-parameter/

---

**Q43) A Database Specialist needs to deploy a new CloudFormation stack for an application that uses an Amazon RDS database. For data compliance purposes, the template should be designed in a way that prevents accidental deletion or data loss in the database. If the stack is deleted, AWS CloudFormation must keep the database resource without any changes.**
**Which of the following should be implemented to meet this requirement? (Select THREE.)**

- Verify that there is no attached DeletionPolicy attribute to the RDS resource.
- ✅ Set the DeletionProtection attribute of the AWS CloudFormation template to true.

**Explanation:-**With the DeletionPolicy attribute, you can preserve or (in some cases) backup a resource when its stack is deleted. You specify a DeletionPolicy attribute for each resource that you want to control. If a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource by default.
Delete - CloudFormation deletes the resource including its content if applicable during stack deletion.
Retain - CloudFormation keeps the resource without deleting the resource or its contents when its stack is deleted
Snapshot - For resources that support snapshots, AWS CloudFormation creates a snapshot for the resource before deleting it.
To keep a resource when its stack is deleted, specify Retain for that resource. You can use retain for any resource. For example, you can retain a nested stack, Amazon S3 bucket, or EC2 instance so that you can continue to use or modify those resources after you delete their stacks. For resources that support snapshots, such as AWS::EC2::Volume, specify Snapshot to have AWS CloudFormation create a snapshot before deleting the resource.
Note that this capability also applies to stack update operations that lead to resources being deleted from stacks. For example, if you remove the resource from the stack template and then update the stack with the template. This capability does not apply to resources whose physical instance is replaced during stack update operations. For example, if you edit a resource's properties such that AWS CloudFormation replaces that resource during a stack update.
DeleteAutomatedBackup - A value that indicates whether to remove automated backups immediately after the DB instance is deleted. This parameter isn't case-sensitive. The default is to remove automated backups immediately after the DB instance is deleted
DeletionProtection - A value that indicates whether the DB instance has deletion protection enabled. The database can't be deleted when deletion protection is enabled. By default, deletion protection is disabled.
References:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbcluster.html

- Set the DeletionPolicy attribute of the template to Snapshot.

**Explanation:-**This option is incorrect because, with this setting, CloudFormation will create a snapshot of the resource before deleting it. This means that the database resources will still be affected.

- ✅ Ensure that the DeleteAutomatedBackups property of the AWS::RDS::DBInstance resource is set to false.

**Explanation:-**With the DeletionPolicy attribute, you can preserve or (in some cases) backup a resource when its stack is deleted. You specify a DeletionPolicy attribute for each resource that you want to control. If a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource by default.
Delete - CloudFormation deletes the resource including its content if applicable during stack deletion.
Retain - CloudFormation keeps the resource without deleting the resource or its contents when its stack is deleted
Snapshot - For resources that support snapshots, AWS CloudFormation creates a snapshot for the resource before deleting it.
To keep a resource when its stack is deleted, specify Retain for that resource. You can use retain for any resource. For example, you can retain a nested stack, Amazon S3 bucket, or EC2 instance so that you can continue to use or modify those resources after you delete their stacks. For resources that support snapshots, such as AWS::EC2::Volume, specify Snapshot to have AWS CloudFormation create a snapshot before deleting the resource.
Note that this capability also applies to stack update operations that lead to resources being deleted from stacks. For example, if you remove the resource from the stack template and then update the stack with the template. This capability does not apply to resources whose physical instance is replaced during stack update operations. For example, if you edit a resource's properties such that AWS CloudFormation replaces that resource during a stack update.
DeleteAutomatedBackup - A value that indicates whether to remove automated backups immediately after the DB instance is deleted. This parameter isn't case-sensitive. The default is to remove automated backups immediately after the DB instance is deleted
DeletionProtection - A value that indicates whether the DB instance has deletion protection enabled. The database can't be deleted when deletion protection is enabled. By default, deletion protection is disabled.
References:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbcluster.html

- ✅ Configure the DeletionPolicy attribute to Retain.

**Explanation:-**With the DeletionPolicy attribute, you can preserve or (in some cases) backup a resource when its stack is deleted. You specify a

DeletionPolicy attribute for each resource that you want to control. If a resource has no DeletionPolicy attribute, AWS CloudFormation deletes the resource by default.
Delete - CloudFormation deletes the resource including its content if applicable during stack deletion.
Retain - CloudFormation keeps the resource without deleting the resource or its contents when its stack is deleted
Snapshot - For resources that support snapshots, AWS CloudFormation creates a snapshot for the resource before deleting it.
To keep a resource when its stack is deleted, specify Retain for that resource. You can use retain for any resource. For example, you can retain a nested stack, Amazon S3 bucket, or EC2 instance so that you can continue to use or modify those resources after you delete their stacks. For resources that support snapshots, such as AWS::EC2::Volume, specify Snapshot to have AWS CloudFormation create a snapshot before deleting the resource.
Note that this capability also applies to stack update operations that lead to resources being deleted from stacks. For example, if you remove the resource from the stack template and then update the stack with the template. This capability does not apply to resources whose physical instance is replaced during stack update operations. For example, if you edit a resource's properties such that AWS CloudFormation replaces that resource during a stack update.
DeleteAutomatedBackup - A value that indicates whether to remove automated backups immediately after the DB instance is deleted. This parameter isn't case-sensitive. The default is to remove automated backups immediately after the DB instance is deleted
DeletionProtection - A value that indicates whether the DB instance has deletion protection enabled. The database can't be deleted when deletion protection is enabled. By default, deletion protection is disabled.
References:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbcluster.html
○ Ensure that the MultiAZ property of the AWS::RDS::DBInstance resource is set to true.
**Explanation:-**This option is incorrect because this property simply specifies whether the database instance is a multiple Availability Zone deployment or not. It doesn't direct CloudFormation to keep the database resources without any changes.

---

**Q44) A Database Specialist is setting up and deploying an application that uses an Amazon Aurora PostgreSQL DB cluster. The database must have a minimal application downtime during a failover.**
**What approach can accomplish this requirement?**

✓ Enable Aurora DB cluster cache management in the associated parameter group. Set the TCP keepalive parameters for the DB and the application client to a low value.
**Explanation:-**For fast recovery of the writer DB instance in your Aurora PostgreSQL clusters if there's a failover, use cluster cache management for Amazon Aurora PostgreSQL. Cluster cache management ensures that application performance is maintained if there's a failover.
In a typical failover situation, you might see a temporary but large performance degradation after failover. This degradation occurs because when the failover DB instance starts, the buffer cache is empty. An empty cache is also known as a cold cache. A cold cache degrades performance because the DB instance has to read from the slower disk, instead of taking advantage of values stored in the buffer cache.
With cluster cache management, you set a specific reader DB instance as the failover target. Cluster cache management ensures that the data in the designated reader's cache is kept synchronized with the data in the writer DB instance's cache. The designated reader's cache with prefilled values is known as a warm cache. If a failover occurs, the designated reader uses values in its warm cache immediately when it's promoted to the new writer DB instance. This approach provides your application with better recovery performance.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.cluster-cache-mgmt.html
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.BestPractices.html
○ Enable Aurora DB cluster cache management in the associated option group. Configure both the TCP keepalive parameters for the DB and the application client to a high value.
**Explanation:-**This option is incorrect. Although enabling Aurora DB cluster cache management is right, it is still wrong to set the TCP keepalive parameters to a high value. You should set this parameter with a low value to minimize the application downtime.
○ Set the TCP keepalive parameter in the associated parameter group of the PostgreSQL DB cluster to 3600. Use the same TCP keepalive value for the application client.
**Explanation:-**This option is incorrect because setting the keepalive parameter to 3600 means that the application will wait for a whole hour (3600 seconds = 1 hour) before it can detect a database failover. You also have to enable Aurora DB cluster cache management to satisfy the scenario.
○ Enable database activity streams to reduce application downtime during a failover. Set the TCP keepalive parameter in the associated parameter group of the PostgreSQL DB cluster to a low value but configure the keepalive value for the application client to high.
**Explanation:-**This option is incorrect because the database activity streams feature only provides a near real-time data stream of the database activity in your relational database to help you monitor activity. In addition, you have to set the TCP keepalive value to a low value on both the database and application side to minimize the failover latency.

---

**Q45) A company has an enterprise application hosted in an Amazon ECS cluster and uses Amazon RDS for MySQL database with a read replica. The application uses the read replica to generate business reports and avoid performance issues on the primary DB instance. However, the users noticed that the queries to the read replica have a slow response time.**
**Which of the following is the MOST likely root cause of this issue?**

○ The replica DB instance class is higher than the primary DB instance.
**Explanation:-**This option is incorrect because this will actually improve the performance of the read replica and avoid replication delays. This will only be an issue if the read DB instance class is lower than the primary DB instance.
○ The change data capture (CDC) option is enabled on the primary DB instance for continuous data replication.
**Explanation:-**This option is incorrect because Amazon RDS doesn't have a built-in CDC option. This is only applicable for AWS DMS.
✓ There are long-running queries on the primary DB instance.
**Explanation:-**Amazon RDS for MySQL uses asynchronous replication and sometimes the replica isn't able to keep up with the primary DB instance. This can cause a replication lag.
When using Amazon RDS for MySQL read replica with binary log file position based replication, you can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS ReplicaLag metric. The ReplicaLag metric reports the value of the Seconds_Behind_Master field of the SHOW SLAVE STATUS command.
MySQL replication works with three threads: the Binlog Dump thread, the IO_THREAD, and the SQL_THREAD. If there is a delay in replication, first identify whether the lag is caused by the replica IO_THREAD or the replica SQL_THREAD. Then, you can identify the root cause of the lag.
Normally, IO_THREAD doesn't cause large replication delays, because the IO_THREAD only reads the binary logs from the master. However, network connectivity and network latency can affect the speed of the reads between the servers. The replica IO_THREAD could be slow due to high bandwidth usage.
If the replica SQL_THREAD is the source of replication delays, those delays could be due to the following reasons:

Long-running queries on the primary DB instance
Insufficient DB instance class size or storage
Parallel queries executed on the primary DB instance
Binary logs synced to the disk on the replica DB instance
Binlog_format on the replica is set to ROW
Replica creation lag
Long-running queries on the primary DB instance that take an equal amount of time to run on the replica DB instance can increase the Seconds_Behind_Master. For example, if you executed a change on the primary DB instance and it takes one hour to execute, by the time that change starts running on the replica, the lag is one hour. Because the change might also take one hour to complete on the replica, by the time the change is complete, the total lag is approximately two hours. This delay is expected, but you can minimize this lag by monitoring the slow query log on the master. You can also identify long-running statements to reduce lag. Then, break long-running statements into smaller statements or transactions.

- The replica lag metric has reached a value of 0.

**Explanation:-**This option is incorrect because this metric represents the amount of time a read replica DB instance lags behind the source DB instance. If this is 0, then it means that the read replica is fully up-to-date.

---

**Q46) A Database Specialist is setting up a disaster recovery solution for an Amazon Redshift cluster with database encryption enabled using AWS KMS. The backup cluster must be highly available and fault-tolerant even in the event of an AWS region outage.**
**Which of the following is the most suitable solution that meets these requirements?**

- ✅ Launch and configure a snapshot copy grant for a master key in another AWS region. Enable cross-region snapshots in the Redshift cluster to copy snapshots of the cluster to the other region.

**Explanation:-**Snapshots are point-in-time backups of a cluster. There are two types of snapshots: automated and manual. Amazon Redshift stores these snapshots internally in Amazon S3 by using an encrypted Secure Sockets Layer (SSL) connection. Amazon Redshift automatically takes incremental snapshots that track changes to the cluster since the previous automated snapshot.
Automated snapshots retain all of the data required to restore a cluster from a snapshot. You can take a manual snapshot any time. When you restore from a snapshot, Amazon Redshift creates a new cluster and makes the new cluster available before all of the data is loaded, so you can begin querying the new cluster immediately. The cluster streams data on demand from the snapshot in response to active queries, then loads the remaining data in the background.
When you launch an Amazon Redshift cluster, you can choose to encrypt it with a master key from the AWS Key Management Service (AWS KMS). AWS KMS keys are specific to a region. If you want to enable cross-region snapshot copy for an AWS KMS-encrypted cluster, you must configure a snapshot copy grant for a master key in the destination region so that Amazon Redshift can perform encryption operations in the destination region.

- Enable Concurrency Scaling to scale out the Redshift cluster to two or more AWS regions.

**Explanation:-**This option is incorrect because this feature is primarily used to support virtually unlimited concurrent users and concurrent queries, with consistently fast query performance. The scenario doesn't warrant the use of this feature since the objective is to ensure the high availability of the Redshift cluster by having a cross-region snapshot to another AWS Region.

- Enable the cross-region snapshot copy feature to copy snapshots to the other AWS region and use the existing key from the source region.

**Explanation:-**This option is incorrect because the master keys being used by Redshift are regional in scope. You can't use a key from another region for your backup Redshift cluster. You have to configure a snapshot copy grant for a master key in the other AWS region.

- Use Redshift Spectrum to scale out the Redshift cluster to two or more AWS regions. The Redshift cluster in the other region will automatically activate when there is an outage in the primary region.

**Explanation:-**This option is incorrect because Redshift Spectrum simply allows you to query data directly from files hosted in Amazon S3. This feature will not improve the fault-tolerance of the Redshift cluster from an AWS Region outage.

---

**Q47) A Database Specialist runs various tests to a write-intensive batch application that uses an Amazon Aurora PostgreSQL database. The application issues a large number of COMMIT and ROLLBACK commands to the database to persist data and roll back recent changes. Before deploying the application to production, the Specialist needs to ensure that there is no performance impact in processing the workload.**
**Which of the following metric should the Specialist monitor and track?**

- LWLock:buffer_content

**Explanation:-**This option is incorrect because this wait event metric only tracks the sessions that are trying to modify data that has been modified by another session and are waiting for the other session's transaction to be committed or rolled back.

- Lock:transactionid
- ✅ IO:XactSync

**Explanation:-**You manage your Amazon Aurora DB cluster in the same way that you manage other Amazon RDS DB instances, by using parameters in a DB parameter group. Amazon Aurora differs from other DB engines in that you have a DB cluster that contains multiple DB instances. As a result, some of the parameters that you use to manage your Amazon Aurora DB cluster apply to the entire cluster, while other parameters apply only to a particular DB instance in the DB cluster.
Cluster-level parameters are managed in DB cluster parameter groups. Instance-level parameters are managed in DB parameter groups. Although each DB instance in an Aurora PostgreSQL DB cluster is compatible with the PostgreSQL database engine, some of the PostgreSQL database engine parameters must be applied at the cluster level, and are managed using DB cluster parameter groups. Cluster-level parameters are not found in the DB parameter group for a DB instance in an Aurora PostgreSQL DB cluster.
In the IO:XactSync wait event, a session is issuing a COMMIT or ROLLBACK, requiring the current transaction's changes to be persisted. Aurora is waiting for Aurora storage to acknowledge persistence.
This wait most often arises when there is a very high rate of commit activity on the system. You can sometimes alleviate this wait by modifying applications to commit transactions in batches. You might see this wait event at the same time as CPU waits in a case where the DB load exceeds the number of virtual CPUs (vCPUs) for the DB instance. In this case, the storage persistence might be competing for CPU with CPU-intensive database workloads. To alleviate this scenario, you can try reducing those workloads, or scaling up to a DB instance with more vCPUs.
References:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Reference.html
https://aws.amazon.com/blogs/database/analyzing-amazon-rds-database-workload-with-performance-insights/
https://www.postgresql.org/docs/10/monitoring-stats.html#WAIT-EVENT-TABLE

- LWLock:SubtransControlLock

**Explanation:-**This option is incorrect because this metric just tracks the sessions that lookup or manipulate the parent/child relationship between a transaction and a subtransaction.

---

**Q48) A Database Specialist is migrating multiple applications and databases from the on-premises data center to AWS Cloud.**

**One of the applications requires a heterogeneous database migration where an on-premises Oracle database needs to be migrated and transformed into an Amazon RDS for PostgreSQL database. Schema and code transformation must be done to migrate the data to the Amazon RDS successfully.**
**What is the most suitable approach to migrate this database in AWS?**

Convert the source schema using the change data capture (CDC) option of the AWS Database Migration Service (DMS) to match the target database. Migrate the data from the on-premises source database to an Amazon RDS for PostgreSQL database using the AWS DMS. Continuous data replication is enabled by default.
**Explanation:-**This option is incorrect because you can't convert the source schema using the change data capture (CDC) option. You have to use AWS SCT instead. Moreover, continuous data replication is not enabled by default. Take note that the CDC option of AWS DMS is primarily used to replicate ongoing changes from the source database to the target.
✅ Convert the source schema using the AWS Schema Conversion Tool (SCT) to match the target database. Migrate the data from the on-premises source database to an Amazon RDS for PostgreSQL database using the AWS Database Migration Service (DMS) with the change data capture (CDC) option enabled for continuous data replication.
**Explanation:-**AWS Database Migration Service helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from the most widely used commercial and open-source databases.
AWS Database Migration Service can migrate your data to and from most of the widely used commercial and open-source databases. It supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle to Amazon Aurora. Migrations can be from on-premises databases to Amazon RDS or Amazon EC2, databases running on EC2 to RDS, or vice versa, as well as from one RDS database to another RDS database. It can also move data between SQL, NoSQL, and text-based targets.
In heterogeneous database migrations, the source and target databases engines are different, like in the case of Oracle to Amazon Aurora, Oracle to PostgreSQL, or Microsoft SQL Server to MySQL migrations. In this case, the schema structure, data types, and database code of source and target databases can be quite different, requiring a schema and code transformation before the data migration starts. That makes heterogeneous migrations a two-step process.
References:
https://aws.amazon.com/dms/
https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Introduction.html
Use AWS Database Migration Service (DMS) to convert the source schema to match that of the target database. Migrate the data using the AWS Schema Conversion Tool (SCT) from the source database to an Amazon RDS for PostgreSQL database. Continuous data replication is enabled by default.
**Explanation:-**This option is incorrect because you have to use AWS SCT to convert the source schema then perform the migration using AWS DMS and not the other way around. In addition, continuous data replication is actually disabled by default as you have to configure the CDC first.
Convert the source schema using the AWS Schema Conversion Tool (SCT) to match the target database. Using AWS DataPipeline, migrate the data from the on-premises source database to an Amazon RDS for PostgreSQL database. Enable the change data capture (CDC) option in AWS DataPipeline for continuous data replication.
**Explanation:-**This option is incorrect because AWS DataPipeline doesn't have a change data capture (CDC) option. This is only available in AWS DMS.

---

**Q49) A Database Specialist is managing a DynamoDB table that holds sensor data. The table has provisioned read/write capacity units, local secondary indexes, global secondary indexes, and an associated IAM policy. Every recorded data is set to expire after 7 days using the Time to Live (TTL) feature. An update to the applicationâ€™s code has caused the DynamoDB table to have inconsistent records and needs to be restored from a backup.**
**What must the Database Specialist do after restoring the table? (Select TWO.)**

✅ Re-apply the Time to Live (TTL) settings to the restored table.
**Explanation:-**When you create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes.
When you do a restore, you can change the following table settings:
Global secondary indexes (GSIs)
Local secondary indexes (LSIs)
Billing mode
Provisioned read and write capacity
Encryption settings
However, some settings are not carried over on the restored table and you must manually configure them after restoring.
You must manually set up the following on the restored table:
Auto scaling policies
AWS Identity and Access Management (IAM) policies
Amazon CloudWatch metrics and alarms
Tags
Stream settings
Time to Live (TTL) settings
Since the DynamoDB table has an existing associated IAM policy and is using the Time To Live (TTL) feature, the Database Specialist must manually re-apply those two settings on the restored table to complete the restoration.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Backup.Tutorial.html
Associate the existing global secondary indexes (GSIs) to the restored table.
✅ Associate the existing Identity and Access Management (IAM) policy to the restored table.
**Explanation:-**When you create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes.
When you do a restore, you can change the following table settings:
Global secondary indexes (GSIs)
Local secondary indexes (LSIs)
Billing mode
Provisioned read and write capacity
Encryption settings
However, some settings are not carried over on the restored table and you must manually configure them after restoring.

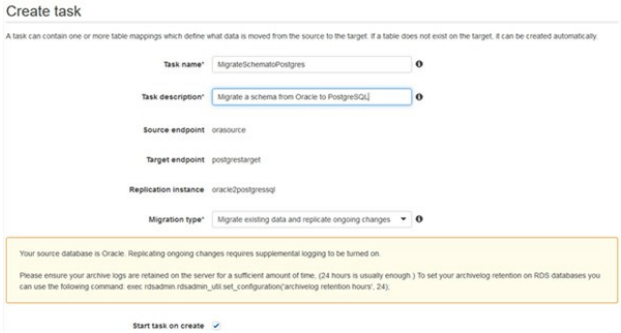
You must manually set up the following on the restored table:
Auto scaling policies
AWS Identity and Access Management (IAM) policies
Amazon CloudWatch metrics and alarms
Tags
Stream settings
Time to Live (TTL) settings
Since the DynamoDB table has an existing associated IAM policy and is using the Time To Live (TTL) feature, the Database Specialist must manually re-apply those two settings on the restored table to complete the restoration.
References:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Backup.Tutorial.html

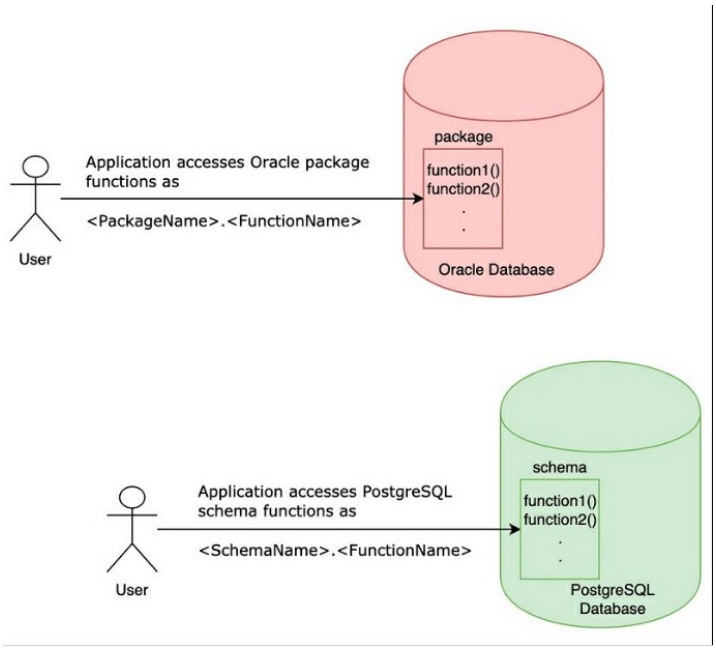
- Re-apply the provisioned read and write capacity to the restored table.
- Associate the existing local secondary indexes (LSIs) to the restored table.

---

**Q50) A Database Specialist is planning to migrate its existing on-premises Oracle database to Amazon RDS for PostgreSQL DB instance and intends to use AWS Database Migration Service to minimize downtime. The Manager wants to validate the migration of data from the source to the target once it completes, and ensure it was successful before the cutover. The Specialist does not want to impact the performance of the Oracle database.**
**Which approach will MOST effectively meet these requirements?**

✅ Create the AWS DMS task with Enable validation and Enable CloudWatch logs settings turned on.
**Explanation:-**
AWS Database Migration Services (DMS) provides support for data validation to ensure that your data was migrated accurately from the source to the target. If you enable it for a task, AWS DMS begins comparing the source and target data immediately after a full load is performed for a table, and reports any mismatches. Also, for a CDC-enabled task, AWS DMS compares the incremental changes and reports any mismatches. Data validation can be done during:
- A new AWS DMS task
- An existing AWS DMS task
- A completed AWS DMS task
- Revalidation from the table statistics section of the AWS DMS task
During data validation, AWS DMS compares each row in the source with its corresponding row at the target and verifies that those rows contain the same data. AWS DMS issues appropriate queries to retrieve the data to accomplish the validation. Note that these queries will consume additional resources at the source and the target and additional network resources.




References:
https://aws.amazon.com/premiumsupport/knowledge-center/validation-feature-dms/
https://aws.amazon.com/blogs/database/validating-database-objects-after-migration-using-aws-sct-and-aws-dms/

- Use the table metrics of the AWS DMS task to verify that the data definition language (DDL) statements are completed.

**Explanation:-**This option is incorrect. It would monitor the progress of the task but not the data transfer.

- Use the AWS Schema Conversion Tool (AWS SCT) and create a database migration assessment report.

**Explanation:-**This option is incorrect. It is performed ideally before the migration itself.

- Use the AWS Schema Conversion Tool (AWS SCT) to validate the schema conversion between Oracle and PostgreSQL objects. Verify the datatype of the columns.

**Explanation:-**This option is incorrect. It does not validate the data that was migrated successfully to the target database.