RDS Optimization

Summary of recommendations and solution approach

Documentation and guide for the implementation of RDS instances convertible plans within 2 business days

Things to Consider:

Key pain points:
Confused on Saving Plans, Reserved Instance, and Convertible modes a plans for RDS

Areas requiring urgent attention (e.g., cost overruns, inefficient architectures):
→ Cost Overruns in RDS (RDS Instance R7 family, 128 RAM, 16 VCPU)

Cost-related incidents or overspending scenarios in the past:
→ They claim RDS is incurring extra cost ($4000 claimed)

Application Nature: CRM

Areas where costs are difficult to control or predict
→ RDS Cost is exceptionally high for them, according to the client

Introduction:

As per the meeting notes, we know that the nature of the application is CRM. They have used monitoring tools like AWS Cost Explorer, Budgets, and CloudWatch for usage and monitoring. The infrastructure and application are deployed using AWS Services like EC2, K8s, and CRM. Workload is containerized and EC2-based for dev and stage. The client's key pain points are related to cloud usage, cost management, and are unable to decide on saving plans: use of reserved instances or convertible modes for RDS. This document provides a guided solution for the optimal cost-saving implementation of RDS.

Problem Statement:

The major issue is the high cost incurred by RDS, where the monthly average cost is $4000. The RDS instance is R7 family with 128GB RAM and 16 vCPUs. This means they are using **db.r7g.4xlarge**, which is based on x64 ARM-based CPU architecture.

Cost Tracking and Alerts:

The client is puzzled about the RDS instance saving plan and the implementation of convertible plans for RDS instances. Since the client is using AWS Cost Explorer, AWS Budgets, and CloudWatch for monitoring, we can use the following cloud financial management tools for cost tracking and alerts:

Tool Name	Provider	Key Features	Cost Model	Integration	Scalability	Best For
AWS Cost Explorer	AWS Native	• Basic cost visualization • 12-month historical data • Simple forecasting	Free	Native AWS integration	Low	**Crawl Stage**: Ad-hoc cost tracking
AWS Budgets	AWS Native	• Custom budget alerts • Threshold notifications (e.g., 80% of forecast)	Free	Works with SNS, Slack	Medium	**Crawl/ Walk Stage**: Basic alerts
AWS CloudWatch	AWS Native	• Real-time monitoring of AWS resources	Free (for EC2, Lambda, etc.)	• Native integration.	High	Real-time infrastructure

		and applications • Log management and analysis • Custom metrics creation	Pay-as-you-go (custom metrics, alarms)	• Third-party integrations via AWS SDKs and APIs		and application monitoring

The above AWS tools help us track costs and utilize alerting tools available natively.

Solution and Recommendation approach:

Reserved Instances

As stated above, the client is using **RDS (R7 family, 128GB RAM, 16 vCPUs)** and is incurring $**4,000 average monthly costs**. Amazon RDS's **"Reserved Instances"** is a feature using which we can save costs by committing to a specific instance type, region, and term (either 1 or 3 years). Reserved DB instances are not physical instances, but rather a billing discount applied to the use of certain on-demand DB instances in our account. Discounts for reserved DB instances are tied to instance type and AWS Region. While they don't reserve capacity (except zonal RIs), they apply a billing discount when our usage matches the RI's specifications. However, RDS Reserved Instances do offer instance size flexibility within the same instance family.

RDS Instance Size Flexibility:

RDS **does not support** convertible reserved instances, but the RDS RIs offer size flexibility for the following database engines:

- MySQL
- MariaDB
- PostgreSQL
- Amazon Aurora and Oracle (BYOB – Bring Your Own License edition).

When we purchase a reserved DB instance, we can specify the instance class. So, if we have a DB instance and need to scale it to a larger capacity, the reserved DB instance is automatically applied to the scaled DB instance. It means that our reserved DB instances are automatically

applied across all DB instance class sizes. This means that the discounted RI rate will **apply to usage of any size in the same instance family and** can only scale in their instance class type. For example: a db.m5.2xlarge MySQL RI can be automatically applied to 2 db.m5.xlarge instances.

The general process for working with reserved DB instances is:

1. Acquire the Information about the available reserved DB instance.
2. After that, purchase a reserved DB instance offering.
3. Lastly, we collect information about our existing reserved DB instances.

Procedure:

- When we purchase a reserved DB instance in Amazon RDS, we purchase a commitment to getting a discounted rate on a specific DB instance type for the duration of the reserved DB instance.
- To use an Amazon RDS reserved DB instance, we create a new DB instance. This DB instance must have the same specifications as the reserved DB instance for the following:
  a. AWS Region
  b. DB engine (The DB engine's version number doesn't need to match.)
  c. DB instance type
  d. DB instance size (RDS for Microsoft SQL Server and Amazon RDS for Oracle License Included)
  e. Edition (RDS for SQL Server and RDS for Oracle)
  f. License type (license-included or bring-our-own-license)

- If the specifications of the new DB instance match an existing reserved DB instance for our account, we are billed at the discounted rate offered for the reserved DB instance.
- If not, the DB instance is billed at an on-demand rate. We can modify a DB instance that we're using as a reserved DB instance. If the modification is within the specifications of the reserved DB instance, part or all of the discount still applies to the modified DB instance.
- If the modification is outside the specifications, such as changing the instance class, the discount no longer applies.

We can compare usage for different reserved DB instance sizes by using normalized units per hour. For example, one unit of usage on two db.r3.large DB instances is equivalent to eight

normalized units per hour of usage on one db.r3.small. The following table shows the number of normalized units per hour for each DB instance size.

Instance size	Single-AZ normalized units per hour (deployment with one DB instance)	Multi-AZ DB instance normalized units per hour (deployment with one DB instance and one standby)	Multi-AZ DB cluster normalized units per hour (deployment with one DB instance and two standbys)
micro	0.5	1	1.5
small	1	2	3
medium	2	4	6
large	4	8	12
xlarge	8	16	24
2xlarge	16	32	48
4xlarge	32	64	96
6xlarge	48	96	144
8xlarge	64	128	192

(Source: Size-flexible reserved DB instances)

Reserved DB Instances Offering types:

There are 3 types of purchase options for reserved DB instances. They are as follows:

1. **No Upfront**

- Provides access to a reserved DB instance without requiring an upfront payment.
- Billed with a discounted hourly rate for every hour within the term.
- Only available as a one-year reservation.

Partial Upfront

- This method requires part of the reserved DB instance to be paid upfront.
- The remaining hours in the term are billed at a discounted hourly rate, regardless of usage.
- Acts as a replacement for the previous Heavy Utilization option.

All Upfront

- Full payment is made at the start of the term.
- No extra cost for the remainder of the term, except for the number of hours used.

References:

- Hosted PostgreSQL - Amazon RDS for PostgreSQL - AWS
- Amazon RDS Reserved Instances
- Reserved DB instances for Amazon RDS
- Part II: RDS - The Ultimate Guide to Saving Money with AWS Reserved "Anything"
- How to Reduce Amazon RDS Costs with Reserved Instances | Parquantix
- Amazon RDS Reserved Instances: Everything You Need to Know!
- Are RDS reserved instances upgradable to a higher version of the same family?