***Databases***

***There are two types of Databases:***

1. ***Relational Databases***

*Databases structured to recognize relations between stored items of information*

***Examples –***

* 1. *Microsoft SQL Server*
  2. *Oracle DB*
  3. *MySQL*
  4. *IBM DB2*
  5. *PostgreSQL*
  6. *SQLite*
  7. *Ingres*
  8. *MariaDB*

1. ***Non-Relational Database:*** *A non-relational database is a database that does not incorporate the table/key model that relational database management systems promote.*

***Examples –***

* 1. *DynamoDB*
  2. *MongoDB*
  3. *Cassandra*
  4. *Coachbase*
  5. *HBase*
  6. *Redis*
  7. *Neo4j*

***Final Verdict***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **AWS RDS** | **AWS DynamoDB** | **Databases on EC2** |
| **Difficulty Level for Installation** | Difficult | Easy | Moderate |
| **Difficulty Level for Implementation** | Difficult | Easy | Moderate |
| **Difficulty Level for Scaling** | Unknown | Easy | Easy |
| **Comparison**  **(Migrating to other Clouds)** | Migrating Databases to Other Clouds possible. | Migrating Databases to Other Clouds possible. | Migrating Data from Database to Other Clouds is tiresome. |
| **Comparison**  **(Implementation Time)** | More Time is needed to host, scale, maintain and implement these Databases. | Minimal Time is needed to host and scale Database as it can be set to automatic scaling. | Moderate Time is needed for initial hosting and scaling Databases as it can be set to automatic scaling later on. |
| **Comparison**  **(Coding Time)** | Lot of Coding and Research needs to be done before Installation for setting up Database and then running. | Minimal Coding for Database set up, Implementation and Running. | Some Coding needs to be done along with Dependent Software Installations for setting up Database and then running. |
| **Pricing**  **Considering all terms** | Costlier than both. | Cheaper than RDS | Cheaper than RDS. Can prove cheaper than DynamoDB. |
| **Viable Option as per our Requirements?** | No | Yes | Yes |

***AWS Storage Options***

***AWS Storage mainly comprises of the following options:***

|  |  |  |
| --- | --- | --- |
| ***AWS RDS*** | ***AWS DynamoDB*** | ***Databases on AWS EC2*** |
| ***Database Options:***   1. ***Amazon Aurora*** 2. ***MySQL*** 3. ***Oracle*** 4. ***MariaDB*** 5. ***PostgreSQL*** 6. ***Microsoft SQL*** | ***Database Options:***   1. ***DynamoDB*** | ***Database Options:***   1. ***EBS Volumes*** 2. ***Self-Hosted DB*** |

***Amazon RDS (Relational Database System)***

Features of Amazon RDS

1. RDS is a web service that provides the capabilities of:
   1. MySQL
   2. Oracle
   3. MariaDB
   4. PostgreSQL
   5. Microsoft SQL Server

Relational databases as a managed, cloud-based service

1. RDS eliminates much of the administrative overhead associated with
   1. Launching
   2. Managing
   3. Scaling

Your own relational database on Amazon EC2 or in another computing environment.

1. RDS is a great solution for Cloud-based fully managed relational database system.
2. RDS is optimal for new applications with **structured data that requires more sophisticated querying and joining capabilities** than that provided by Amazon’s NoSQL databases.
3. RDS provides full compatibility with the databases supported and direct access to native database engines, code and libraries and is ideal for existing applications that rely on these databases.

IOPS Performance for Amazon RDS

RDS Provisioned IOPS, where the IOPS can be specified when the instance is launched and is guaranteed over the life of the instance

Durability and Availability

1. RDS leverages Amazon EBS volumes as its data store.
2. If enabled, RDS will automatically perform a full daily backup of your data during the specified backup window, and will also capture DB transaction logs.
3. User can initiate backups at time and they are not deleted unless deleted explicitly by the user.
4. RDS Multi AZ’s feature enhances both the durability and the availability of the database by synchronously replicating the data between a primary RDS DB instance and a standby instance in another Availability Zone, which prevents data loss.
5. RDS provides a DNS endpoint and in case of a failure on the primary, it automatically fails over to the standby instance.
6. RDS also allows Read replicas for the supported databases, which are replicated asynchronously.

Scalability and Elasticity

1. RDS resources can be scaled elastically in several dimensions:
   1. Database storage size
   2. Database storage IOPS rate
   3. Database instance compute capacity
   4. Number of read replicas
2. RDS supports “pushbutton scaling” of both database storage and compute resources. Additional storage can either be added immediately or during the next maintenance cycle
3. RDS for MySQL also enables you to scale out beyond the capacity of a single database deployment for read-heavy database workloads by creating one or more read replicas.
4. Multiple RDS instances can also be configured to leverage database partitioning or sharding to spread the workload over multiple DB instances, achieving even greater database scalability and elasticity.

Interfaces

1. RDS APIs and the AWS Management Console provide a management interface that allows you
   1. To create, delete, modify, and terminate RDS DB instances
   2. To create DB snapshots
   3. To perform point-in-time restores
2. There is no AWS data API for Amazon RDS.
3. Once a database is created, RDS provides a DNS endpoint for the database which can be used to connect to the database.
4. Endpoint does not change over the lifetime of the instance even during the failover in case of Multi-AZ configuration.

Pricing for Amazon RDS

1. RDS offers a tiered pricing structure, based on:
   * The size of the database instance
   * The deployment type (Single-AZ/Multi-AZ)
   * The AWS region.
2. Pricing for RDS is based on several factors:
   * The DB instance hours (per hour)
   * The amount of provisioned database storage (per GB-month and per million I/O requests)
   * Additional backup storage (per GB-month)
   * Data transfer in/out (per GB per month)

Amazon Aurora Pricing

|  |  |  |
| --- | --- | --- |
|  | MySQL Compatible Edition | PostgreSQL Compatible Edition |
| Region | Asia Pacific (Mumbai) | Region: Asia Pacific (Mumbai) |
| Standard Instance | db.t2.medium - $0.118 per hour | - |
| (Standard Instance)  Memory Optimized | db.r3.large - $0.33 per hour | db.r4.large - $0.33 per hour |
| Reserved Instance | Db.t2.medium –   * 1-Year Term: $75.92 per month (No upfront) * 3-Year Term: $13.14 per month (Partial Upfront) | Db.r4.large –   * 1-Year Term: $158.41 per month (No upfront) * 3-Year Term: $51.10 per month (Partial Upfront) |
| Database Storage | $0.11 per GB-month | $0.11 per GB-month |
| I/O Rate | $0.22 per 1 million requests | $0.22 per 1 million requests |
| Backup Storage | No additional charges | No additional charges |
| Data Transfer Rate | * Incoming from Internet: $0.00 per GB * Outgoing to Internet:   + Up to 1 GB/ Month - $0.00 per GB   + Next 9.999 TB/Month - $0.1093 per GB * Outgoing to Singapore Region: $0.086 per GB | * Incoming from Internet: $0.00 per GB * Outgoing to Internet:   + Up to 1 GB/Month - $0.00 per GB   + Next 9.999 TB/Month - $0.1093 per GB * Outgoing to Singapore Region: $0.086 per GB |
|  | Data transferred between Amazon RDS and Amazon EC2 Instances in the same Availability Zone is free. | Data transferred between Amazon RDS and Amazon EC2 Instances in the same Availability Zone is free. |

PostgreSQL Pricing:

|  |  |  |
| --- | --- | --- |
|  | PostgreSQL | PostgreSQL |
| Region | Singapore | Mumbai |
| Standard Instance | db.t2.medium - $0.112 per hour | db.t2.medium - $0.106 per hour |
| (Standard Instance)  Memory Optimized | db.r3.large - $0.30 per hour | db.r3.large - $0.285 per hour |
| Reserved Instance | Db.t2.medium –   * 1-Year Term: $65.408 per month (No upfront) * 3-Year Term: $28.47 per month (Partial Upfront) | Db.t2.medium –   * 1-Year Term: $68.62 per month (No upfront) * 3-Year Term: $29.20 per month (Partial Upfront) |
| Database Storage | SSD - $0.138 per GB-month | SSD - $0.131 per GB-month |
| I/O Rate | $0.11 per IOPS-month | $0.105 per IOPS-month |
| Backup Storage | No additional charges | No additional charges |
| Data Transfer Rate | * Incoming from Internet: $0.00 per GB * Outgoing to Internet:   + Up to 1 GB/ Month - $0.00 per GB   + Next 9.999 TB/Month - $0.12 per GB * Outgoing to Mumbai Region: $0.09 per GB | * Incoming from Internet: $0.00 per GB * Outgoing to Internet:   + Up to 1 GB/ Month - $0.00 per GB   + Next 9.999 TB/Month - $0.1093 per GB * Outgoing to Singapore Region: $0.086 per GB |
|  | Data transferred between Amazon RDS and Amazon EC2 Instances in the same Availability Zone is free. | Data transferred between Amazon RDS and Amazon EC2 Instances in the same Availability Zone is free. |

Case-dependent alternatives for Amazon RDS

1. If our applications don’t require advanced features such as joins and complex transactions and is more oriented toward indexing and querying data, **DynamoDB** would be more appropriate for this needs.
2. RDS provides pushbutton scaling and it only scales up and has limited scale out ability. If fully-automated scaling is needed, **DynamoDB** may be a better choice.
3. If the application **makes heavy use of files (audio files, videos, images, etc.), it is a better choice to use S3 to store the objects.**
4. RDS does not provide admin access and does not enable the full feature set of the database engines. So if the application requires complete, OS-level control of the database server with full root or admin login privileges, a self-managed database on **EC2** may be a better match.

***Amazon DynamoDB***

Features of Amazon DynamoDB

1. Amazon DynamoDB makes it simple and cost-effective to store and retrieve any amount of data, and serve any level of request traffic.
2. DynamoDB being a managed service helps offload the administrative burden of operating and scaling a highly-available distributed database cluster.
3. DynamoDB helps meet the latency and throughput requirements of highly demanding applications by providing extremely fast and predictable performance with seamless throughput and storage scalability.
4. DynamoDB provides:
   * Eventually-consistent reads (by default)
   * Strongly-consistent reads (optional)
   * Implicit item-level transactions for item put, update, delete, conditional operations, and increment/decrement.
5. DynamoDB supports three data types: number, string, and binary, in both scalar and multi-valued sets.
6. Primary key can either be a single-attribute hash key or a composite hash-range key.
7. Local secondary indexes provide additional flexibility for querying against attributes other than the primary key.
8. DynamoDB is ideal for existing or new applications that need a flexible NoSQL database with low read and write latencies, and the ability to scale storage and throughput up or down as needed without code changes or downtime.
9. Use cases require a highly available and scalable database because downtime or performance degradation has an immediate negative impact on an organization’s business

IOPS Performance for Amazon DynamoDB

1. SSDs and limited indexing on attributes provides high throughput and low latency and drastically reduces the cost of read and write operations.
2. Provisioned throughput capacity reservations are elastic and can be increased or decreased on demand.

Durability and Availability

DynamoDB has built-in fault tolerance that automatically and synchronously replicates data across three AZ’s in a region for high availability and to help protect data against individual machine, or even facility failures.

Scalability and Elasticity

1. DynamoDB is both highly-scalable and elastic.
2. DynamoDB provides unlimited storage capacity, and the service automatically allocates more storage as the demand increases
3. Data is automatically partitioned and re-partitioned as needed, while the use of SSDs provides predictable low-latency response times at any scale.
4. DynamoDB is also elastic, in that you can simply “dial-up” or “dial-down” the read and write capacity of a table as your needs change.

Interfaces

1. DynamoDB provides a low-level REST API, as well as higher-level SDKs in different languages
2. APIs provide both a management and data interface for Amazon DynamoDB, that enable table management (creating, listing, deleting, and obtaining metadata) and working with attributes (getting, writing, and deleting attributes; query using an index, and full scan).

Pricing for Amazon DynamoDB

* DynamoDB simply asks you to specify the target utilization rate and minimum to maximum capacity that you want for your table.
* DynamoDB handles the provisioning of resources to achieve your target utilization of read and write capacity, then auto scales your capacity based on usage.
* Optionally, you can directly specify read and write capacity if you prefer to manually manage table throughput.

|  |  |  |
| --- | --- | --- |
| Resource Type | Details | Monthly Price |
| Write Throughput | 1 Write Capacity Unit = 1 Write per second | $0.47 per WCU per month |
| Read Throughput | 1 Read Capacity Unit = 2 Reads per second | $0.09 per RCU per month |
| Indexed Data Storage | Hourly Rate per GB of Disk space | $0.25 per 1 GB of Data |

Case-dependent alternatives for Amazon DynamoDB

1. If the application uses structured data and required joins, complex transactions or other relationship infrastructure provided by traditional database platforms, it is better to use **RDS** or Database installed on an **EC2 instance**
2. If the application **uses large blob data**for e.g. media, files, videos etc**., it is better to use S3 to store the objects and use DynamoDB to store metadata.**
3. DynamoDB uses SSD drives and is optimized for workloads with a high I/O rate per GB stored. If the applications stores very large amounts of data that are infrequently accessed, **S3** might be a better choice.
4. For porting an existing application using databases, **RDS** or **database installed on the EC2 instance** would be a better and seamless solution.

***Databases on AWS EC2***

Features

1. EC2 with EBS volumes allows hosting a self-managed relational database.
2. Ready to use, prebuilt AMIs are also available from leading database solutions.
3. Databases on EC2 is an ideal scenario for users whose application requires a specific traditional relational database not supported by Amazon RDS*for e.g. IBM DB2, Informix, or Sybase*
4. Users or applications that require a maximum level of administrative control and configurability which is not provided by RDS, is achieved under this tier.

IOPS Performance for Amazon EC2 based Databases

1. Performance depends on the size of the underlying EC2 instance, the number and configuration of the EBS volumes and the database itself
2. Performance can be increased by scaling up memory and compute resources by choosing a larger Amazon EC2 instance size.
3. For database storage, it is usually best to use EBS Provisioned IOPS volumes. To scale up I/O performance, the Provisioned IOPS can be increased, the number of EBS volumes changed, or use software RAID 0 (disk striping) across multiple EBS volumes, which will aggregate total IOPS and bandwidth.

### Durability & Availability

As the database on EC2 uses EBS as storage, it has the same durability and availability provided by EBS and can be further enhanced by using EBS snapshots or by using third-party database backup utilities (such as Oracle’s RMAN) to store database backups in Amazon S3

### Scalability & Elasticity

Users of traditional relational database solutions on Amazon EC2 can take advantage of the scalability and elasticity of the underlying AWS platform by creating AMI and spawning multiple instances

Pricing for Amazon EC2 based Databases

Cost for running a database on EC2 instance is mainly determined by:

1. The size and the number of EC2 instances running
2. The size of the EBS volume used for database storage
3. Any third party licensing cost for the database

|  |  |  |
| --- | --- | --- |
|  | On-Demand Pricing | Spot Instances Pricing |
| Description | On-Demand instances let you pay for compute capacity by the hour or second (minimum of 60 seconds) with no long-term commitments. | Spot prices are set by Amazon EC2 and fluctuate periodically depending on the supply of and demand for Spot instance capacity. |
| Region | Mumbai | Mumbai |
| Instance Type | t2.medium (Linux)- $0.0576 per Hour  CPUs – 2  Memory – 4GB  Storage Type – EBS only | t2.medium (Linux) - $0.0281 per Hour |
| Data Transfer Rate | * Incoming from Internet using private IPv4 address: $0.00 per GB * Incoming from Internet using Elastic IPv4 address: $0.01 per GB * Outgoing to Internet using private IPv4 address: $0.00 per GB * Outgoing to Internet using Elastic IPv4 address - $0.01 per GB * Data Transfer Out to Internet:   + First 1GB/Month - $0.00 per GB   + Up to 10TB/Month - $0.1093 per GB | * Incoming from Internet using private IPv4 address: $0.00 per GB * Incoming from Internet using Elastic IPv4 address: $0.01 per GB * Outgoing to Internet using private IPv4 address: $0.00 per GB * Outgoing to Internet using Elastic IPv4 address - $0.01 per GB * Data Transfer Out to Internet:   + First 1GB/Month - $0.00 per GB   + Up to 10TB/Month - $0.1093 per GB |
| Elastic IP Addresses | * $0.00 for one Elastic IP address associated with a running instance * $0.005 per additional Elastic IP address associated with a running instance per hour on a pro rata basis * $0.005 per Elastic IP address not associated with a running instance per hour on a pro rata basis | * $0.00 for one Elastic IP address associated with a running instance * $0.005 per additional Elastic IP address associated with a running instance per hour on a pro rata basis * $0.005 per Elastic IP address not associated with a running instance per hour on a pro rata basis |
| Auto Scaling | No Additional Fees | No Additional Fees |

**EBS Pricing** must also be taken into consideration for EC2 Storage:

* **General Purpose SSD (gp2)** - $0.114 per GB-month
* **Provisioned IOPS SSD (io1)** –
  + $0.131 per GB-month
  + $0.068 per IOPS-month
* **Optimized HDD (st1)** - $0.051 per GB-month

Case-dependent alternatives for Amazon EC2 based Databases

1. If the applications don’t require advanced features such as joins and complex transactions and is more oriented toward indexing and querying data, **DynamoDB** would be more appropriate for this needs.
2. If the application makes heavy use of files (audio files, videos, images, and so on), it is a better choice to use **S3** to store the objects instead of database engines Blob feature and use **RDS** or **DynamoDB** only to save the metadata.
3. Relational databases on EC2 leverages the scalability and elasticity of the underlying AWS platform, but this requires system administrators or DBAs to perform a manual or scripted task. If you need pushbutton scaling or fully-automated scaling, **DynamoDB** or **RDS** may be a better choice.
4. If the application using **RDS** supported database engine and all the features are available, **RDS** would be a better choice instead of self-managed relational database on EC2