[Storage 2](#_Toc1558631)

[Database 2](#_Toc1558632)

[Identity & Compliance 2](#_Toc1558633)

[Elastic Block Storage 2](#_Toc1558634)

[Metrics 2](#_Toc1558635)

[Use Cases for Use of EBS 3](#_Toc1558636)

[Types & Use Cases 3](#_Toc1558637)

[Root Volumes 3](#_Toc1558638)

[Secondary Volumes 3](#_Toc1558639)

[EBS life-time vs Instance Life-time 3](#_Toc1558640)

[EBS Snapshots 3](#_Toc1558641)

[EBS Cost 3](#_Toc1558642)

[EBS Performance 3](#_Toc1558643)

[EBS Resilience 4](#_Toc1558644)

[EBS Security 4](#_Toc1558645)

[Encrypting EBS Drive data 4](#_Toc1558646)

[Encrypting EBS Snapshots 4](#_Toc1558647)

[EBS Operational Excellence 4](#_Toc1558648)

[EBS-Optimized EC2 Instances 4](#_Toc1558649)

[RDS 5](#_Toc1558650)

[Scaling Amazon RDS Instances Vertically and Horizontally 5](#_Toc1558651)

[Vertical Scaling RDS Instance 5](#_Toc1558652)

[How to detect Load on the primary database? 5](#_Toc1558653)

[Read Replicas 5](#_Toc1558654)

[Use Cases for Read Replica’s 5](#_Toc1558655)

[Constraints for Read Replicas 6](#_Toc1558656)

[How Many Read Replicas? Do Users connect to them individually? 6](#_Toc1558657)

[Read Replicas in Multiple AZ? 6](#_Toc1558658)

[Read Replicas in different Regions? 6](#_Toc1558659)

[Time lag between transaction and Read Replica Update 6](#_Toc1558660)

[Which Databases support RDS Read-replicas? 6](#_Toc1558661)

[Load Balancing Read Replica’s 6](#_Toc1558662)

# Storage

S3 – Simple Storage Service (Object Storage)

S3 Glacier – (Archive Storage)

EBS – Elastic Block Storage (Block Storage)

EFS – Elastic File System (File Storage)

SnowFamily – Data Migration Devices (Snowball, Snowball Edge, SnowMobile)

Storage Gateway – Hybrid Storage Integration (Cache & Backup)

# Database

RDS – Relational Database Service (Relational)

Aurora – MySQL compatible (Relational)

ElastiCache – In-Memory Cache (Redis, Memcached)

DynamoDB – NoSQL Database Service (NoSQL – Key-Value Database)

DocumentDB – MongoDB Compatible (NoSQL - Document Database)

Neptune – (NoSQL Graph Database)

TimeStream – (NoSQL TimeSeries Database)

Quantum – (NoSQL Ledger Database)

RedShift – Petabyte-scale Data-Warehouse

RDS on VMWARE (On-Premisis RDS)

Database Migration Service

# Identity & Compliance

IAM

# S3 – Simple Storage Service

## S3 Overview

Object Store – stores files

Unlimited Storage

Highly Available

Highly Durable

Buckets and Files

### Buckets

Bucket Names are Global and Not Regional

Once a Bucket is created, nobody else can use the same name in any region

100 Buckets/Account to start with. Can be increased if required.

Unlimited number of files in a bucket

### Keys and objects

Key is the name of the object – all that after the bucket name

## Data Consistency Model

* Read After Write consistency for PUTS of New Objects
* Eventual Consistency for overwrite PUTS and DELETES

### Implications of Consistency Model

* Reading a file immediately after creating it on S3 will return the file
* Reading a file immediately after updating it might return old or new file
* Reading a file immediately after delete might return the file instead of error

## S3 Settings

### Bucket Settings

|  |  |
| --- | --- |
| Setting | Description |
| Bucket Name | Must be Unique across all existing bucket names in S3.  If used as origin for CloudFront Distribution, they have restrictions |
| Region | Buckets belong to specific regions  Best Practice is to create S3 Bucket in the same region as the Compute |
| Versioning | Keep Versions of Buckets |
| Server Access Logging | Enable/Disable |
| Tags |  |
| Object Level Logging | Record Object-level API activitity using AWS CloudTrail for additional cost |
| Default Encryption | Automatically Encrypt objects when stored in S3 |
| CloudWatch Request Metrics | Monitor requests in your bucket for additional cost |
| Manage Public ACLs | Block new Public ACLs and uploading public objects  Remove Public Access granted through public ACLs |
| Manage Public Bucket Policies | Block new public bucket Policies  Block public and cross-account access if bucket has public policies |
| System Permissions | Allow/Deny Amazon S3 Log Delivery Group Write access to this bucket.  This must be enabled for a bucket that we will use as Target-bucket for Server Access Logging. |
|  |  |

## S3 Object Settings

Users – User Id/E-Mail, Permissions to object, Permissions to Modify Object Permissions

Public Permission – Enable/Disable

Storage Class – Standard/Intelligent-Tiering/Standard-IA/One Zone-IA/Glacier/RRS

Encryption – None/Amazon S3 Master-Key/AWS KMS Master Key

Metadata -

Tag – Tags to Search, Organize and manage access

## S3 Resilience

## S3 Performance

S3 Read/Write Performance

Scaling S3 with Prefix

CloudFront

Transfer Acceleration

## S3 Security

3 Ways to set permissions in S3

1. Using IAM  
   Custom/Inline Policy can be set to IAM User/Group/Role for S3 Access. Can manage very large number of buckets

{

"Sid": "AnveshiTechS3ListBuckets",

"Effect": "Allow",

"Action": [

"s3:List\*"

],

"Resource": [

"arn:aws:s3:::anveshi-tech",

"arn:aws:s3:::anveshi-tech/share/developers"

]

},

{

"Sid": "AnveshiTechS3ShareForDevelopers02",

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:PutObject",

"s3:DeleteObject"

],

"Resource": [

"arn:aws:s3:::anveshi-tech/share/developers/\*"

]

}

1. Using ACL (Access Control List)

* ACL are the first and old method of securing S3.
* Grant Basic Read/Write permissions to other Accounts; not to individual users of the account
* No option to provide conditional permissions or explicity-deny permissions
* Use Bucket ACL to grant permission to S3 Log Delivery Group to write access Logs to bucket
* Can specify permissions for Buckets and Individual Files

1. Using Bucket Policies

Bucket Policy is very similar to IAM User Policy

Can provide explicit allow and deny policies to specified ARNs for designationed Actions.

## S3 Cost Effectiveness

S3 Lifecycle Management

## S3 Operation Excellence

S3 Default Bucket Properties

S3 Object Properties

S3 Versioning

S3 Versioning with Pricing

S3 Signed URLs

CloudFront Signed URLs

CloudFront Origin Access Identity

CloudFront Signed Cookies

AWS Storage Gateway

100 S3 Buckets by Default per Account

Multi-Part Uploads

S3 Scaling

Cross-Region Replication

Cloud-Front Integration with S3

S3 Transfer Acceleration

S3 Security and Encryption

S3 and Storage Gateway Integration

## S3 Server Access Logging

* Provides detailed records for requests made to a bucket
* Useful for Security, Audit, to understand customer base/Access pattern, Understanding Amazon Bill (For proper usage of Infrequent Access/Glacier)
* Log Record – Requester, Bucket Name, Request Name, Request Action, Response Status, Error Code
* Billing
  + No Extra Charges
  + S3 charges for storing and accessing the log files created
* Disabled by Default
* Target Bucket
  + Use a different/separate bucket for this purpose
  + Use a Prefix for Log Object Keys to identify them easily
  + Logs are saved to bucket in same Region
* Log Format
  + <Target Prefix>YYYY-mm-DD-HH-MM-SS-UniqueString
* Log Files
  + Log records are periodically collected, consolidated and saved in files.

## S3 URLs

## S3 Performance

## S3 Storage Classes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | S3-Standard | S3 Standard-IA | S3-OneZone IA | S3 - RRS | Glacier |
|  |  |  |  |  |  |
| Durability | 11 9’s | 11 9’s | 11 9’s | 11 9’s | 11 9’s |
| Availability | 99.99% | 99.9% | 99.5% | 99.99% | N/A |
| Cost | $ 0.023/GB | $0.0125/GB | $ 0.010/GB | $0.024/GB | $0.004/GB |
| Availability Zones | >=3 | >=3 | 1 |  | >=3 |
| Minimum Storage Duration Charge | NA | 30 days | 30 days |  | 90 days |
| First Byte Latency | Milliseconds | Milliseconds | Milliseconds |  | Minutes or Hours |

|  |  |  |  |
| --- | --- | --- | --- |
|  | S3 | EBS | EFS |
| Storage Type | Object Store | Block Store | File Store |
| Minimum File Size | 0 |  |  |
| Maximum File Size | 5 TB |  |  |
| PUT Opload Max | 5 GB |  |  |
| Multipart Upload |  |  |  |
|  |  |  |  |

S3 – Storage infra that is Extremely Durable, Highly Available, Infinitely Scalable

11 9’s (99.999999999 %) Durability i.e. 1 file lost every 8 Years

Calculation:

Average size of file = 800 MB

Assumed Data = 1PB

Number of Objects = 1PB/800MB = 1.2 Billion

# CloudFront

# Elastic Block Storage

* Block-level storage volumes for use with EC2 instances
* Highly available, highly reliable
* Persist independently from life of instance

Provides Block Storage – What is Block Storage?

Attached to EC2 Instances

Can be attached to only 1 EC2 Instance

## Metrics

Minimum Size:

Maximum Size:

Number of EBS Volumes that can be attached

IOPS:

## Use Cases for Use of EBS

* Need quick/fast access to data
* Need long-term persistence
* Primary storage for File-systems/Databases
* Apps that need granular, raw, un-formatted, block-level storage

## Types & Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Code | Base Performance |  |
| General Purpose SSD | gp2 | 3 IOPS/GiB |  |
| Provisioned IOPS SSD | io1 |  |  |
| Throughput Optimized HDD | st1 |  |  |
| Cold HDD | sc1 |  |  |

Instance Store Vs EBS

Cold HDD

Provisioned IOPS

## Root Volumes

The following EBS Types can be Root Volumes – General Purpose SSD (gp2), Provisioned IOPS SSD (io1) and Magnetic (standard).

## Secondary Volumes

All 5 types can function as Secondary volumes.

## EBS life-time vs Instance Life-time

## EBS Snapshots

What is it for?

How much time does it take?

Where is it stored?

How to transfer EBS Snapshots between Regions?

## EBS Cost

## EBS Performance

Cloudwatch metrics for EBS Performance

## EBS Resilience

## EBS Security

### Encrypting EBS Drive data

Encryption at Rest or Encryption in Transfer?

### Encrypting EBS Snapshots

## EBS Operational Excellence

Logging

Management

Backing-Up EBS

Copying Volumes Vs taking Snapshots

## EBS-Optimized EC2 Instances

How to increase size of EBS Volumes attached to EC2 Instances?

# RDS

Which DBs are supported by RDS?

MySQL, PostgreSql, MSSQL Server, Maria DB,

Load Balancing RDS Instances

## Scaling Amazon RDS Instances Vertically and Horizontally

<https://aws.amazon.com/blogs/database/scaling-your-amazon-rds-instance-vertically-and-horizontally/>

Scale Vertically when application uses roughly equal number of reads and writes. For Read-Heavy applications, Scale Horizontally using Read-replica’s.

### Vertical Scaling RDS Instance

* DB can be vertically scaled any time
* Check licensing for Oracle/SQL Server as they are bound by CPU Cores
* On Scaling up/down, only VCPUs and Memory with change based on Instance Type. Storage is de-coupled from this and does not change. Performance can be improved by changing from GP SSD to Provisioned IOPS SSD
* Apply immediately or wait for maintenance window
* Minimal downtime. if in Multi-AZ environment. Standby DB gets updated then failover to updated DB happens. Single Instance DB will be un-available during Scale operation

### Horizontal Scaling

Making DB Multi-AZ **DOES NOT HELP** in Scaling. The Standby DB does not take Read requests. It is only kept Synchronously updated as part of Write operation.

To Improve performance of Read-Heavy Database, Read Replicas can be created. For Write-heavy applications, only way to scale is Vertical Scaling or if possible, move to NoSQL Databases.

Aurora can have 15 Read Replicas. MySQL, Postgres & MariaDB can have up-to 5 read replicas.

## Multi-AZ Deployments

Provides High-Availability and Failover support for DB Instances

If enabled, RDS automatically provisions and maintains a **SYNCHRONOUS, STANDBY** Replica of MasterDB in a different availability Zone.

### How many Multi-AZ replicas?

### Behavior on Failover

### How to detect Load on the primary database?

## Read Replicas

Elastically scale-out beyond capacity of single DB Instance

Read-heavy Database Workloads

Read-replication use Engine’s Built-in replication, they are subject to its strengths and weaknesses

### Use Cases for Read Replica’s

1. Shift Read operations beyond Source DB for Read-heavy workloads
2. Serving Read Traffic when Source DB is unavailable
3. Business Reporting and Data Warehousing
4. Disaster recovery in same region or different region

### Constraints for Read Replicas

* Automatic Backups must be enabled on Source DB Instance
* Mysql 5.6, PostgreSQL 9.3.5, All Instances of MariaDB and Aurora support Read Replicas
* Source DB will suffer I/O suspension when read replica is being created.
* Read Replicas should have as much or More compute and storage resources than Source DB instances.
* Read Replica’s of Read-replicas supported only for Aurora, MySQL and MariaDB, not for Postgres.

### How Many Read Replicas? Do Users connect to them individually?

Aurora – 15. MySQL, Postgres, MariaDB – 5

Can connected individually or via Route53.

### Read Replicas in Multiple AZ?

Read Replica’s can be configured to support Multi-AZ configuration to support disaster recovery and minimize downtime from upgrades. However, as with Multi-AZ, the secondary instances are backup only.

When Multi-AZ is enabled on Source DB, in event of Multi-AZ failover, associated Read replicas will automatically resume replication once failover is completed.

### Read Replicas in different Regions?

Yes. Read Replica’s can be created in different Regions. This is only for MySQL, MariaDB, Postgres and Aurora.

### Time lag between transaction and Read Replica Update

Cloudwatch metric (Replica Lag) shows number of seconds that replica is behind the master.

### Which Databases support RDS Read-replicas?

MySQL, MariaDB, PostgreSQL, Aurora support RDS RRs. Oracle and MSSQL support Native RRs only.

Also, MSSQL does not support cross region RRs

### Load Balancing Read Replica’s

Elastic Load Balancer cannot be used to Load Balance between Read Replica’s.

You can use Amazon Route 53 weighted record sets to distribute requests across your read replicas. Within a Route 53 hosted zone, create individual record sets for each DNS endpoint associated with your read replicas and give them the same weight. Then, direct requests to the endpoint of the record set

<https://aws.amazon.com/premiumsupport/knowledge-center/requests-rds-read-replicas/>

There are other options mentioned in this article:

<https://aws.amazon.com/blogs/database/scaling-your-amazon-rds-instance-vertically-and-horizontally/>

Here, they use Layer 4 (HAProxy) OR Layer 7 load Balancers (MaxScale, MySQL Proxy, ProxySQL and other commercial solutions in AWS Marketplace)

# Todo: Amazon Aurora

# DynamoDB (Key-Value NoSQL Database)

# Neptune (Graph NoSQL Database)

# DocumentDB (Document NoSQL Database – MongoDB compatible)

# Heavy Database Read Requests

How to handle heavy Database Read Requests

What is Heavy Read Requests?

1. Place CloudFront distribution in front of Database
2. Place Elasticache to improve performance of DB
3. Enable Multi-AZ on DB
4. Place Web-server to take load

## Database Performance Metrics

# S3

Transfer Acceleration