AWS S3 & RDS

CS516 – Cloud Computing

Computer Science Department

Maharishi International University

Maharishi International University - Fairfield, Iowa

Content

- S3 concepts
  - Storage classes & Object lifecycles
  - Object versioning
  - Global replication
  - Static website hosting and CloudFront
- Amazon RDS
  - DB instance and engine
  - Parameter group
- Amazon Aurora
  - Features
  - Read replica
  - Aurora serverless
  - Cloning

Simple Storage Service - S3

Amazon S3 has a simple **interface** that you can use as a drive to store and retrieve any amount of data, at any time, from anywhere on the web. It can scale **infinitely**.

It gives any user access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites.

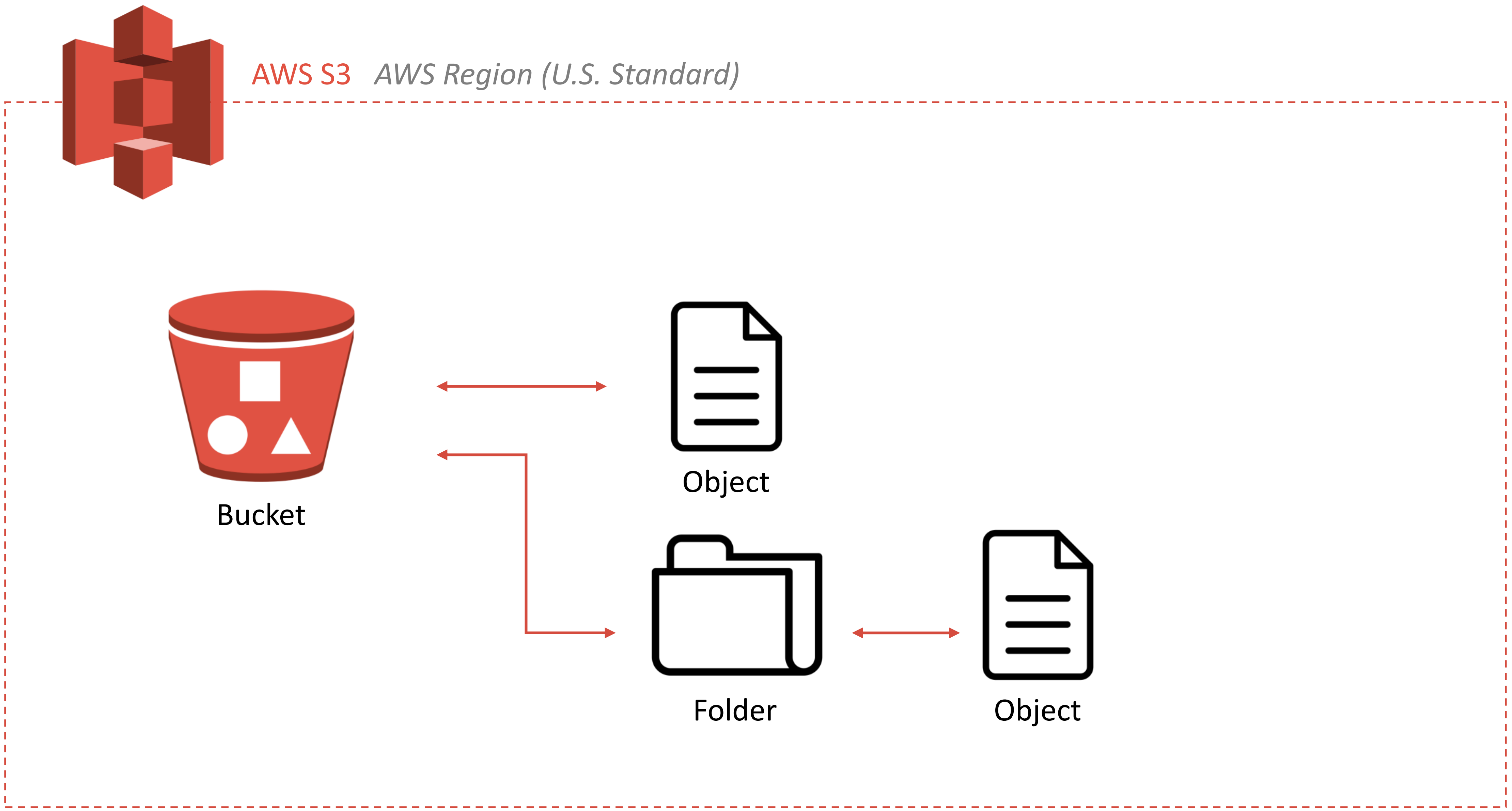
Learn more: Introduction to Amazon S3

AWS S3 Concepts

- **Buckets** is a container for objects stored in Amazon S3. Every object is contained in a bucket.
- **Objects** are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. Maximum object size of 5TB.
- **Key** is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, key, and version ID (**bucket + key + version**) uniquely identify each object.

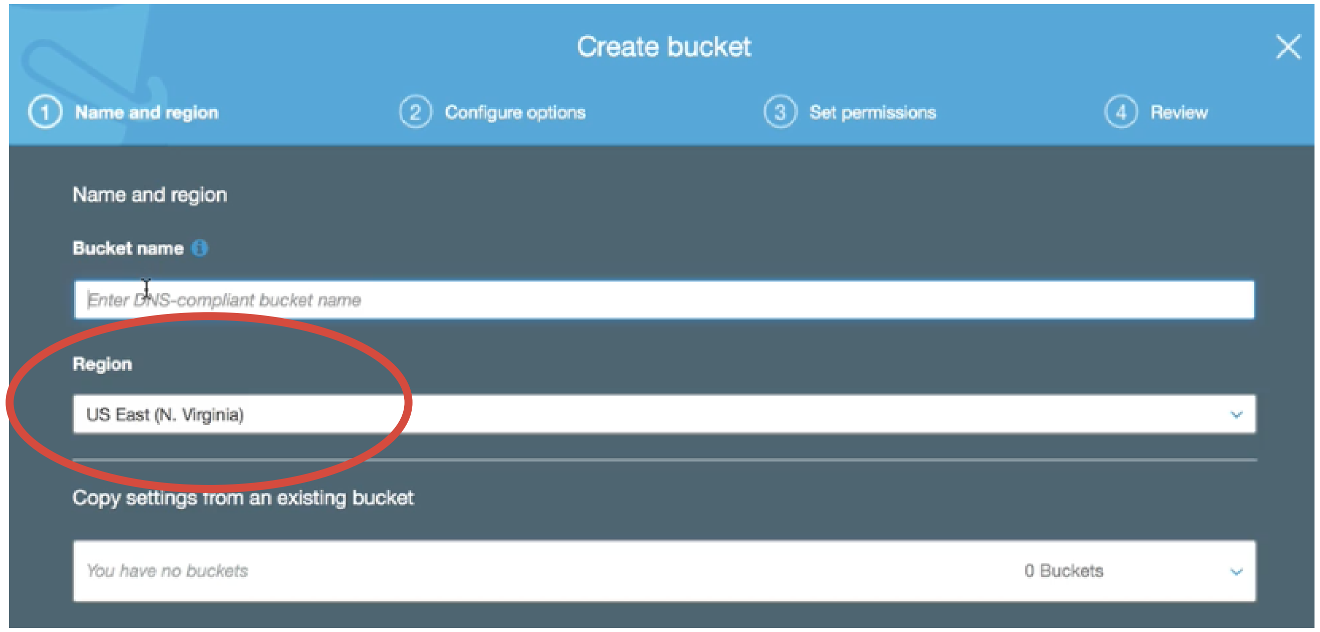
S3 bucket is a regional service but S3 namespace is global.

You can create a folder in a S3 bucket also know as prefix.


AWS S3
AWS Region (U.S. Standard)
Bucket
Object
Folder
Object

AWS S3 Regions

When you create a bucket, you must select a specific region for it. This means that any data you upload to the S3 bucket will be physically located in a data center in that region.



Name and region

Bucket name

Enter DNS-compliant bucket name

Region

US East (N. Virginia)

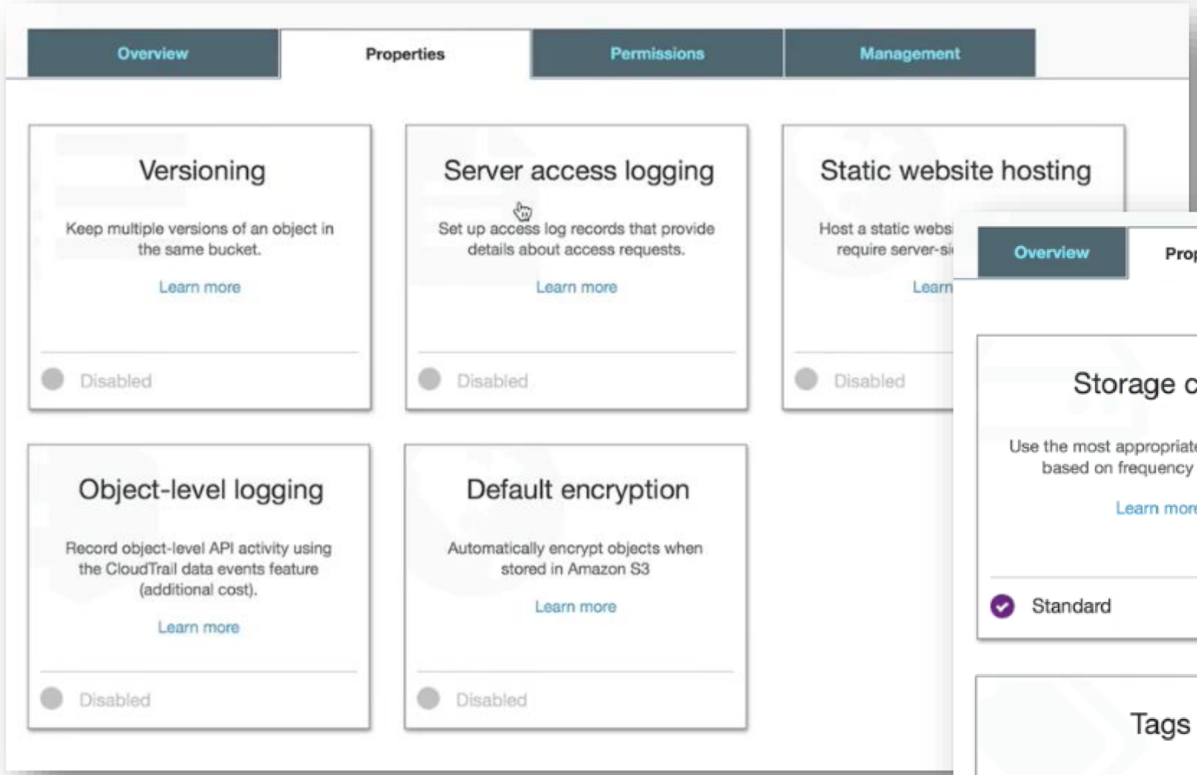
Copy settings from an existing bucket

You have no buckets 0 Buckets

Best practice is to select the region that is physically closest to you to reduce transfer latency.

Buckets must have a **globally unique name**.

Bucket and Object Properties



Versioning

Keep multiple versions of an object in the same bucket.

Learn more

Disabled

Server access logging

Set up access log records that provide details about access requests.

Learn more

Disabled

Static website hosting

Host a static webs
require server-si

Learn

Disabled

Object-level logging

Record object-level API activity using the CloudTrail data events feature (additional cost).

Learn more

Disabled

Default encryption

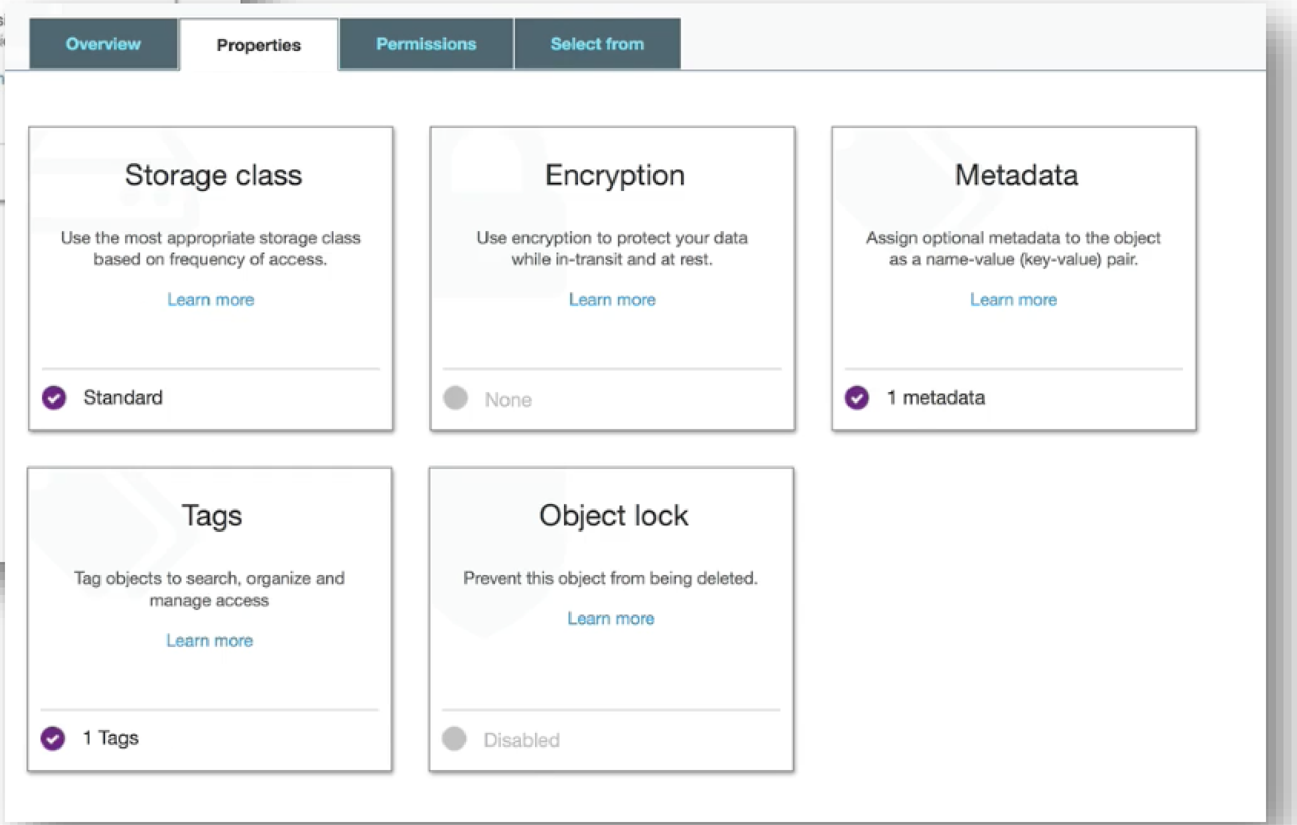
Automatically encrypt objects when stored in Amazon S3

Learn more

Disabled

Bucket Properties

Object Properties



Storage class

Use the most appropriate storage class based on frequency of access.

Learn more

Standard

Encryption

Use encryption to protect your data while in-transit and at rest.

Learn more

None

Metadata

Assign optional metadata to the object as a name-value (key-value) pair.

Learn more

1 metadata

Tags

Tag objects to search, organize and manage access

Learn more

1 Tags

Object lock

Prevent this object from being deleted.

Learn more

Disabled

S3 Storage Classes

A storage class represents the classification assigned to each object.

Each storage class has varying attributes that dictate:

- Storage cost
- Object availability
- Object durability
- Frequency of access

Each object must be assigned a storage class (standard is the default class)

Read more: Amazon S3 Storage Classes

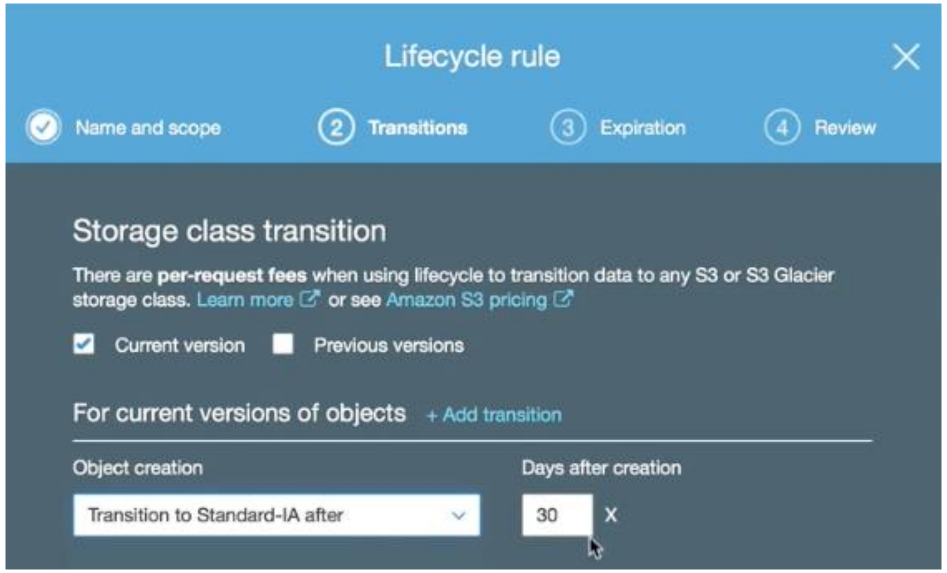
S3 Storage Classes

- **S3 Standard** - For mobile applications, content distribution.
- **S3 Intelligent-Tiering** - Automatically moves objects between two access tiers based on access patterns
- **S3 Standard - Infrequent Access (IA)** - For backups.
- **S3 One Zone IA** - Secondary backup, data you can recreate.
- **Glacier** - For long-term archival storage.
- **Glacier Deep Archive** - For long-term archival storage.

Object Durability is the percent (%) over a one-year time period that a file stored in S3 **will not be lost**.
Object Availbility is the percent (%) over a one-year time period that a file stored in S3 **will be accessible**.

S3 Object Lifecycle

An object lifecycle is a set of rules that automate the migration of an object's **storage class** to a different storage class, or its deletion, based on specified time intervals for cost optimization.


Lifecycle rule

Name and scope · Transitions · Expiration · Review

Storage class transition

There are **per-request fees** when using lifecycle to transition data to any S3 or S3 Glacier storage class. Learn more or see Amazon S3 pricing

[✓] Current version [ ] Previous versions

For current versions of objects + Add transition

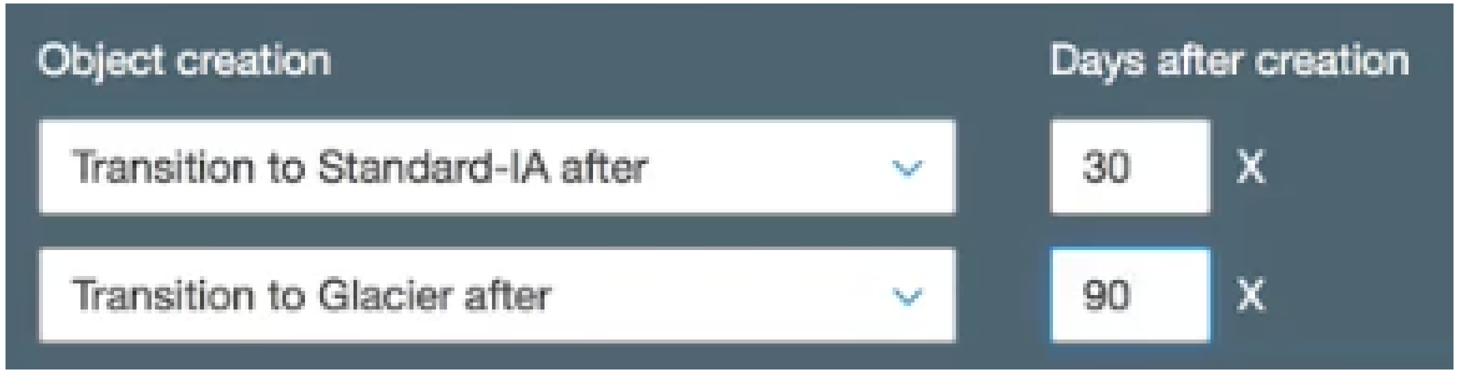
Object creation	Days after creation
Transition to Standard-IA after	30

Read more about [Object Lifecycle Management]()

Object Lifecycle Example

1. I have a work file that I am going to access every day for the next 30 days.
2. After 30 days, I may only need to access that file once a week for the 60 next days.
3. After that (90 days total) I will probably never access the file again but want to keep it just in case.

What is the best solution to meet the usage needs and minimize storage cost?



Object creation	Days after creation
Transition to Standard-IA after	30
Transition to Glacier after	90

Lifecycle Management

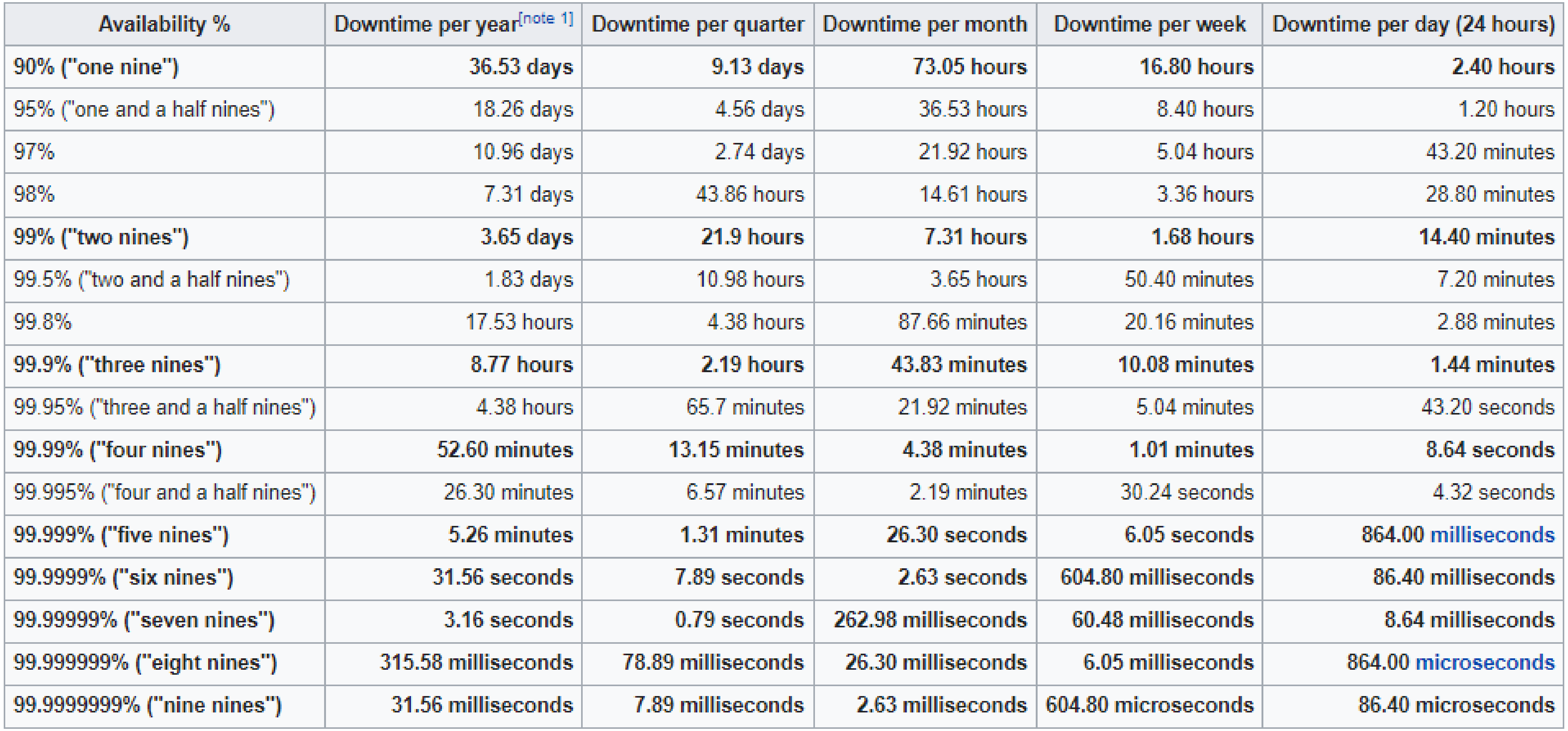
The Lifecycle functionality is located on the bucket level.

A lifecycle policy can be applied to:

- The entire bucket (applied all the objects in the bucket)
- One specific folder within a bucket (applied to all the objects in that folder)
- One specific object within a bucket

You can always delete a lifecycle policy or manually change the storage class back to whatever you like.

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes



Availability %	Downtime per year[note 1]	Downtime per quarter	Downtime per month	Downtime per week	Downtime per day (24 hours)
90% ("one nine")	**36.53 days**	**9.13 days**	**73.05 hours**	**16.80 hours**	**2.40 hours**
95% ("one and a half nines")	18.26 days	4.56 days	36.53 hours	8.40 hours	1.20 hours
97%	10.96 days	2.74 days	21.92 hours	5.04 hours	43.20 minutes
98%	7.31 days	43.86 hours	14.61 hours	3.36 hours	28.80 minutes
99% ("two nines")	**3.65 days**	**21.9 hours**	**7.31 hours**	**1.68 hours**	**14.40 minutes**
99.5% ("two and a half nines")	1.83 days	10.98 hours	3.65 hours	50.40 minutes	7.20 minutes
99.8%	17.53 hours	4.38 hours	87.66 minutes	20.16 minutes	2.88 minutes
99.9% ("three nines")	**8.77 hours**	**2.19 hours**	**43.83 minutes**	**10.08 minutes**	**1.44 minutes**
99.95% ("three and a half nines")	4.38 hours	65.7 minutes	21.92 minutes	5.04 minutes	43.20 seconds
99.99% ("four nines")	**52.60 minutes**	**13.15 minutes**	**4.38 minutes**	**1.01 minutes**	**8.64 seconds**
99.995% ("four and a half nines")	26.30 minutes	6.57 minutes	2.19 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	**5.26 minutes**	**1.31 minutes**	**26.30 seconds**	**6.05 seconds**	**864.00 milliseconds**
99.9999% ("six nines")	**31.56 seconds**	**7.89 seconds**	**2.63 seconds**	**604.80 milliseconds**	**86.40 milliseconds**
99.99999% ("seven nines")	**3.16 seconds**	**0.79 seconds**	**262.98 milliseconds**	**60.48 milliseconds**	**8.64 milliseconds**
99.999999% ("eight nines")	**315.58 milliseconds**	**78.89 milliseconds**	**26.30 milliseconds**	**6.05 milliseconds**	**864.00 microseconds**
99.9999999% ("nine nines")	**31.56 milliseconds**	**7.89 milliseconds**	**2.63 milliseconds**	**604.80 microseconds**	**86.40 microseconds**

11 Nines Data Durability

Conceptually, if you store 1 million objects in S3 for 10 million years, you would expect to lose 1 file.

There’s a higher likelihood of an asteroid destroying Earth within a million years.

S3 Permissions

S3 permissions allow you to have **granular control** over who can view, access, and use specific buckets and objects.

Permissions functionality can be found on the **bucket** and **object** level.

There are 3 types of permissions:

- **Identity-based** – IAM user or role has policies to access S3.
- **Resource-based (bucket policy)** – You can define policies on the S3 bucket itself with principal.
- **Access Control List (ACL)** – Legacy but still supported and used. We can set policies to its objects with ACL.

Note: You must provide public access to **both the bucket and object** in order to make it available to the world.

Read more: IAM in Amazon S3

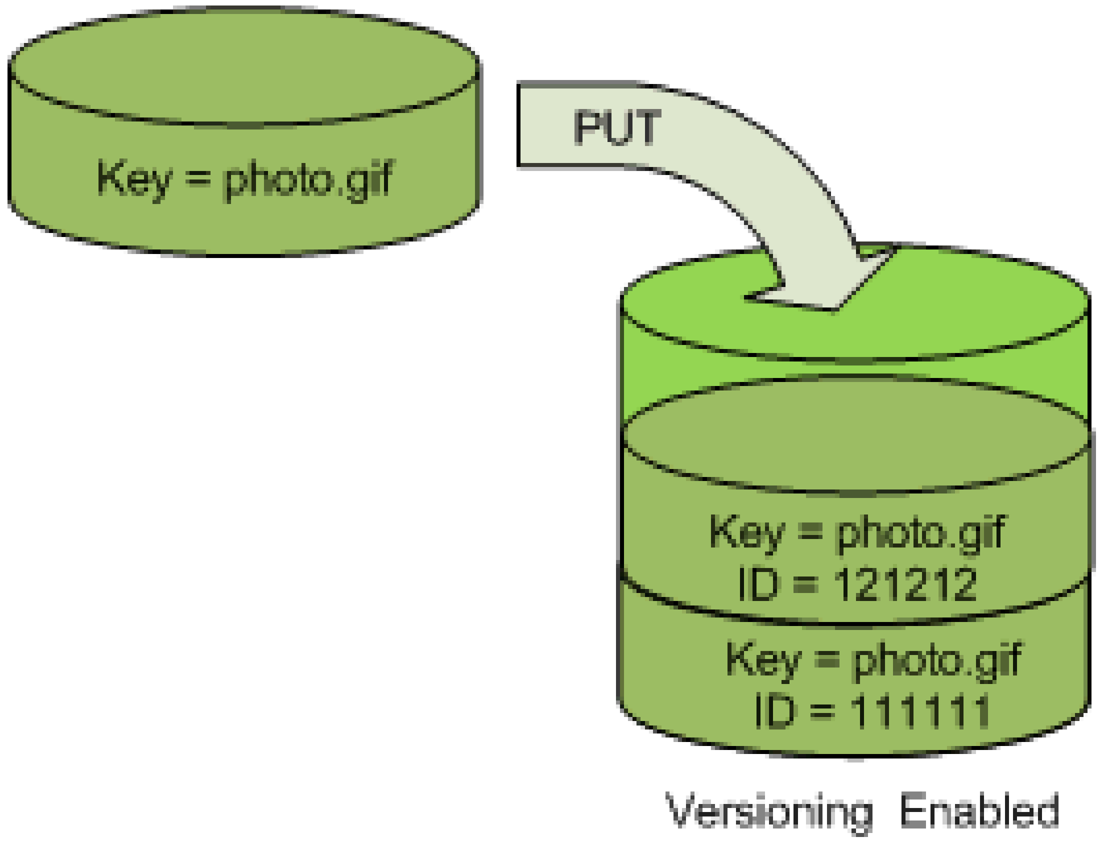
S3 Versioning

S3 versioning is a feature that keeps track of and stores all versions of an object so that you can access and use an older version if you like.

Versioning can only be set on the **bucket level** and applies to all objects in the bucket.

Protect against unintended deletes (ability to restore a version)

Read more: [Object Versioning]



S3 static web hosting

AWS recommends to block all public access to your bucket.

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both.

With public access, you can host a **static website** using Amazon S3. Furthermore, you can use AWS CloudFront (CDN service) on top of the S3 bucket so your static website will be cached globally.

If you get a 403 (Forbidden) error, make sure the bucket policy allows public reads.

The website URL will be:

`<bucket-name>.s3-website.<AWS-region>.amazonaws.com`

S3 Global CloudFront

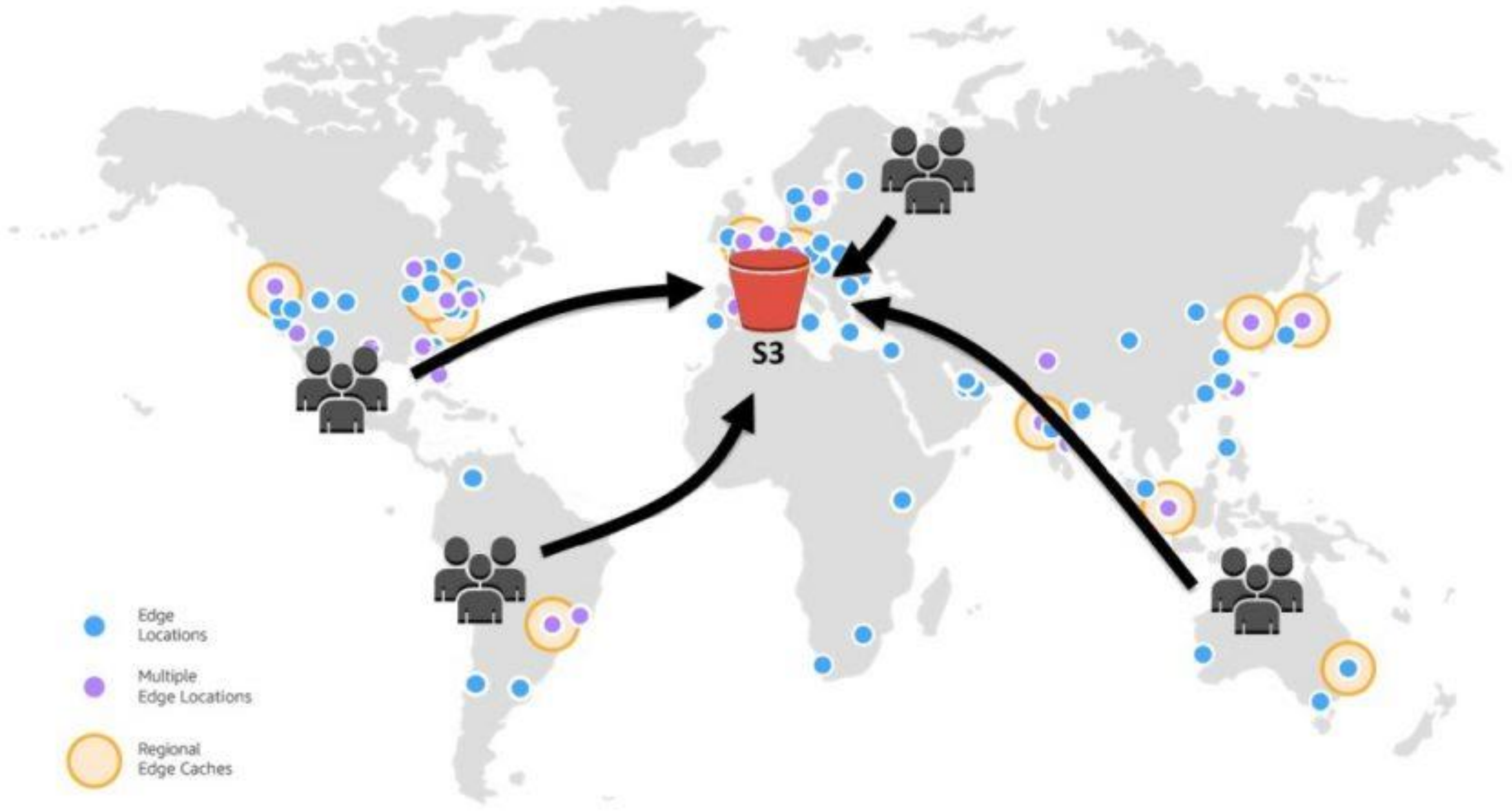
If you have a website, application, or another web resource, you probably have static content. Static content includes files like images, videos, or music, or even scripts like .css or js.

In the pre-cloud era, you would put those files on a standard server, and then serve them on the internet to all of your viewers, across the globe, from one specific geo location.

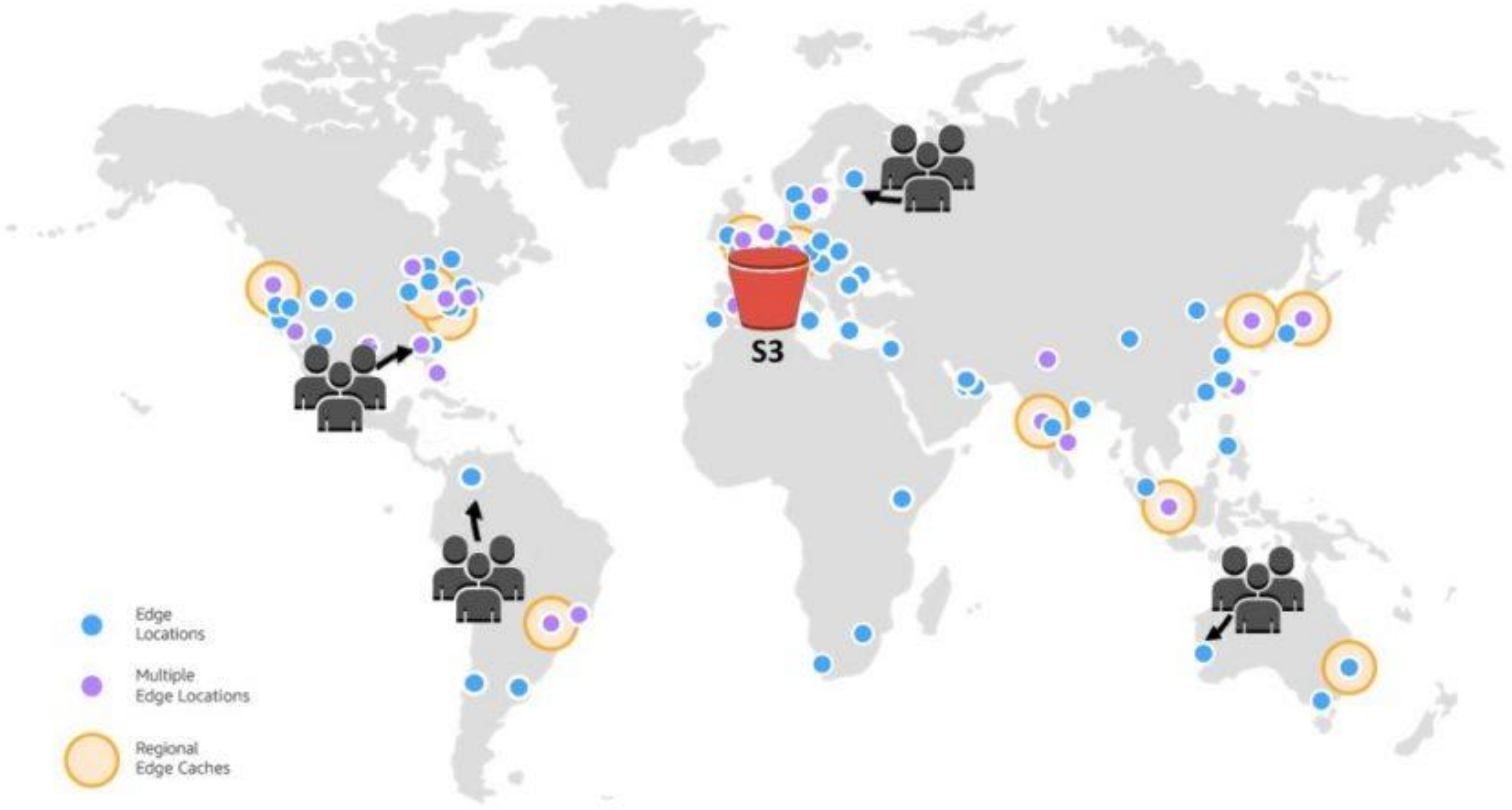
But with cloud services, there’s a solution that provides **faster delivery** and **better scalability**.

Host your static web on S3 and distribute the content with AWS CloudFront that is a **Content Delivery Network** (CDN) service.

Static contents on S3 without CloudFront

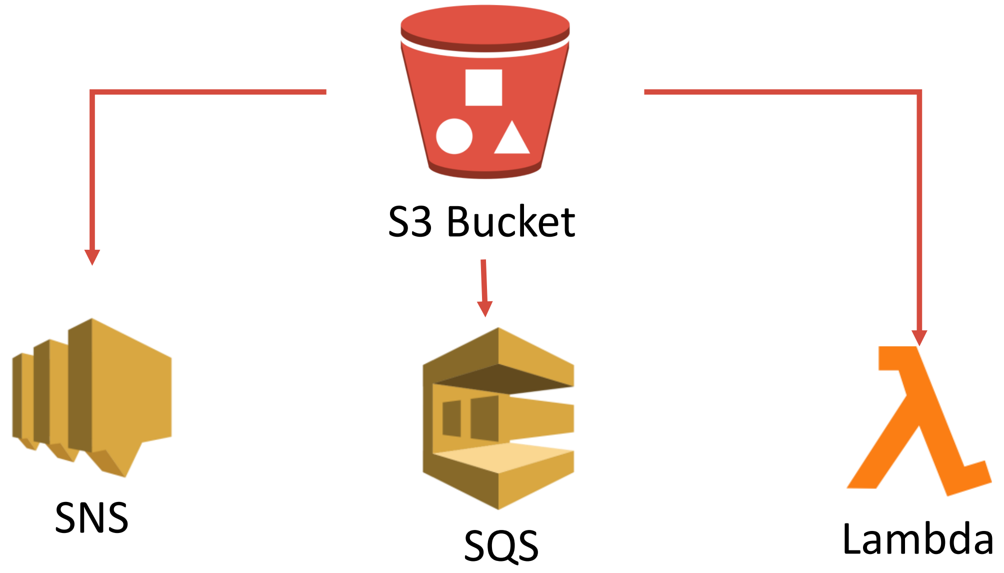


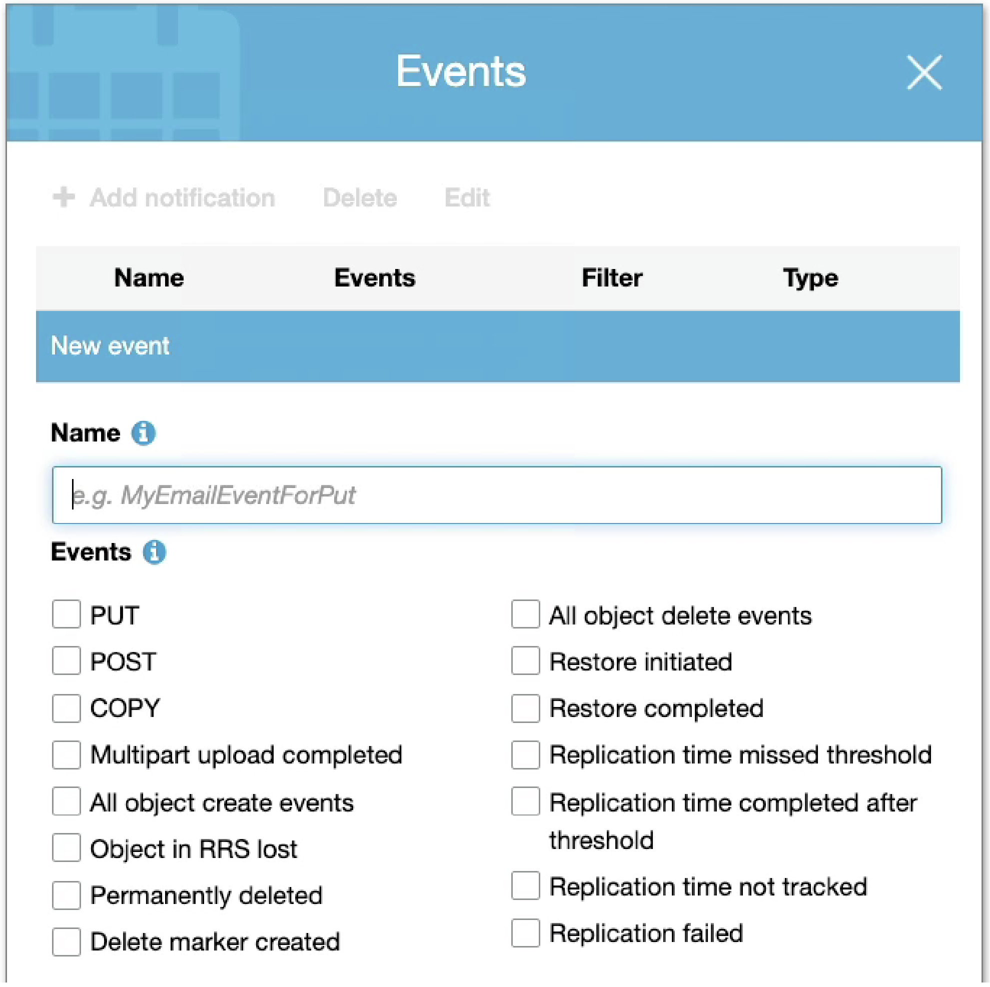
Static contents on S3 with CloudFront



S3 Event Notifications

You can use the Amazon S3 Event Notifications feature to receive notifications when certain events happen in your S3 bucket such as a new object is created, object got removed.





Name	Events	Filter	Type
New event			

Name

e.g. MyEmailEventForPut

Events

☐ PUT	☐ All object delete events
☐ POST	☐ Restore initiated
☐ COPY	☐ Restore completed
☐ Multipart upload completed	☐ Replication time missed threshold
☐ All object create events	☐ Replication time completed after threshold
☐ Object in RRS lost	☐ Replication time not tracked
☐ Permanently deleted	☐ Replication failed
☐ Delete marker created	

S3 Pre-Signed URLs

Only the object owner has permission to access objects in S3 bucket. However, the object owner can optionally share objects with others by creating a presigned URL to **grant time-limited permission to download the objects**.

Valid for 3600 seconds by default.

```
aws s3 presign s3://for-sale/book.pdf
```

Output:

```
https:// for-
sale.s3.amazonaws.com/book.pdf?AWSAccessKeyId=AKIAEXAMPLEACCESSKEY&Signat
ure=EXHCcBe%EXAMPLEKnz3r8O0AgEXAMPLE&Expires=1555531131
```

S3 Global Replication

Replication enables automatic, asynchronous copying of objects across Amazon S3 buckets between different accounts and **regions**.

Object may be replicated to a single destination bucket or multiple destination buckets.

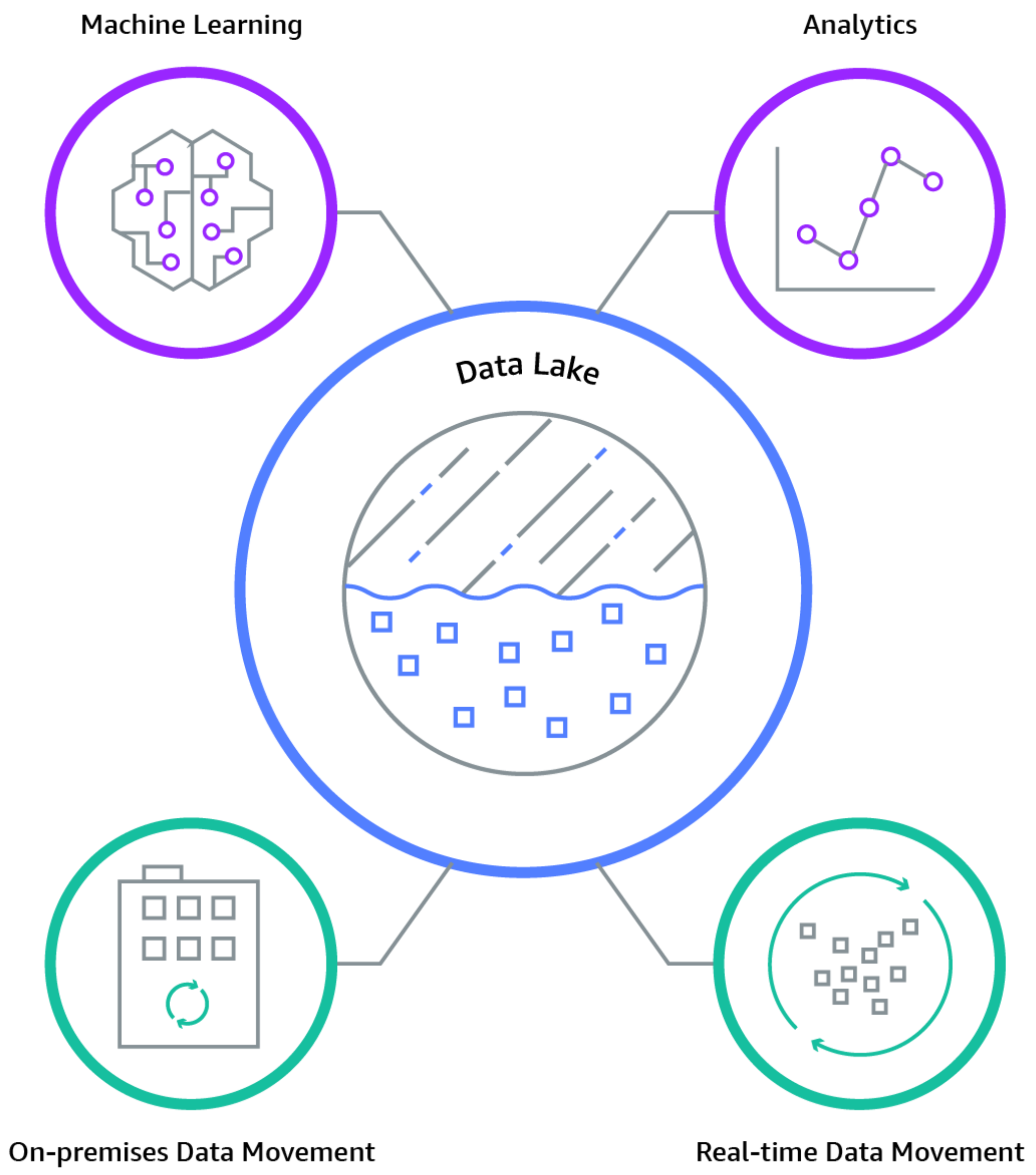
Replicate objects in less than 15 minutes.

AWS DataLake

A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale.

The data lake is essentially all organization data on AWS S3.

Organizations that successfully generate business value from their data, will outperform their peers.



Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely.

Object Lock provides two ways to manage object retention:

- Retention period — Specifies a fixed period of time during which an object remains locked. During this period, your object is WORM-protected and can't be overwritten or deleted.
- Legal hold — Provides the same protection as a retention period, but it has no expiration date. Instead, a legal hold remains in place until you explicitly remove it.

Object Lock works only in versioned buckets.

Multipart Upload

Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order.

If transmission of any part fails, you can retransmit that part without affecting other parts.

After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object.

In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation.

Multipart Upload benefits

- **Improved throughput** - You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues** - Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- **Pause and resume object uploads** - You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or stop the multipart upload.
- **Begin an upload before you know the final object size** - You can upload an object as you are creating it.

S3 Encryption

There are 4 methods of encrypting objects in S3:

- **SSE-S3**: encrypts S3 objects using keys handled & managed by AWS
- **SSE-KMS**: AWS Key Management Service to manage encryption keys
- **SSE-C**: when you want to manage your own encryption keys
- **Client Side Encryption**

AWS S3 Pricing

Pricing vary **by region** (charged per GB used) and **by type of request**. For example, storage price is much cheaper when using IA type of bucket. But when you retrieve an object, it costs more.

It charges for:

- Storage
- Requests & data retrieval
- Data transfer
- Management & analytics
- Replication

With Requester Pays buckets, the requester pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing data.

Read more about Storage Pricing

Amazon Relational Database Service (RDS)

RDS is a web service that makes it easier to set up, operate, and **scale** a **relational database** in the AWS Cloud.

It provides cost-efficient, resizable (you can only scale up) capacity for an industry-standard relational databases and manages common database administration tasks.

RDS benefits

- Amazon RDS manages **backups**, software patching, automatic failure detection, and recovery.
- To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also **restricts access** to certain system procedures and tables that require advanced privileges.
- Create **read replicas** to increase read scaling.
- You can get high availability by deploying read replicas (instances) in multiple AZs and **fail over** when problems occur.

DB instance and engine

- **DB instance** is an isolated database environment in the AWS Cloud. Your DB instance can contain multiple databases. You run DB instance in the **VPC** in private subnet. A **security group** controls the access to a DB instance.
- Each DB instance run **DB engine** such as MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server. Each DB engine has its own supported features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases.

RDS parameter group

You manage your **DB engine configuration** by associating your DB instances with **parameter groups**.

Examples of configs in parameter group:

- Increase maximum number of connections.
- Enable BinLog.

Read Replicas

A read replica is a copy of the primary instance that reflects changes to the primary in almost real time, in normal circumstances. You can use a read replica to offload read requests or analytics traffic from the primary instance. Additionally, for disaster recovery, you can perform a regional migration.

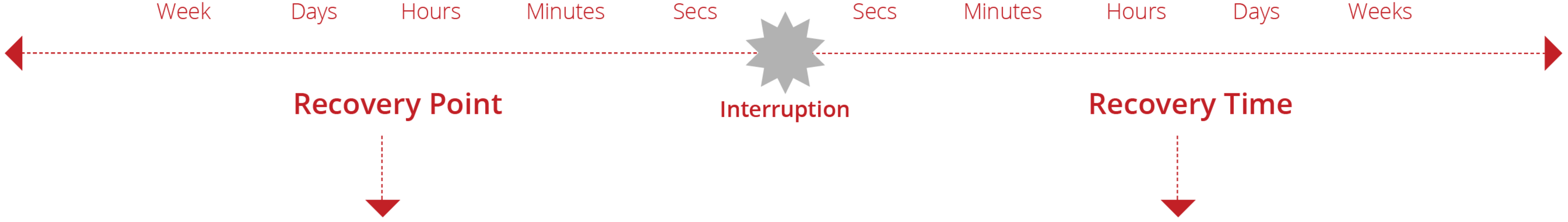
Amazon RDS Read Replicas provide

- enhanced performance
- scalability
- increased read throughput
- durability

Amazon RDS uses

- the **engines' native asynchronous replication** to update the read replica whenever there is a change to the source DB instance.
- or the shared cloud storage in Aurora.

RPO vs. RTO



RECOVERY POINT OBJECTIVE

- Focuses on how you rebound from the loss of your data
- The interval of time between data backups and the loss of data
- Determines how often you should backup your data
- Considers how often your data changes

RECOVERY TIME OBJECTIVE

- Focuses on your business as a whole
- How fast you need to recover your data
- Determines how much preparation and budget you need to recover
- Considers how much downtime you can handle

RDS Backups and Snapshots

Backups are **automatically enabled** in RDS daily. The retention period is 7 days by default. You can decrease or increase it up to 35 days.

There is also the **continuous backup** and point-in-time recovery (PITR) feature. With this, database and backup administrators are able to reduce their recovery point objective (RPO) to 5 minutes or under. It is not supported in Amazon Aurora.

DB Snapshots are manually **triggered by the user**.

When you restore backups, you need to create and configure a new instance which takes approximately 45 minutes.

RDS automatic failover

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a **synchronous standby** replica in a different Availability Zone. Standby replica provides:

- data redundancy
- eliminate I/O freezes
- minimize latency spikes during system backups
- enhance availability during planned system maintenance

You can also failover to **read-replica** and promote read-replicate as a primary database instance.

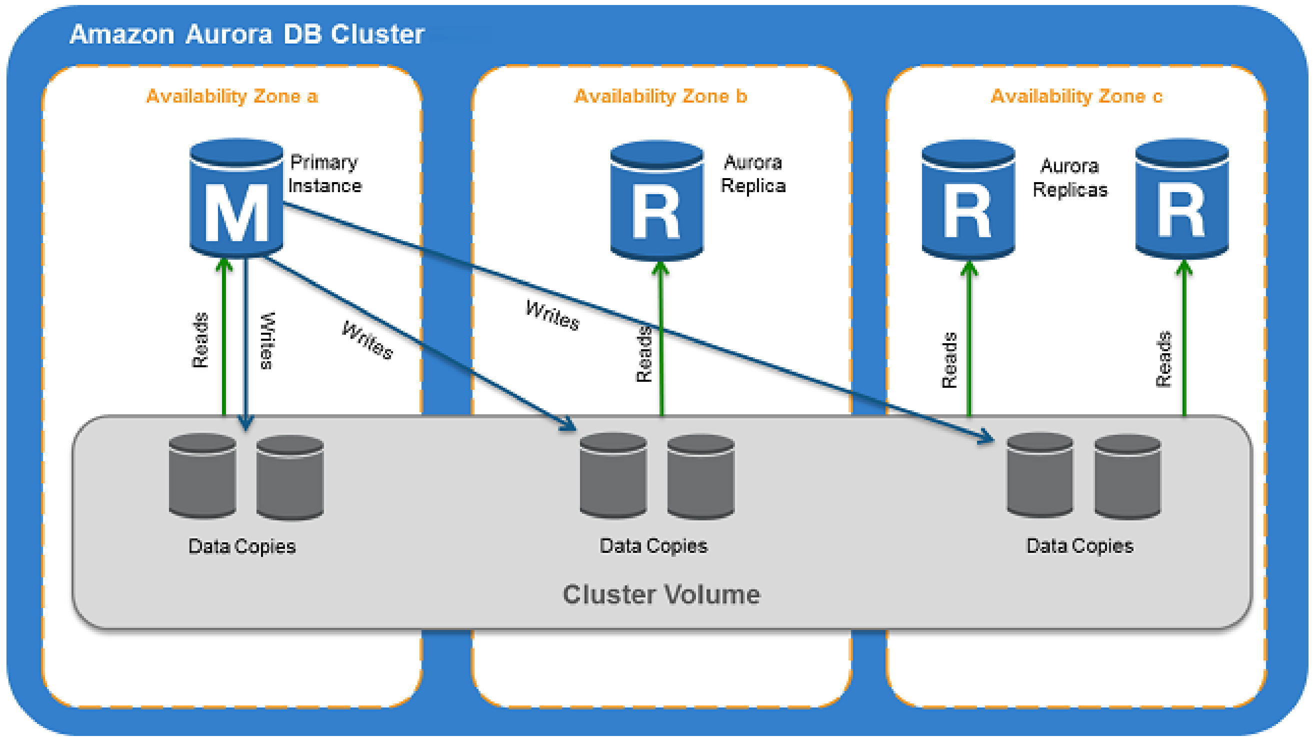
Amazon Aurora

AWS build Amazon Aurora on top of the open-source **MySQL** and **PostgreSQL**. It provides more features and is fully managed by AWS.

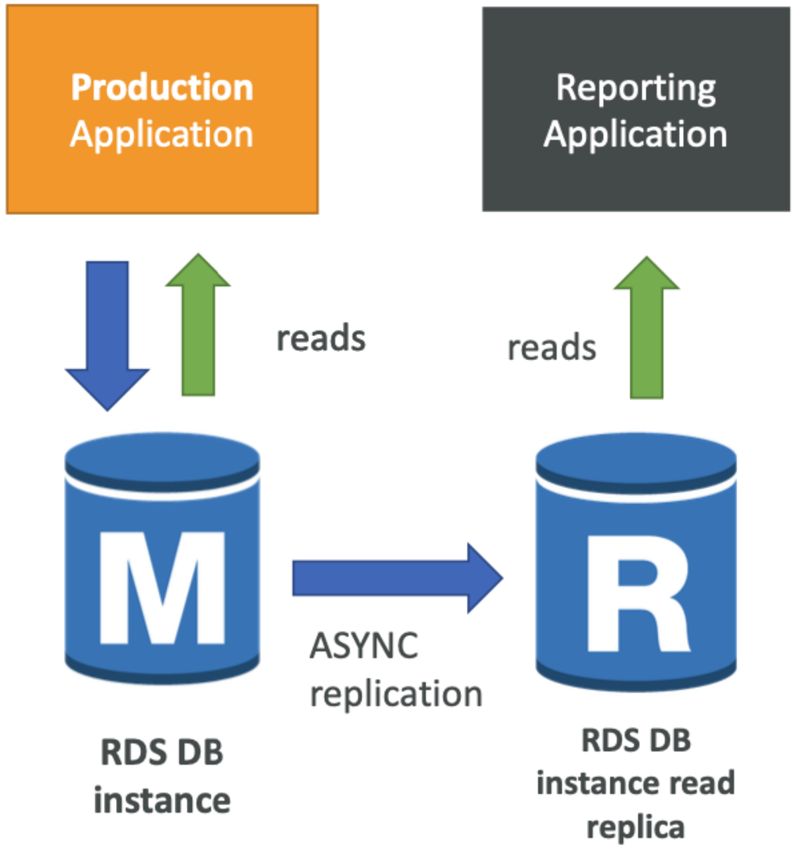
Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL. It takes advantage of fast **distributed storage**. The underlying storage grows automatically as needed.

Storage is decoupled in Amazon Aurora.

Amazon Aurora



Non-Aurora



Amazon Aurora Features

Backtracking – you return the state of an Aurora cluster to a specific point in time, without restoring data from a backup. It completes within seconds, even for large databases.

Global databases – a single database that spans multiple AWS Regions, enabling low-latency global reads and disaster recovery from any Region-wide outage. It provides built-in fault tolerance.

Parallel queries – provides faster analytical queries over your current data while maintaining high throughput for your core transactional workload.

Amazon Aurora DB clusters

An Amazon Aurora DB cluster consists of one or more **DB instances** and a **cluster volume.** An Aurora **cluster volume** is a virtual database storage volume that spans **multiple AZs**, with each AZ having a copy (replication) of the DB cluster data. Two types of DB instances make up an Aurora DB cluster:

1. **Primary DB instance** – Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary DB instance.
2. **Aurora Replica** – Connects to the same storage volume as the primary DB instance and supports **only read** operations.

Amazon Aurora Read Replica

Amazon Aurora replicas **share the same underlying storage** as the source instance, lowering costs and **avoiding the need to copy data** to the replica nodes.

Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary DB instance. Maintain **high availability** by locating Aurora Replicas in separate AZs or Regions.

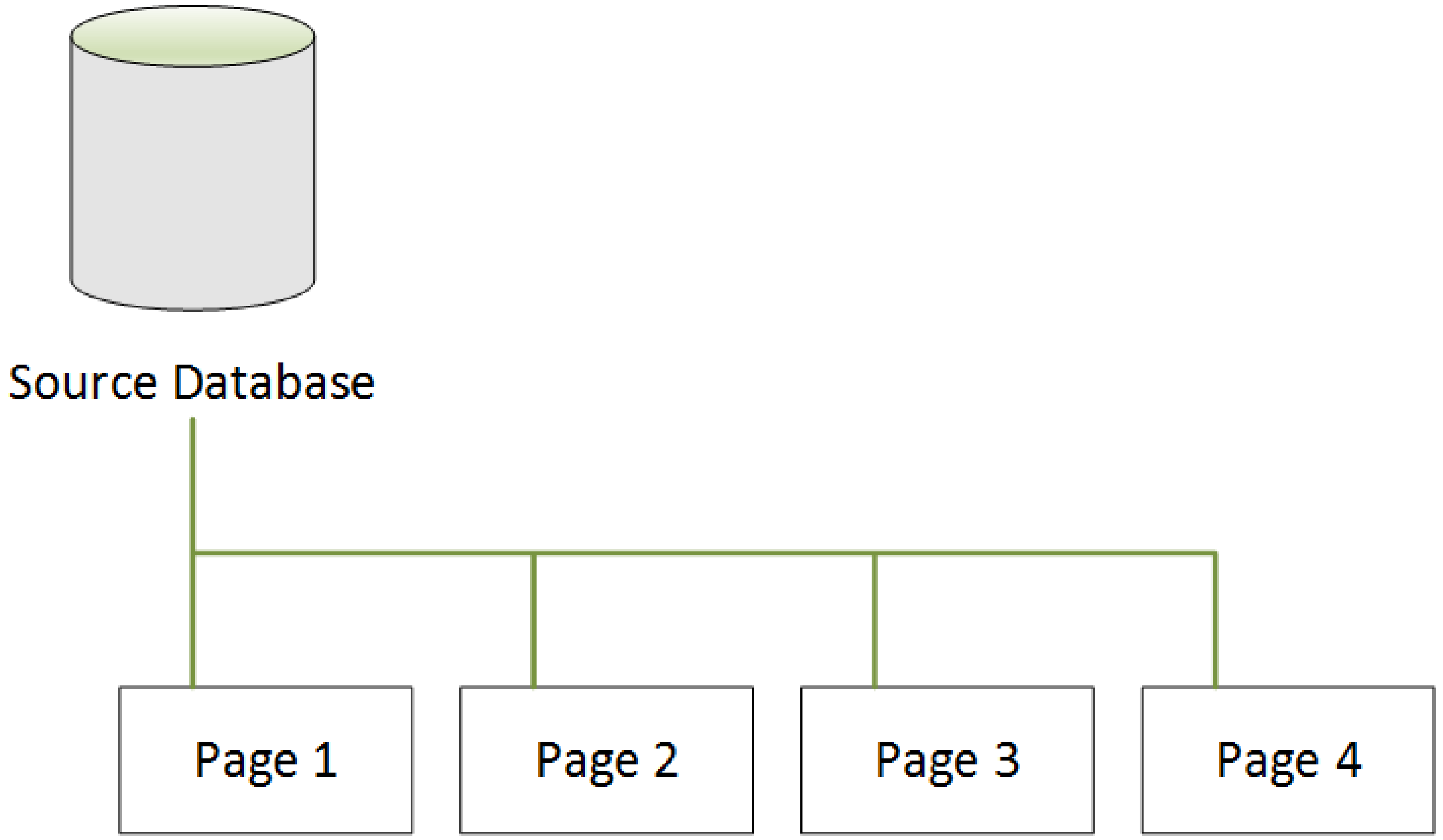
Aurora automatically **fails over** to an Aurora Replica in case the primary DB instance becomes unavailable.

Cloning an Aurora DB cluster volume

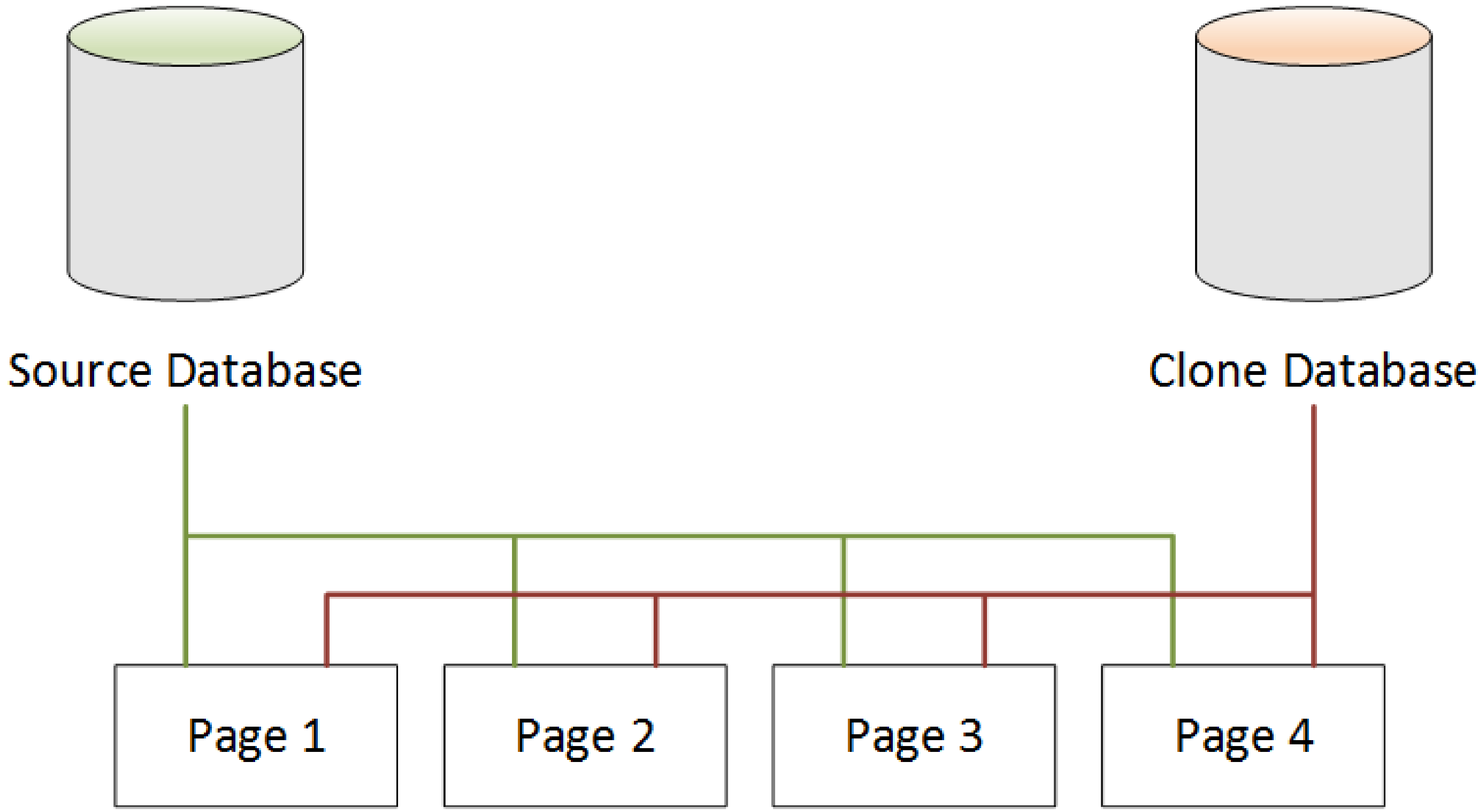
Using the Aurora cloning feature, you can quickly and cost-effectively create a new DB cluster containing a **duplicate of an Aurora cluster volume** and **all its data**. We refer to the new cluster and its associated cluster volume as a clone.

Creating a clone is faster and more space-efficient than physically copying the data using a different technique such as restoring a snapshot.

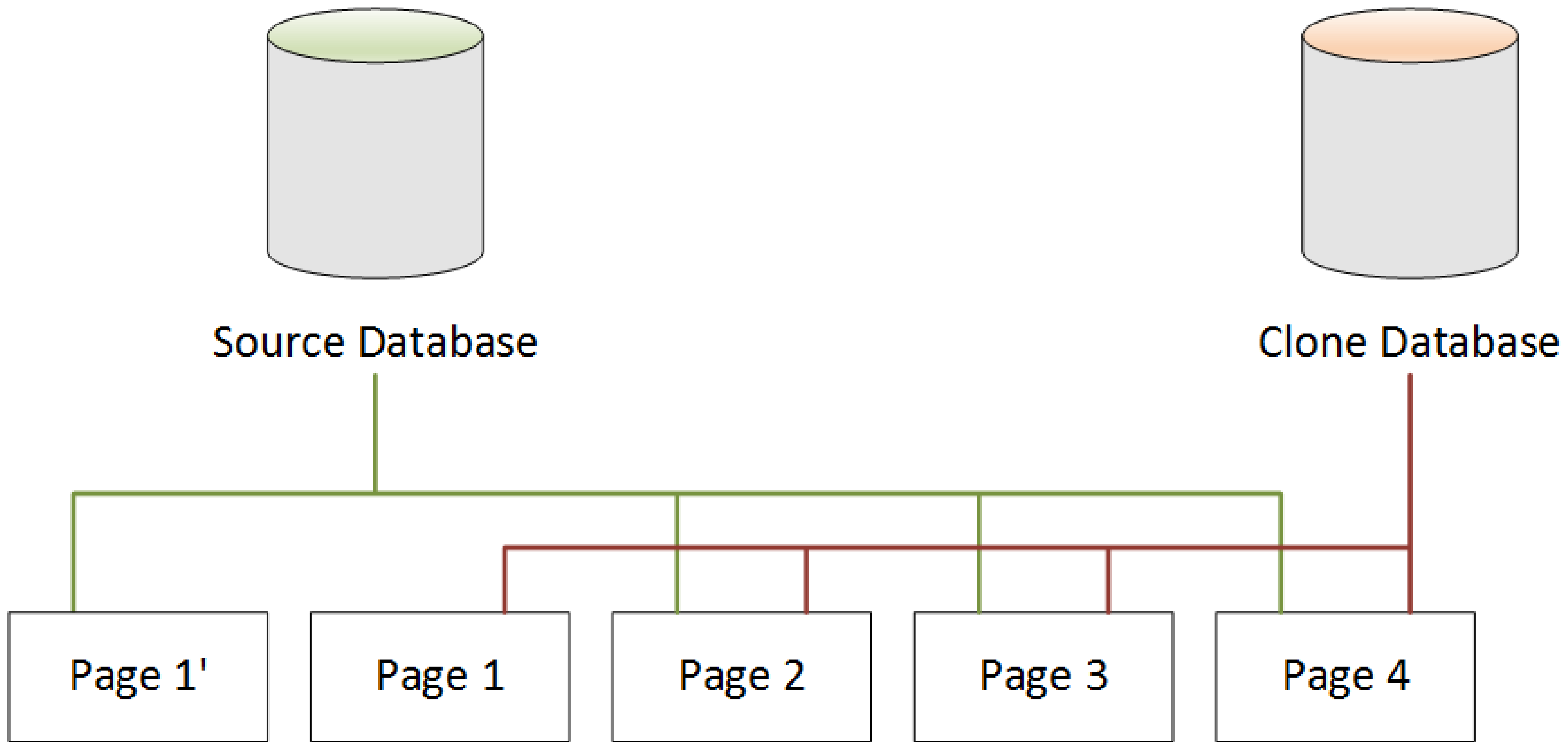
Before cloning



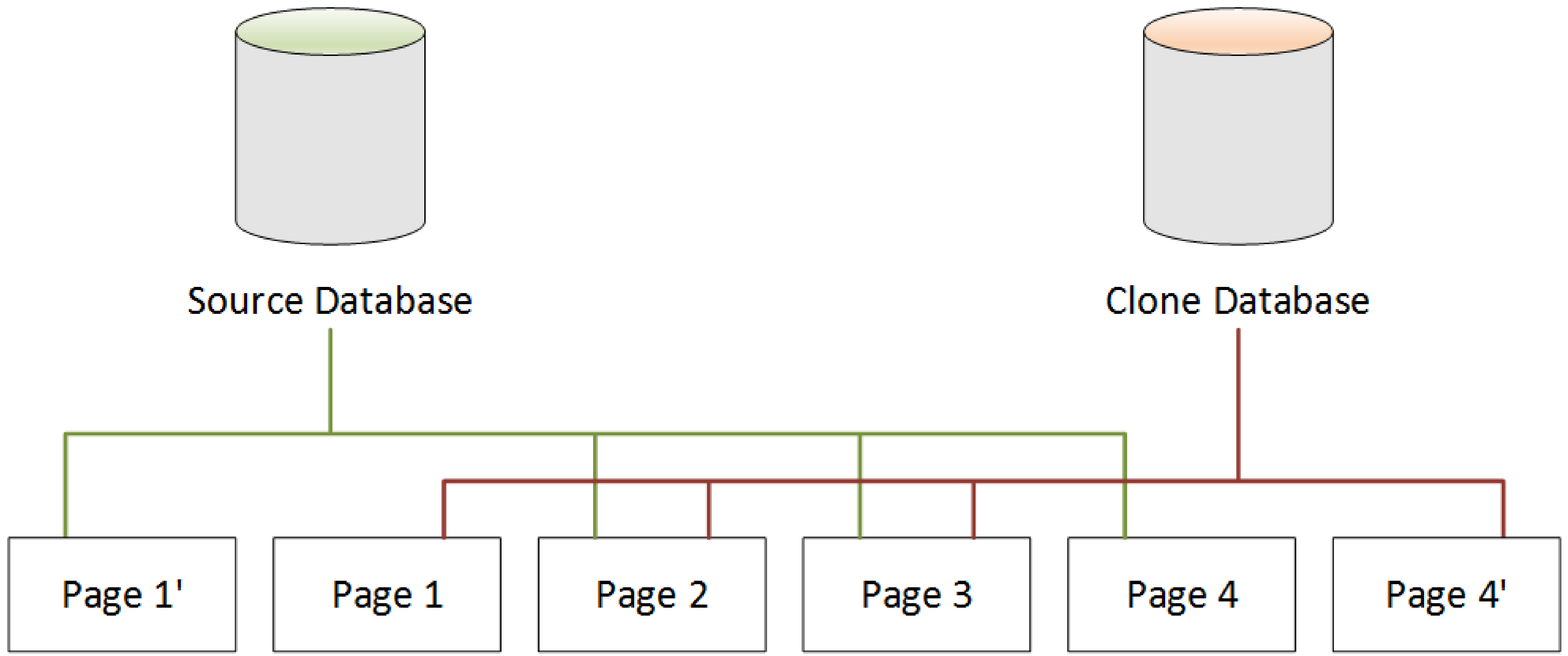
After cloning



When a change occurs on the source cluster volume



When a change occurs on the clone



Aurora serverless

Aurora serverless is an **on-demand autoscaling** configuration for Amazon Aurora. An Aurora Serverless DB cluster is a DB cluster that scales compute capacity up and down based on your application's needs.

Aurora Serverless is relatively simple, cost-effective option for **infrequent, intermittent,** or **unpredictable** workloads.

SQL vs NoSQL

- When to use NoSQL?
  - Transactional applications; OLTP (Online transaction processing)
  - You need hyper scale
  - Low latency
  - Flexible data models
  - Very large amounts of data and very large numbers of users
- When to use SQL?
  - Ad hoc queries (complex queries); data warehousing; OLAP (online analytical processing)
  - Your data is predictable and highly structured
  - Your workload volume is consistent

Read more: From SQL to NoSQL