

# AWS S3

***CS516 – Cloud Computing***  
***Computer Science Department***  
***Maharishi International University***

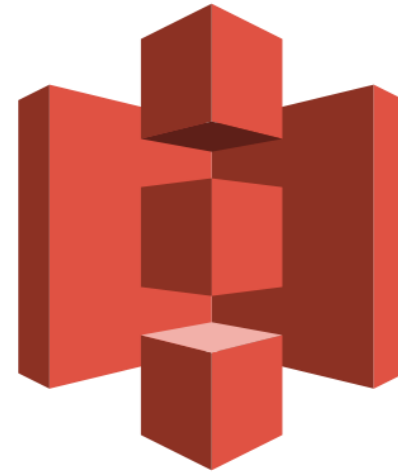
# Maharishi International University - Fairfield, Iowa



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University.

# Content

- Storage classes
- Object lifecycles
- Global replication
- VPC Gateway Endpoint for S3
- Static website hosting
- S3 with CloudFront
- Object versioning
- Object encryption



# Simple Storage Service - S3

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

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

In S3, you can store anything such as images, videos, blobs, and so on. You can even store a payload. And encrypt the payload with built-in S3 encryption.

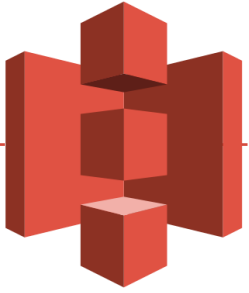
Learn more: [Introduction to Amazon S3](#)

# 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 identifies each object.

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

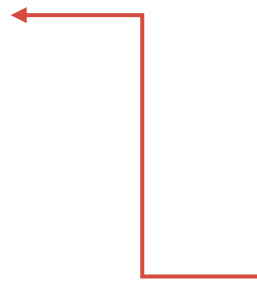
You can create a **folder** in an S3 bucket to organize your data.



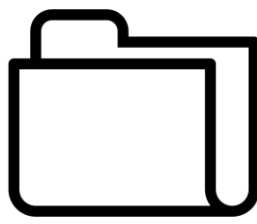
AWS S3 *AWS Region (U.S. Standard)*



Bucket



Object



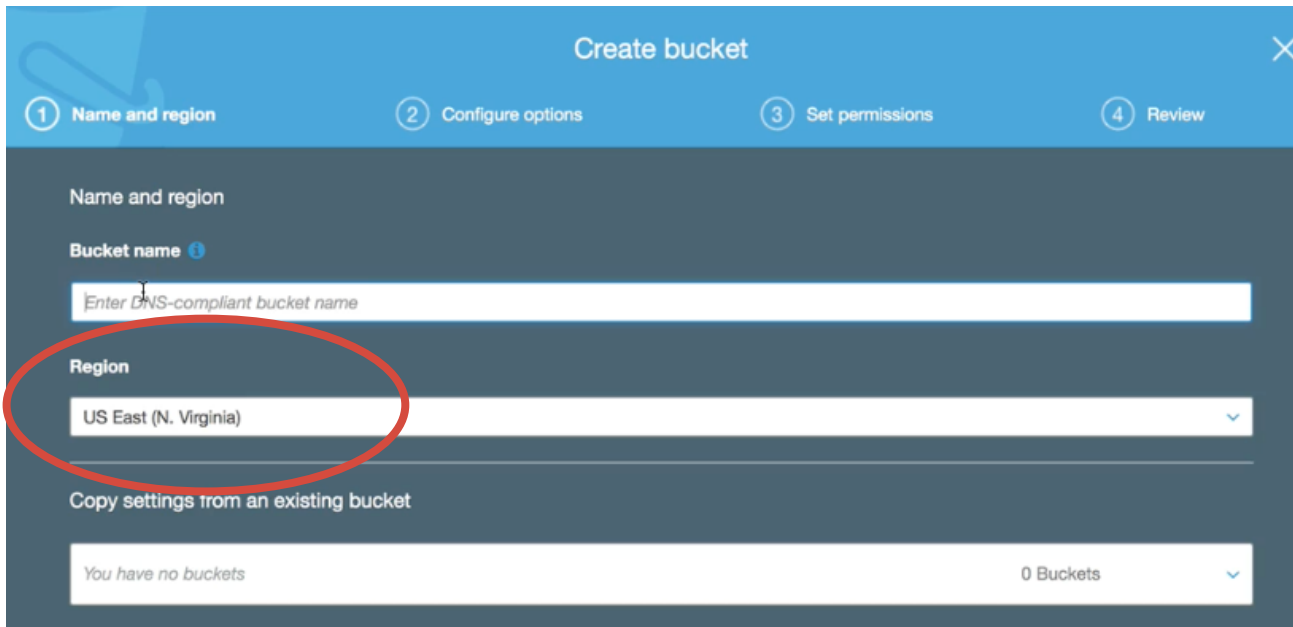
Folder



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. Later on, you can replicate your data in different regions.



The screenshot shows the 'Create bucket' wizard in the AWS Management Console. The first step, 'Name and region', is active. It contains a text input for 'Bucket name' with a placeholder 'Enter DNS-compliant bucket name'. Below it, the 'Region' dropdown menu is highlighted with a red circle, showing 'US East (N. Virginia)' as the selected option. At the bottom, there is a section for 'Copy settings from an existing bucket' with a message 'You have no buckets' and a dropdown showing '0 Buckets'.

Best practice is to select the region that is physically closest to you, to reduce transfer latency. You can also stack up CloudFront on top of the S3 to reduce latency and save costs.

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

# 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** levels.

There are 3 types of permissions in S3:

- **Identity-based** – IAM user or role has policies to access S3.
- **Resource-based** – You can define policies on the S3 bucket itself with the principal.
- **Access Control List (ACL)** – Sets policies to both a bucket and an object. With identity-based and resource-based policies, you cannot **assign a policy to an object**.

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

Read more about [IAM in Amazon S3](#)



# Block vs File vs Object storages

File storage stores data as a single piece of information **in a folder** to help organize it among other data. Our laptops.

Block storage takes a file apart **into singular blocks** of data and then stores these blocks as separate pieces of data. Complex but fast. Used in servers.

Object storage **is a flat structure** in which files are spread out among hardware. Unlimited scaling. The cloud.

Read more: [File storage, block storage, or object storage?](#)

# S3 design principles

- **Decentralization** – Distributed system since running on many servers in the cloud. The application we are running in the cloud is also a distributed system.
- **Decompose** into small building blocks and autonomous modules with local responsibility – Similar to microservices or React components or Java classes
- **Simplicity** – An EventBridge also provides simple solutions for complex apps.
- **Asynchrony** is loosely coupled which improves latency and fault tolerance.
- **Controlled concurrency** and **controlled parallelism**. It is managed. You don't need to worry about concurrency, deadlock, etc. S3 kept parallelism in mind from the start. You can read and write at the same time. You can also increase the application read performance by parallelizing requests. S3 provides controlled concurrency out of the box. There is no connection limit. You can have many clients connecting to S3 without worrying.

[AWS re:Invent 2022 - Keynote with Dr. Werner Vogels](#)

# 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 about [Amazon S3 Storage Classes](#)

# S3 Storage Classes

- **S3 Standard** - For mobile applications, content distribution. (\$0.023 per GB)
- **S3 Standard - Infrequent Access (IA)** - For backups. (\$0.0125 per GB)
- **S3 One Zone IA** - Secondary backup, data you can recreate. (\$0.01 per GB)
- **Glacier** - For long-term archival storage. (\$0.004 per GB)
- **Glacier Deep Archive** - For long-term archival storage.
- **S3 Intelligent-Tiering** - Automatically moves objects between two access tiers based on access patterns

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

**Object Availability** is the percent (%) over a one-year time period that a file stored in S3 **will be accessible**.

	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

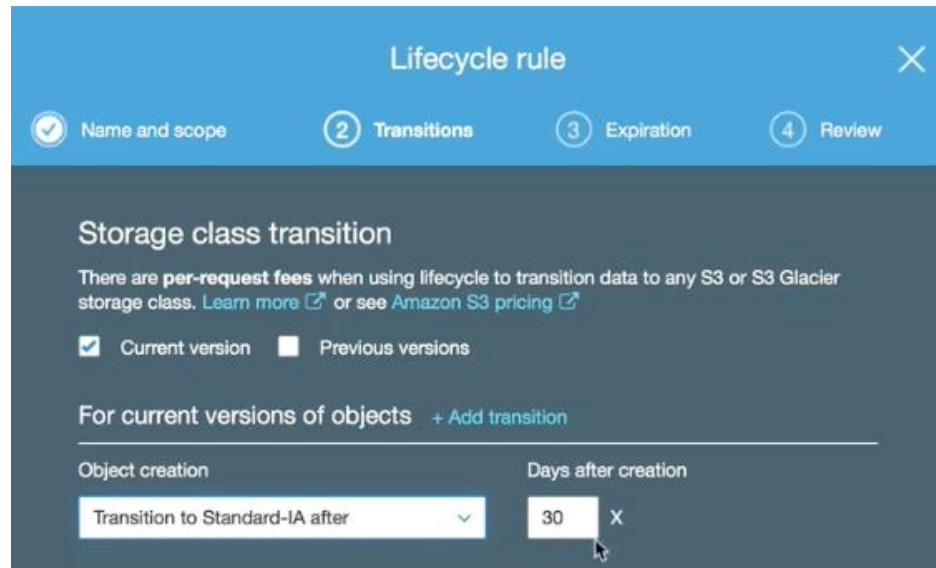
# 11 Nines Data Durability

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 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.



The screenshot shows the 'Lifecycle rule' configuration window in the AWS console. The 'Transitions' tab is selected, showing options for 'Storage class transition'. A checkbox for 'Current version' is checked. Below, a table shows a transition rule: 'Transition to Standard-IA after' with a value of '30' days. The interface includes a progress bar at the top with four steps: 'Name and scope', 'Transitions', 'Expiration', and 'Review'.

**Lifecycle rule**

1 Name and scope 2 **Transitions** 3 Expiration 4 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 X

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 user needs and minimize storage costs?

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



# 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 static web hosting

AWS recommends blocking all public access to your bucket. Because all actions (read/write) on S3 cost. With public access, you can host a **static website** using Amazon S3. If you get a 403 (Forbidden) error, make sure the bucket policy allows public reads. But making a bucket and its object public is not a good practice since bills can get higher.

The best practice is to put AWS CloudFront (CDN service) on top of the S3 bucket so your static website will be cached globally. Then you can create Origin Access Identity (**OAI**) and put that in the bucket policy.

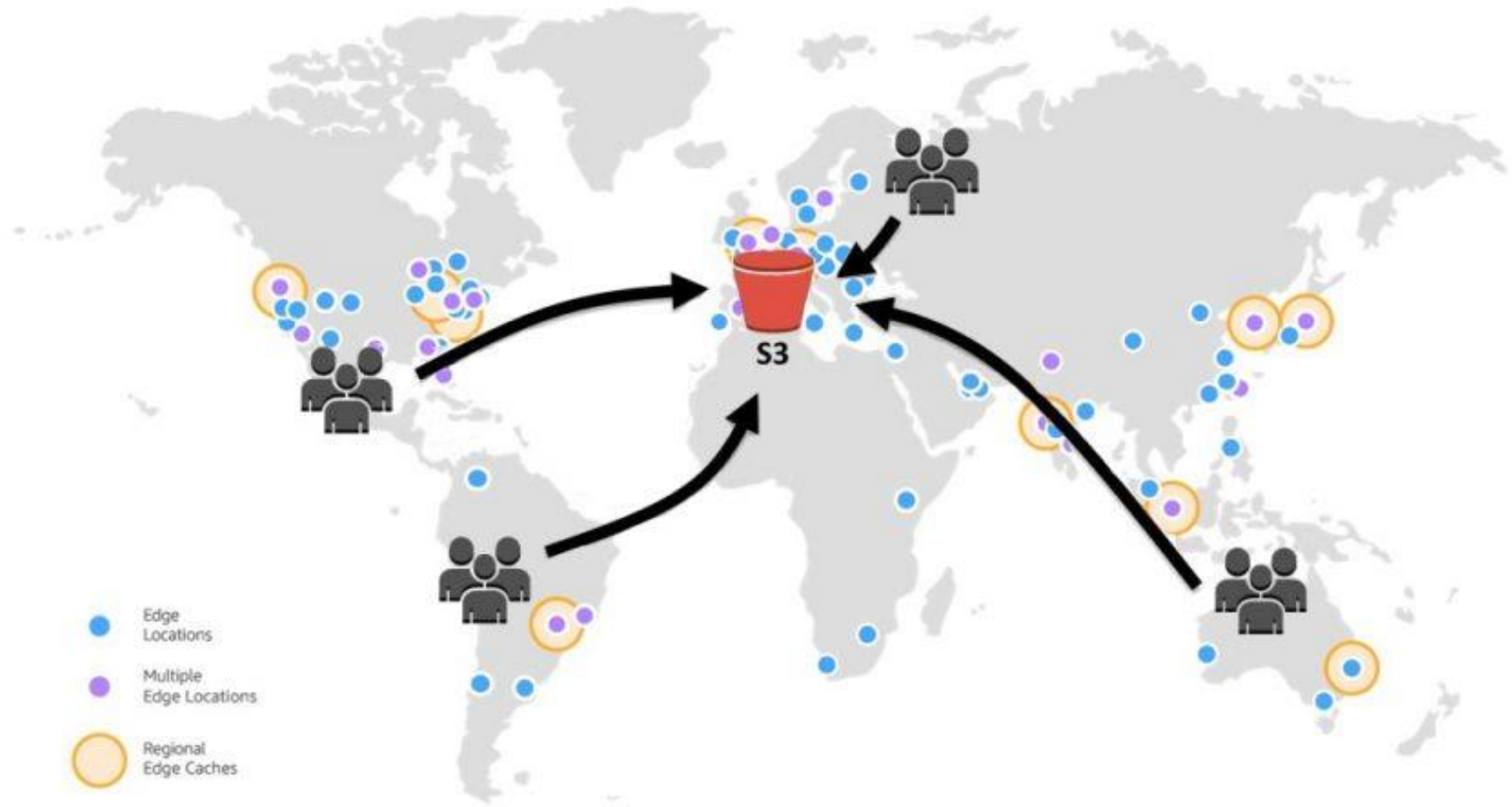
For more details on: [Restricting access to an Amazon S3 origin](#).

# 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 which is a Content Delivery Network (CDN) service.

# Static contents on S3 without CloudFront



# Static contents on S3 without CloudFront

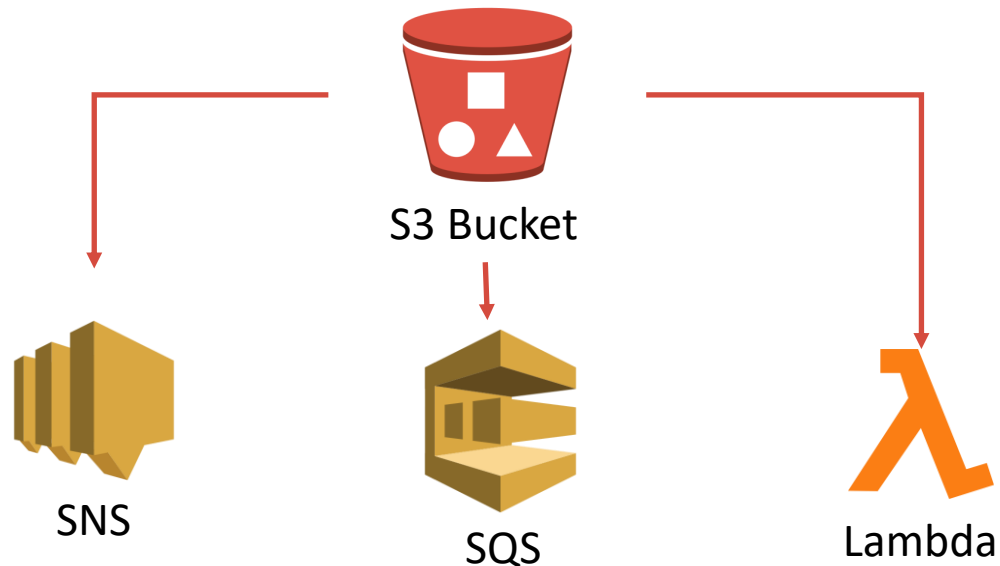


# CloudFront features

- CloudFront plays a security role by blacklisting countries.
- Reduces S3 cost. Because the content is delivered from the edge server and there are no API calls against S3. In AWS, most API calls charge.
- You can run business logic code with Lambda@Edge.
- You can implement an advanced caching behavior based on query strings and headers.
- You can serve private content.
- You can do video streaming.
- You can do a path routing. For example, if the request has “us” in the path, then forward it to a bucket in the US region. If it is “ap”, then forward it to Asia Pacific, etc.

# S3 Event Notifications

You can use the Amazon S3 Event Notifications feature to receive notifications when certain events happen in your S3 buckets such as a new object being created, or an object getting removed.



## Events

[+ Add notification](#) [Delete](#) [Edit](#)

Name	Events	Filter	Type
New event			

**Name** ⓘ

**Events** ⓘ

☐ PUT

☐ POST

☐ COPY

☐ Multipart upload completed

☐ All object create events

☐ Object in RRS lost

☐ Permanently deleted

☐ Delete marker created

☐ All object delete events

☐ Restore initiated

☐ Restore completed

☐ Replication time missed threshold

☐ Replication time completed after threshold

☐ Replication time not tracked

☐ Replication failed

# S3 Global Replication

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

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

Customers needing a predictable replication time backed by a Service Level Agreement (SLA) can use Replication Time Control (RTC) to replicate objects in less than 15 minutes.



# When to use S3 Replication

- **Data redundancy** – If you need to maintain multiple copies of your data in the same, or different AWS Regions, or across different accounts. S3 Replication powers your global content distribution needs, compliant storage needs, and data sharing across accounts. Replica copies are identical to the source data, that retain all metadata, such as the original object creation time, ACLs, and version IDs.
- **Replicate objects to more cost-effective storage classes** — You can use S3 Replication to put objects into S3 Glacier, S3 Glacier Deep Archive, or another storage class in the destination buckets.
- **Maintain object copies under a different account**

# When to use Cross-Region Replication

S3 Cross-Region Replication (**CRR**) is used to copy objects across Amazon S3 buckets in different AWS Regions.

- **Meet compliance requirements** — Although Amazon S3 stores your data across multiple geographically distant Availability Zones by default, compliance requirements might dictate that you store data at even greater distances (regions).
- **Minimize latency** — If your customers are in two geographic locations, you can minimize latency in accessing objects by maintaining object copies in AWS Regions that are geographically closer to your users.
- **Increase operational efficiency** — If you have compute clusters in two different AWS Regions that analyze the same set of objects, you might choose to maintain object copies in those Regions.

# When to use Same-Region Replication

Same-Region Replication (**SRR**) is used to copy objects across Amazon S3 buckets in the same AWS Region. SRR can help you do the following:

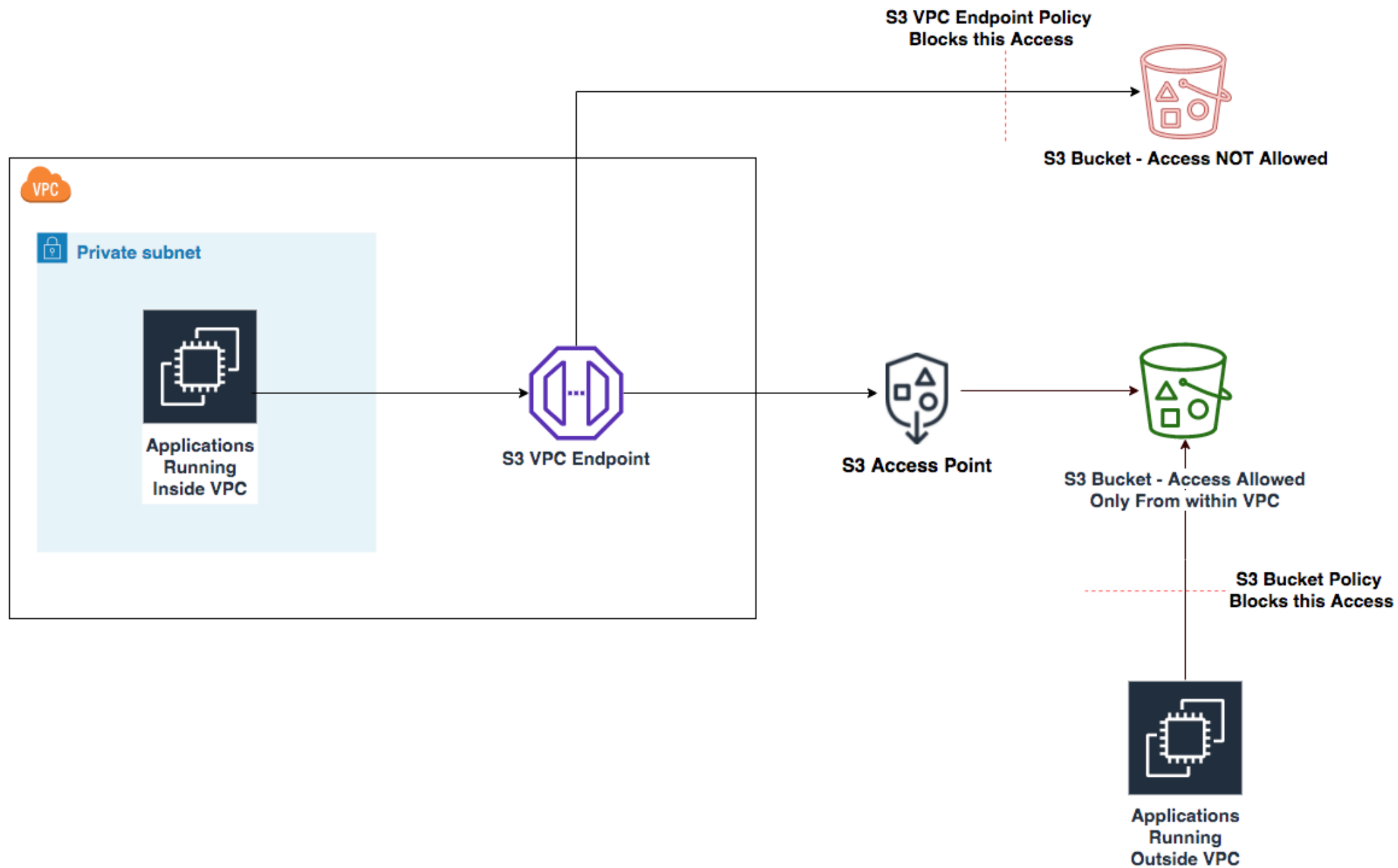
- **Aggregate logs into a single bucket** — If you store logs in multiple buckets or across multiple accounts, you can easily replicate logs into a single, in-Region bucket. This allows for simpler processing of logs in a single location.
- **Configure live replication between production and test accounts** — If you or your customers have production and test accounts that use the same data, you can replicate objects between those multiple accounts, while maintaining object metadata.

# S3 Gateway Endpoint

VPC endpoints are easy to configure, highly reliable, and provide a secure connection to public AWS resources such as S3 without going out to the public internet. It uses AWS infrastructure. Hence, it is much safer and faster.

You can use an S3 bucket policy to indicate which VPCs and which VPC endpoints have access to your S3 buckets.

For more details: [Amazon S3 and VPC Endpoints](#)

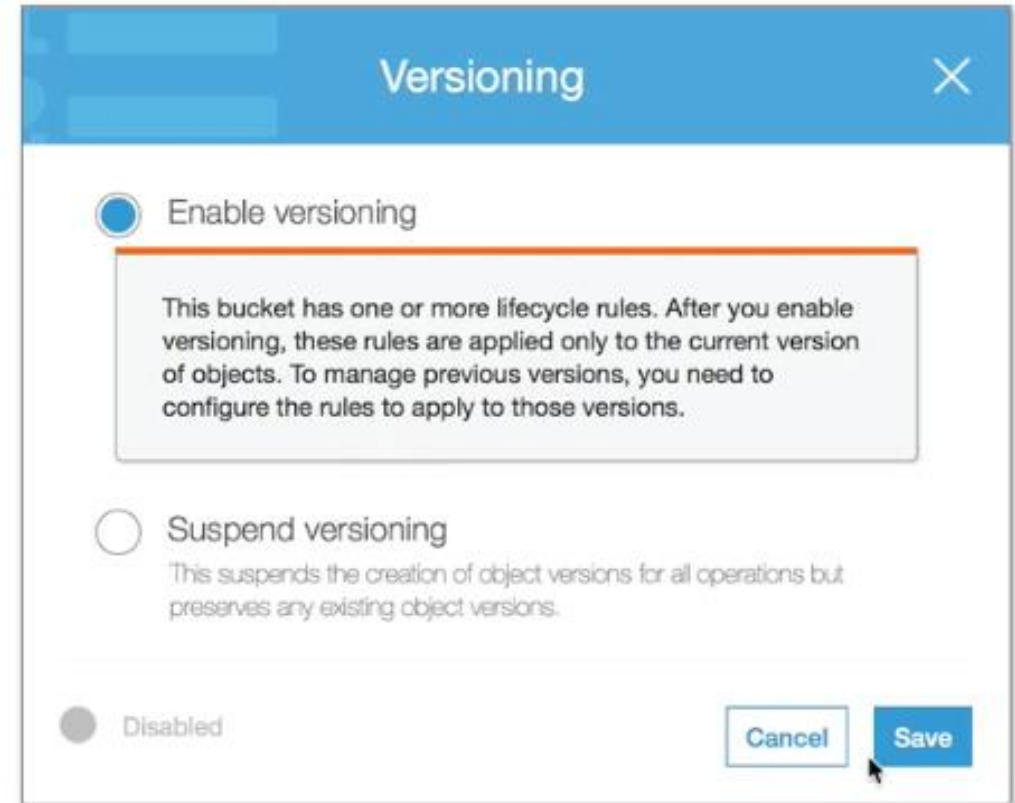


# S3 Versioning

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.

Read more about [Object Versioning](#)



The screenshot shows the 'Versioning' configuration window for an S3 bucket. The window has a blue header with the title 'Versioning' and a close button (X). Below the header, there are three radio button options: 'Enable versioning' (which is selected), 'Suspend versioning', and 'Disabled'. A warning box is present next to the 'Enable versioning' option, stating: 'This bucket has one or more lifecycle rules. After you enable versioning, these rules are applied only to the current version of objects. To manage previous versions, you need to configure the rules to apply to those versions.' At the bottom right, there are 'Cancel' and 'Save' buttons. A mouse cursor is pointing at the 'Save' button.

Protect against unintended deletes (ability to restore a version)

# 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.

# S3 Pre-Signed URLs

The most important feature. It enables storing blobs (image, video, html, JS, so on) directly from the client app without keeping any secret tokens in your React app. Again very important. Everything in React app is public!! Because it is on the client side.

S3 Pre-Signed URLs allows clients to download an object from the bucket with a temporary URL.

Learn more: [Securing AWS S3 uploads using presigned URLs](#)



# 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.

Makes your app faster and complex. For example, more parameters to provide and edge cases such as what happens when there are incomplete files.

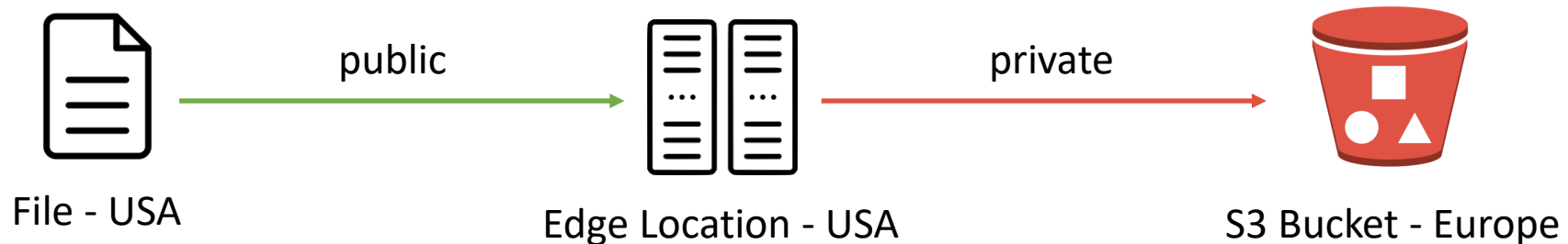
# Multipart Upload benefits

- **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 Transfer Acceleration (upload only)

Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region directly. Similar to a Global Accelerator that we discussed briefly yesterday.

To accelerate downloading, you can use CloudFront or S3 Global Replication.



# S3 Encryption

There are 4 methods of encrypting objects in S3:

1. **SSE-S3**: Encrypts S3 objects using keys handled & managed by AWS. Out-of-box feature. Nothing to do on your side. All objects stored in S3 are encrypted.
2. **SSE-KMS**: You can use AWS Key Management Service which provides keys to encrypt your data in S3.
3. **SSE-C**: The key is generated by you (the appliance) and use that key to encrypt data.
4. **Client-Side Encryption**: Encrypt before storing in S3 in your client app.

# 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 an (Infrequent) IA type of bucket. But if the object is retrieved frequently, 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.

Delete an object API call doesn't cost. But you must get an object to delete that cost. Fetching 1 million objects costs around \$5 depending on the region. It might take 30 minutes to delete or replicate 1 million objects based on a Medium blog. But you will never know the exact time the object is deleted. Because it is done by the AWS and the S3 itself is async. Maybe are they deleting them once a day at night? Who knows.

Read more about [Storage Pricing](#)