Storage services
EBS, EFS, S3, Glacier, Storage Gateway

AWS Certified Solutions Architect– Professional (SAP-C01)
Study notes - Sep'2019

AWS Storage options

If You Need:	Consider Using:
Persistent local storage for Amazon EC2, for relational and NoSQL databases, data warehousing, enterprise applications, Big Data processing, or backup and recovery	Amazon Elastic Block Store (EBS)
A simple, scalable, elastic file system for Linux-based workloads for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, so your applications have the storage they need – when they need it.	Amazon Elastic File System (Amazon EFS)
A fully managed file system that is optimized for compute-intensive workloads, such as high performance computing, machine learning, and media data processing workflows, and is seamlessly integrated with Amazon S3	Amazon FSx for Lustre
A fully managed native Microsoft Windows file system built on Windows Server so you can easily move your Windows-based applications that require file storage to AWS, including full support for the SMB protocol and Windows NTFS, Active Directory (AD) integration, and Distributed File System (DFS).	Amazon FSx for Windows File Server
A scalable, durable platform to make data accessible from any Internet location, for user-generated content, active archive, serverless computing, Big Data storage or backup and recovery	Amazon Simple Storage Service (Amazon S3)
Highly affordable long-term storage that can replace tape for archive and regulatory compliance	Amazon Glacier
A hybrid storage cloud augmenting your on-premises environment with Amazon cloud storage, for bursting, tiering or migration	AWS Storage Gateway
A portfolio of services to help simplify and accelerate moving data of all types and sizes into and out of the AWS cloud	Cloud Data Migration Services
A fully managed backup service that makes it easy to centralize and automate the back up of data across AWS services in the cloud as well as on premises using the AWS Storage Gateway.	AWS Backup

S3

AWS S3 - Primer

Amazon S3 Standard, S3 Standard–IA, S3 One Zone-IA, and S3 Glacier are all designed to provide **99.999999999%** durability of objects over a given year. This is achieved by storing objects on multiple devices across a minimum of three Availability Zones (AZs) in an Amazon S3 Region before returning SUCCESS. For S3 One Zone-IA, objects are stored in 1 zone. Nevertheless, follow best practices such as **secure access permissions**, Cross-Region Replication, versioning, and a functioning, regularly tested backup

Storage classes:

1. **S3 Standard** for general-purpose storage of frequently accessed data
   High throughput & low latency. 11 9s durable
   99.99% availability

1. **S3 Intelligent-Tiering** for data with unknown or changing access patterns;

2. **S3 Standard-IA and S3 One Zone-IA** long-lived, but less frequently accessed data such as DR
   Same as STD but **99.9 & 99.5**% availability

1. **S3 Glacier and S3 Glacier Deep Archive** for long-term archive and digital preservation.

S3 Storage Classes can be configured at the object level and a single bucket can contain objects stored in Standard, Intelligent-Tiering (IT), IA, and One Zone-IA. You can upload objects directly to IT or use S3 Lifecycle policies to transfer objects from Std. and Std-IA to S3 IT. You can also archive objects from

	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

AWS S3 - Security - Data Protection

Inter-network Network privacy

Traffic Between Service and On-Premises Clients and Applications

You have two connectivity options between your private network and AWS:

- An AWS Site-to-Site VPN connection.
- An AWS Direct Connect connection.

1. Clients must support TLS 1.0. We recommend TLS 1.2 or above.
2. Clients must also support cipher suites with Perfect Forward Secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE).
3. Additionally, you must sign requests using an access key ID and a secret access key that are associated with an IAM principal, or you can use the AWS Security Token Service (STS) to generate temporary security credentials to sign requests.

Traffic Between AWS Resources in the Same Region

Use VPC (Gateway) endpoint for S3 that allows private secure connectivity from VPC to Amazon S3

AWS S3 - Security - Data Protection

Encryption

Data protection refers to protecting data while in-transit (as it travels to and from Amazon S3) and at rest (while it is stored on disks in Amazon S3 data centers). You can protect data in transit using Secure Sockets Layer (SSL) or client-side encryption. You have the following options for protecting data at rest in Amazon S3:

- **Server-Side Encryption** (securing DATA AT REST) – Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.
- **Client-Side Encryption** (securing DATA IN TRANSIT)– Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.

SSE - Three mutually exclusive options depending on how you choose to manage the encryption keys:

- Use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)
- Use Server-Side Encryption with Keys Stored in AWS KMS (SSE-KMS)
- Use Server-Side Encryption with Customer-Provided Keys (SSE-C)

***CSE* - To enable client-side encryption, you have the following options:**

- Use a master key stored in AWS KMS. (details below ...)
- Use a master key you store within your application.

CSE - Option 1: Using a Master Key stored in AWS KMS

- **When uploading an object**–Using the Customer Master Key (CMK) ID, the client first sends a request to the AWS Key Management Service (AWS KMS) for a key that it can use to encrypt your object data. AWS KMS returns two versions of a randomly generated data encryption key:
  - A plaintext version that the client uses to encrypt the object data
  - A cipher blob of the same data encryption key that the client uploads to Amazon S3 as object metadata
- **When downloading an object**–The client downloads the encrypted object from Amazon S3 along with the cipher blob version of the data encryption key stored as object metadata. The client then sends the cipher blob to AWS KMS to get the plaintext version of the key so that it can decrypt the object data.

AWS KMS(SSE-KMS) vs S3 vs CMK

Amazon S3-Managed Encryption Keys (SSE-S3)

Server-side encryption protects data at rest. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it rotates regularly. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

If you need server-side encryption for all of the objects that are stored in a bucket, use a bucket policy. For example, the following bucket policy denies permissions to upload an object unless the request includes the **x-amz-server-side-encryption** header to request server-side encryption:

```
{
  "Version": "2012-10-17",
  "Id": "PutObjPolicy",
  "Statement": [
    {
      "Sid": "DenyIncorrectEncryptionHeader",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption":
"AES256"
        }
      }
    },
    {
      "Sid": "DenyUnEncryptedObjectUploads"
```

Protecting Data Using SSE with CMKs Stored in AWS Key Management Service (SSE-KMS)

S3 uses AWS KMS customer master keys (CMKs) to encrypt your Amazon S3 objects. SSE-KMS encrypts only the object data. Any object metadata is not encrypted. If you use customer managed CMKs, you use AWS KMS via the Console or AWS KMS APIs to centrally create encryption keys, define the policies that control how keys can be used, and audit key usage to prove that they are being used correctly. You can use these keys to protect your data in Amazon S3 buckets.

For example, it lets you create, rotate, disable, and define access controls and audit the encryption keys that are used to protect your data.

```
{
   "Version":"2012-10-17",
   "Id":"PutObjPolicy",
   "Statement":[{
         "Sid":"DenyUnEncryptedObjectUploads",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::YourBucket/*",
         "Condition":{
            "StringNotEquals":{
"s3:x-amz-server-side-encryption":"aws:kms"
            }
         }
      }
   ]
}
```

Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C)

Using server-side encryption with customer-provided encryption keys (SSE-C) allows you to **set your own encryption keys**. With the encryption key you provide as part of your request, S3 manages both the encryption, as it writes to disks, and decryption, when you access your objects. Therefore, you don't need to maintain any code to perform data encryption and decryption. The only thing you do is manage the encryption keys you provide.

When you upload an object, Amazon S3 uses the encryption key you provide to apply AES-256 encryption to your data and removes the encryption key from memory.

Important
S3 does not store the encryption key you provide. Instead, it stores a randomly salted HMAC value of the encryption key to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means if you lose the encryption key, you lose the object.

Headers to pass from application
x-amz-server-side-encryption-customer-algorithm
x-amz-server-side-encryption-customer-key
x-amz-server-side-encryption-customer-key-MD5

Protecting Data Using Server-Side Encryption with keys stored in AWS KMS(SSE-KMS)

1. **Server-side encryption: SSE-S3**
   - KMS generates this **data key** and encrypts it using the **master key that you** specified earlier;
   - KMS then returns this **encrypted data key** along with the **plaintext data key** to Amazon S3.
   - Amazon S3 encrypts the object using the **plaintext data key first ,** and then stores the now encrypted object (along with the encrypted object key) and deletes the plaintext object key from memory.
   - To retrieve this encrypted object, Amazon S3 sends the encrypted data key to AWS KMS.
   - AWS KMS decrypts the data key using the correct master key and returns the decrypted (plaintext) object key to S3.
   - With the plaintext object key, S3 decrypts the encrypted object and returns it to you.
   - Each object is encrypted with a unique data key.
     - This key is encrypted with a periodically rotated key managed by AWS S3.
     - Amazon S3 server-side encryption uses 256-bit Advanced Encryption Standard (AES) key for both object and master keys.
- This feature is offered at no additional cost beyond what you pay for using Amazon S3.

3. **Server-side encryption using KMS:**
   - You can encrypt the data in Amazon S3 by defining an AWS KMS master key within your account that you want to use to encrypt the unique object (data) key that will ultimately encrypt your object (data).
   - When you upload your object, a request is sent to KMS to create an object key.
   - The first time you add an SSE-KMS–encrypted object to a bucket in a region, a default CMK is created for you automatically.
     - This key is used for SSE-KMS encryption unless you select a CMK that you created separately using AWS Key Management Service.
     - Creating your own CMK gives you more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect your data.

2. **Server-side encryption using customer provided keys: (SSE-C)**
   - Clients can use their own encryption key while uploading an object to Amazon S3.
   - This encryption key is used by Amazon S3 to encrypt your data using AES-256.
   - S3 does not store the key, after the object is encrypted, the encryption key supplied by the client is deleted from the Amazon S3 system that used it to protect the client's data.
   - When the client retrieves this object from Amazon S3, they must provide the same encryption key in the request.
     - Amazon S3 verifies that the encryption key matches,
     - Decrypts the object, and returns the object to the requester.

Server-side encryption SSE-KMS

- Amazon S3 supports bucket policies that you can use if you require server-side encryption for all objects that are stored in your bucket.
- For SSE to be requested in an API call, the request has to include the **x-amz-server-side-encryption** header requesting server-side encryption
- If you want SSE-KMS then the **x-amz-server-side-encryption** header has to define SEE-KMS.

```
"s3:x-amz-server-side-encryption":"aws:kms"
```

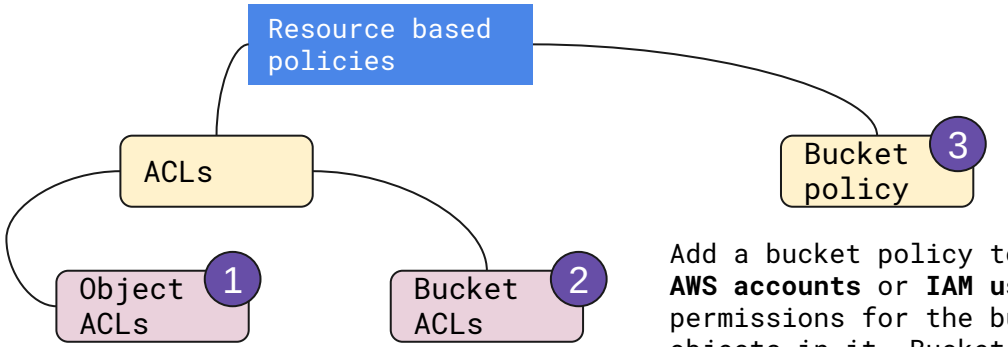
SSE-S3 is AWS managed (b/w S3 and KMS, No Appl)
SSE-KMS is Customer master keys (client sends header)
SSE-C is without KMS (No KMS, client sends key in request)

AWS S3 - Security - IAM

By default, all S3 resources—buckets, objects, and related subresources are private: only the resource owner, an AWS account that created it, can access the resource. The resource owner can **optionally grant access permissions** to others by **writing an access policy**.

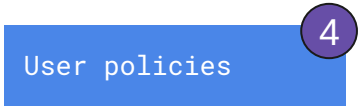
1. **Resource-based policies** - Access policies you attach to your resources (buckets and objects). E.g. bucket policies and access control lists (ACLs)
2. **User policies** - Attach access policies to users in your account.

You may choose to use either or some combination of these to manage permissions to your S3 resources



Each bucket and object has an ACL associated with it. An ACL is a **list of grants identifying grantee and permission granted**.

Add a bucket policy to **grant other AWS accounts** or **IAM users** permissions for the bucket and the objects in it. Bucket policies supplement, and in many cases, replace ACL-based access policies.



Create IAM users, groups, and roles in your account and attach access policies to them granting them access to S3

AWS S3 - Security - IAM

When to Use an Object ACL: 1

1. An object ACL is the only way to manage access to objects not owned by the bucket owner
   An AWS account that owns the bucket can grant another AWS account permission to upload objects. The bucket owner does not own these objects. The AWS account that created the object must grant permissions using object ACLs.
1. Permissions vary by object and you need to manage permissions at the object level
2. Object ACLs control only object-level permissions – There is a single bucket policy for the entire bucket, but object ACLs are specified per object.

When to Use a Bucket ACL: 2

The only recommended use case for the bucket ACL is to grant write permission to the Amazon S3 Log Delivery group to write access log objects to your bucket

When to Use a Bucket Policy: 3

You want to manage cross-account permissions for all Amazon S3 permissions – You can use ACLs to grant cross-account permissions to other accounts, but ACLs support only a finite set of permission

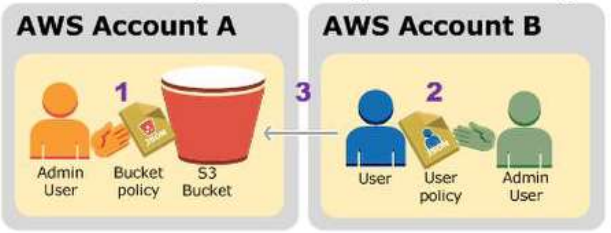
When to Use a User Policy: 4

In general, you can use either a user policy or a bucket policy to manage permissions. You may choose to manage permissions by creating users and managing permissions individually by attaching policies to users (or user groups), or you may find that resource-based policies, such as a bucket policy, work better for your scenario

AWS S3 - Security - IAM - Examples

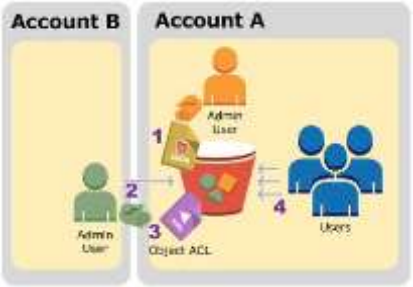
Example 1: Bucket Owner Granting Its Users Bucket Permissions

AWS account can use a bucket policy, a user policy, or both to grant its user permissions on the bucket. You will grant some permissions using a bucket policy and grant other permissions using a user policy.



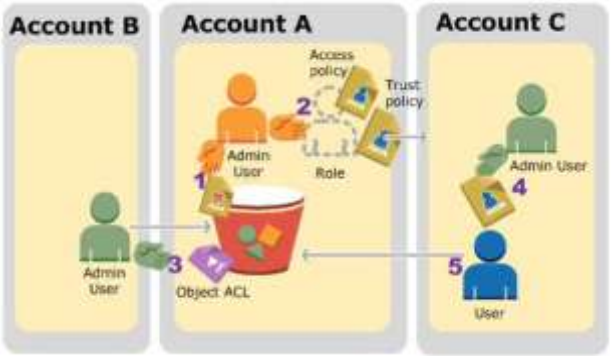
Example 2: Bucket Owner Granting Cross-Account Bucket Permissions

1. Account A admin user attaches a bucket policy granting cross-account permissions to Account B to perform specific bucket operations.
2. Account B admin user attaches user policy to the user delegating the permissions it received from Account A.
3. User in Account B can access an object in the bucket owned by Account A



Example 3: Bucket Owner Granting Its Users Permissions to Objects It Does Not Own

1. Account A admin user attaches a bucket policy with two statements.
    a. Allow cross-account permission to Account B to upload objects.
    b. Allow a user in its own account to access objects in the bucket.
2. Account B admin user uploads objects to the bucket owned by Account A. Account B admin updates the object ACL adding grant that gives the bucket owner full-control permission on the object.
3. User in Account A can access objects, regardless of who owns them.



Example 4: Bucket Owner Granting Cross-account Permission to Objects It Does Not Own

1. Account A administrator user attaches a bucket policy granting Account B conditional permission to upload objects.
2. Account A administrator creates an IAM role, establishing trust with Account C, so users in that account can access Account A. The access policy attached to the role limits what user in Account C can do when the user accesses Account A.
3. Account B administrator uploads an object to the bucket owned by Account A, granting full-control permission to the bucket owner.
4. Account C administrator creates a user and attaches a user policy that allows the user to assume the role.
5. User in Account C first assumes the role, which returns the user temporary security

AWS S3 - VPC Endpoint

A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Two types of VPC endpoints: *interface endpoints* and *gateway endpoints* *(supports S3 and DynamoDB)*.

Legacy Issue for S3 Access:

Until now, if you wanted your EC2 instances to be able to access public resources, you had to use an Internet Gateway, and potentially manage some NAT instances.

VPC Endpoint access for S3 feature introduced in 2015:

EC2 instances running in private subnets of a VPC can now have controlled access to S3 buckets, objects, and API functions that are in the same region as the VPC. You can use an S3 bucket policy to indicate which VPCs and which VPC Endpoints have access to your S3 buckets.

These endpoints are easy to configure, highly reliable, and provide a secure connection to S3 that does not require a gateway or NAT instances.

Bucket policy restricting Access to a Specific VPC

```
{  "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [    {
        "Sid": "Access-to-specific-VPC-only",
        "Principal": "*",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": ["arn:aws:s3:::examplebucket",
                     "arn:aws:s3:::examplebucket/*"],
        "Condition": {
           "StringNotEquals": {
              "aws:sourceVpc": "vpc-111bbb22"
           }
        }
    } ]
}
```

AWS S3 - Requester Pays bucket

Conditions

- ❏ If you enable Requester Pays on a bucket, anonymous access to that bucket is not allowed.
- ❏ You must authenticate all requests involving Requester Pays buckets.
- ❏ Requesters must include x-amz-request-payer in their requests either in the header, for POST, GET and HEAD requests, or as a parameter in a REST request to show that they understand that they will be charged for the request and the data download.

Charges

❏ The requester instead of the bucket owner pays the cost of the request and the data download from the bucket.

❏ The bucket owner always pays the cost of storing data.

However, the bucket owner is charged for the request under the following conditions:

- The requester doesn't include the parameter x-amz-request-payer in the header (GET, HEAD, or POST) or as a parameter (REST) in the request (HTTP code 403).
- Request authentication fails (HTTP code 403).
- The request is anonymous (HTTP code 403).
- The request is a SOAP request.

Requester Pays buckets do not support the following.

- Anonymous requests
- BitTorrent
- SOAP requests
- You cannot use a Requester Pays bucket as the target bucket for end user logging, or vice versa;

AWS S3 - Cross region replication

CRR enables automatic, asynchronous copying of objects across buckets in different AWS Regions.
Buckets configured for cross-region replication can be owned by the same AWS account or by different accounts.

When to use CRR:

- Comply with compliance requirements
- Minimize latency
- Increase operational efficiency
  distributed compute
- Maintain object copies under different ownership
  Restrict access to replicas

What is replicated?

1. Objects created after you add a replication configuration
2. Both unencrypted objects and objects encrypted using SSE-S3 or KMS managed keys (SSE-KMS need explicit enabling)
3. Object metadata.
4. Only objects in the source bucket for which the bucket owner has permissions to read objects and ACLs
5. Object ACL updates
6. Object tags
7. Amazon S3 object lock retention information, if there is any

What isn't replicated?

1. Objects that existed before you added the replication configuration
2. Objects created with server-side encryption using customer-provided (SSE-C) encryption keys OR Objects created with server-side encryption using AWS KMS–managed encryption (SSE-KMS) keys.
3. Objects in the source bucket that the bucket owner doesn't have permissions for
4. Updates to bucket-level subresources.
5. Actions performed by lifecycle configuration.
6. Objects in the source bucket that are replicas that were created by another cross-region replication.

Example 1: Configure CRR When Source and Destination Buckets Are Owned by the Same AWS Account

Example 2: Configure CRR When Source and Destination Buckets Are Owned by Different AWS Accounts

Example 3: Change Replica Owner When Source and Destination Buckets Are Owned by Different AWS Accounts

Example 4: Replicating Encrypted Objects

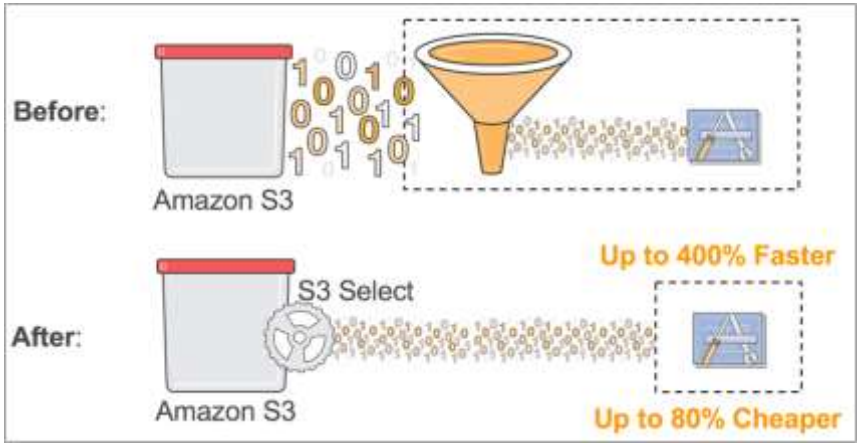
S3 Select and Glacier Select – Retrieving Subsets of Objects

New feature introduced in 2017

Enables applications to **retrieve only a subset of data** from an object by **using simple SQL expressions**. Drastic performance increases – in many cases you can get as much as a **400% improvement**. **Athena**, **Redshift**, and **EMR** as well as partners like Cloudera, DataBricks and Hortonworks will all support S3 Select.

Query pushdown using S3 Select is now supported with **Spark**, **Hive** and **Presto** in EMR. You can use this feature to push down the computational work of filtering from EMR to S3, which improves performance and reduce amount of data transferred b/w EMR and S3.

Glacier Select allows you to to perform filtering directly against a Glacier object using standard SQL statements. Glacier Select works just like any other retrieval job except it has an additional set of parameters you can pass in initiate job request.



S3 Transfer Acceleration

Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration takes advantage of CloudFront's globally distributed edge locations. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path. When using Transfer Acceleration, additional data transfer charges may apply.

You might want to use Transfer Acceleration on a bucket for various reasons, including the following:

- You have customers that upload to a centralized bucket from all over the world.
- You transfer gigabytes to terabytes of data on a regular basis across continents.
- You are unable to utilize all of your available bandwidth over the Internet when uploading to S3.

S3 Batch operations (New in Apr 2019)

Use this new feature to **easily process hundreds, millions, or billions of S3 objects** in a simple and straightforward fashion. You can copy objects to another bucket, set tags or access control lists (ACLs), initiate a restore from Glacier, or invoke an AWS Lambda function on each one.

This feature builds on S3's existing **support for inventory reports** and can use the reports or CSV files to drive your batch operations. You don't have to write code, set up any server fleets, or figure out how to partition the work and distribute it to the fleet. Instead, you create a job in minutes with a couple of clicks, turn it loose, and sit back while S3 uses massive, behind-the-scenes parallelism to take care of the work. You can create, monitor, and manage your batch jobs using the S3 Console, the S3 CLI, or the S3 APIs.

Bucket – An S3 bucket holds a collection of any number of S3 objects, with optional per-object versioning.

Inventory Report – An S3 inventory report is generated each time a daily or weekly bucket inventory is run. A report can be configured to include all of the objects in a bucket, or to focus on a prefix-delimited subset.

Manifest – A list (either an Inventory Report, or a file in CSV format) that identifies the objects to be processed in the batch job.

Batch Action – The desired action on the objects described by a Manifest. Applying an action to an object constitutes an S3 Batch Task.

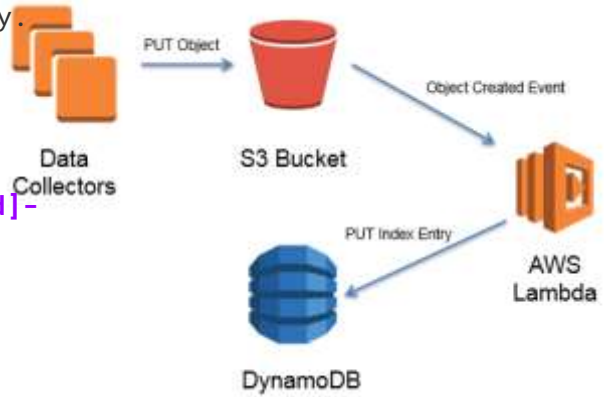
IAM Role – An IAM role that provides S3 with permission to read the objects in the inventory report, perform the desired actions, and to write the optional completion report. If you choose Invoke AWS Lambda function as your action, the function's execution role must grant permission to access the desired AWS services and resources.

Batch Job – References all of the items above. Each job has a status and a priority; higher priority (numerically) jobs take precedence over those with lower priority.

Building and Maintaining an Amazon S3 Metadata Index without Servers

https://aws.amazon.com/blogs/big-data/building-and-maintaining-an-amazon-s3-metadata-index-without-servers/

- Build an external index that maps queryable attributes to the S3 object key.
- This index can leverage data repositories built for fast lookups
- Leverage an index instead of listing keys directly, to dramatically reduce the search space and improve performance.



Assume that the servers upload objects with the following key structure:
[4-digit hash]/[server id]/[year]-[month]-[day]-[hour]-[minute]/[customer id]-[epoch timestamp].data
Example: a5b2/i-31cc02/2015-07-05-00-25/87423-1436055953839.data

This key structure enables sustained, high-access rates to S3 but makes it difficult to find all keys for a given customer or server using S3 LIST operations. For instance, to list all the data objects for a given customer uploaded within the last 24 hours, you would have to iterate over every single key in the bucket and inspect the customer ID for each one separately. Other use cases:

1. Find all objects for a given customer collected during a time range.
2. Calculate the total storage used for a given customer.
3. List all objects for a given customer that contain a transaction record.
4. Find all objects uploaded by a given server during a time range.

Architectural goals → Zero administration cost, Scalable and elastic, Automatic

Multipart Uploads

https://docs.aws.amazon.com/AmazonS3/latest/dev/mpuoverview.html

Item	Specification
Maximum object size	5 TB
Maximum number of parts /upload	10,000
Part numbers	1 to 10,000
Part size	5 MB to 5 GB
Max# of parts returned for a list parts request	1000
Max# of multipart uploads returned in a list multipart uploads request	1000

The Multipart upload API enables you to upload large objects in parts. You can use this API to upload new large objects or make a copy of an existing object

MP-uploading is a 3-step process:
- You initiate the upload,
- you upload the object parts, and after you have uploaded all the parts,
- you complete the multipart upload.

Upon receiving the complete multipart upload request, Amazon S3 constructs the object from the uploaded parts, and you can then access the object just as you would any other object in your bucket.

You can list all of your in-progress multipart uploads or get a list of the parts that you have uploaded for a specific multipart upload.

```
POST /ObjectName?uploads HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string

PUT /ObjectName?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Content-Length: Size
Authorization: authorization string

POST /ObjectName?uploadId=UploadId HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: Date
Content-Length: Size
Authorization: authorization string

<CompleteMultipartUpload>
 <Part>
  <PartNumber>PartNumber</PartNumber>
  <ETag>ETag</ETag>
 </Part>
 ...
</CompleteMultipartUpload>
```

Versioning

https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures.

Versioning-enabled buckets enable you to recover objects from accidental deletion or overwrite. eg:

- If you delete an object, instead of removing it permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can always restore the previous version.
- If you overwrite an object, it results in a new object version in the bucket. You can always restore the previous version.

Buckets can be in one of three states: unversioned (the default), versioning-enabled, or versioning-suspended. Important: Once you version-enable a bucket, it can never return to an unversioned state. You can, however, suspend versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket. The first time you enable a bucket for versioning, objects in it are thereafter always versioned and given a unique version ID. Note the following:

- Objects stored in your bucket before you set the versioning state have a version ID of null. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests.
- The bucket owner (or any user with appropriate permissions) can suspend versioning to stop accruing object versions. When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests.

Pre-signed URLs

https://medium.com/@aidan.hallett/securing-aws-s3-uploads-using-presigned-urls-aa821c13ae8d

By default, all objects are private – meaning only the bucket account owner initially has access to the object.

If you want a user to have access to a specific bucket or objects without making them public, you can provide the user with the appropriate permissions using an IAM policy.

Instead of above, you can also create a presigned URL - meaning users can interact with objects without the need for AWS credentials or IAM permissions.

A presigned URL is a URL that you can provide to your users to grant temporary access to a specific S3 object. Using the URL, a user can either READ the object or WRITE an Object (or update an existing object). The URL contains specific parameters which are set by your application. A pre-signed URL uses three parameters to limit the access to the user;

- Bucket: The bucket that the object is in (or will be in)
- Key: The name of the object
- Expires: The amount of time that the URL is valid

As expected, once the expiry time has lapsed the user is unable to interact with the specified object. AWS gives access to the object through the presigned URL as the URL can only be correctly signed by the S3 Bucket owner.

You can generate a presigned URL programmatically using the AWS SDK for Java or the AWS SDK for .NET. If you are using Microsoft Visual Studio, you can also use AWS Explorer to generate a presigned object URL without writing any code. Anyone who receives a valid presigned URL can then programmatically upload an object.

```
https://presignedurldemo.s3.eu-west-2.amazonaws.com/image.png?
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIAJJWZ7B6WCRGMKFGQ%2F20180210%2Feu-west-2%2Fs3%2Faws4_request&
X-Amz-Date=20180210T171315Z&
X-Amz-Expires=1800&
X-Amz-Signature=12b74b0788aa036bc7c3d03b3f20c61f1f91cc9ad8873e3314255dc479a25351&
X-Amz-SignedHeaders=host
```

CloudFront Signed URLs

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-urls.html

A signed URL includes additional information, for example, an expiration date and time, that gives you more control over access to your content. This additional information appears in a policy statement, which is based on either a canned policy or a custom policy.

How Signed URLs Work

1. In your CloudFront distribution, specify one or more trusted signers, which are the AWS accounts that you want to have permission to create signed URLs.
2. You develop your application to determine whether a user should have access to your content and to create signed URLs for the files or parts of your application that you want to restrict access to.
3. A user requests a file for which you want to require signed URLs.
4. Your application verifies that the user is entitled to access the file: they've signed in, they've paid for access to the content, or they've met some other requirement for access.
5. Your application creates and returns a signed URL to the user.
6. The signed URL allows the user to download or stream the content.
   This step is automatic; the user usually doesn't have to do anything additional to access the content. For example, if a user is accessing your content in a web browser, your application returns the signed URL to the browser. The browser immediately uses the signed URL to access the file in the CloudFront edge cache without any intervention from the user.
7. CloudFront uses the public key to validate the signature and confirm that the URL hasn't been tampered with. If the signature is invalid, the request is rejected.
   If the signature is valid, CloudFront looks at the policy statement in the URL (or constructs one if you're using a canned policy) to confirm that the request is still valid. For example, if you specified a beginning and ending date and time for the URL, CloudFront confirms that the user is trying to access your content during the time period that you want to allow access.
   If the request meets the requirements in the policy statement, CloudFront does the standard operations: determines whether the file is already in the edge cache, forwards the request to the origin if necessary, and returns the file to the user.

CloudFront Signed Cookies

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-cookies.html

CloudFront signed cookies allow you to control who can access your content when you don't want to change your current URLs or when you want to provide access to multiple restricted files, for example, all of the files in the subscribers' area of a website.

How Signed Cookies Work

1. In your CloudFront distribution, you specify one or more trusted signers, which are the AWS accounts that you want to have permission to create signed URLs and signed cookies.
2. You develop your application to determine whether a user should have access to your content and, if so, to send three Set-Cookie headers to the viewer. (Each Set-Cookie header can contain only one name-value pair, and a CloudFront signed cookie requires three name-value pairs.) You must send the Set-Cookie headers to the viewer before the viewer requests your private content. If you set a short expiration time on the cookie, you might also want to send three more Set-Cookie headers in response to subsequent requests, so that the user continues to have access.
   Typically, your CloudFront distribution will have at least two cache behaviors, one that doesn't require authentication and one that does. The error page for the secure portion of the site includes a redirector or a link to a login page.
   If you configure your distribution to cache files based on cookies, CloudFront doesn't cache separate files based on the attributes in signed cookies.
3. A user signs in to your website and either pays for content or meets some other requirement for access.
4. Your application returns the Set-Cookie headers in the response, and the viewer stores the name-value pairs.
5. The user requests a file.
   The user's browser or other viewer gets the name-value pairs from step 4 and adds them to the request in a Cookie header. This is the signed cookie.
6. CloudFront uses the public key to validate the signature in the signed cookie and to confirm that the cookie hasn't been tampered with. If the signature is invalid, the request is rejected.
   If the signature in the cookie is valid, CloudFront looks at the policy statement in the cookie (or constructs one if you're using a canned policy) to confirm that the request is still valid. For example, if you specified a beginning and ending date and time for the cookie, CloudFront confirms that the user is trying to access your content during the time period that you want to allow access.
   If the request meets the requirements in the policy statement, CloudFront serves your content as it does for content that isn't restricted: it determines whether the file is already in the edge cache, forwards the request to the origin if

S3 Pre-signed URLs vs CloudFront Signed URLs vs Origin Access Identity (OAI)

https://tutorialsdojo.com/aws-cheat-sheet-s3-pre-signed-urls-vs-cloudfront-signed-urls-vs-origin-access-identity-oai/

S3 Pre-signed URLs	CloudFront Signed URLs	Origin Access Identity (OAI)

• All S3 buckets and objects by default are **private**. Only the object owner has permission to access these objects. Pre-signed URLs use the owner's security credentials to grant others time-limited permission to download or upload objects.
• When creating a pre-signed URL, you (as the owner) need to provide the following:
  ◦ Your security credentials
  ◦ An S3 bucket name
  ◦ An object key
  ◦ Specify the HTTP method (GET to download the object or PUT to upload an object)
  ◦ Expiration date and time of the URL.
	• You can control user access to your private content in two ways
	  ◦ Restrict access to files in CloudFront edge caches
	  ◦ Restrict access to files in your Amazon S3 bucket (unless you've configured it as a website endpoint)
	• You can configure CloudFront to require that users access your files using either **signed URLs** or **signed cookies**. You then develop your application either to create and distribute signed URLs to authenticated users or to send Set-Cookie headers that set signed cookies on the viewers for authenticated users.
	• When you create signed URLs or signed cookies to control access to your files, you can specify the following restrictions:
	  ◦ An expiration date and time for the URL
	  ◦ (Optional) The date and time the URL becomes valid
	  ◦ (Optional) The IP address or range of addresses of the computers that can be used to access your content
	• You can use signed URLs or signed cookies for any CloudFront distribution, regardless of whether the origin is an Amazon S3 bucket or an HTTP server.
		• You can configure an S3 bucket as the origin of a CloudFront distribution. OAI prevents users from viewing your S3 files by simply using the direct URL for the file. Instead, they would need to access it through a CloudFront URL.
		• To require that users access your content through CloudFront URLs, you perform the following tasks:
		  ◦ Create a special CloudFront user called an **origin access identity**.
		  ◦ Give the origin access identity permission to read the files in your bucket.
		  ◦ Remove permission for anyone else to use Amazon S3 URLs to read the files (through bucket policies or ACLs).
		• You cannot set OAI if your S3 bucket is configured as a website endpoint.

Cross-Origin Resource Sharing (CORS)

https://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

How Do I Configure CORS on My Bucket?

To configure your bucket to allow cross-origin requests, you create a CORS configuration, which is an XML document with rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that will support for each origin, and other operation-specific information.

- Scenario 1: Suppose that you are hosting a website in an S3 bucket named website. Your users load the website endpoint **http://website.s3-website-us-east-1.amazonaws.com**. Now you want to use JavaScript on the webpages that are stored in this bucket to be able to make authenticated GET and PUT requests against the same bucket by using the Amazon S3 API endpoint for the bucket, **website.s3.amazonaws.com**. A browser would normally block JavaScript from allowing those requests, but with CORS you can configure your bucket to explicitly enable cross-origin requests from website.s3-website-us-east-1.amazonaws.com.

- Scenario 2: Suppose that you want to host a web font from your S3 bucket. Again, browsers require a CORS check (also called a preflight check) for loading web fonts. You would configure the bucket that is hosting the web font to allow any origin to make these requests.

```
<CORSConfiguration>
 <CORSRule>
  <AllowedOrigin>http://www.example.com</AllowedOrigin>
  <AllowedMethod>PUT</AllowedMethod>
  <AllowedMethod>POST</AllowedMethod>
  <AllowedMethod>DELETE</AllowedMethod>
  <AllowedHeader>*</AllowedHeader>
 <MaxAgeSeconds>3000</MaxAgeSeconds>
 <ExposeHeader>x-amz-server-side-encryption</ExposeHeader>
 <ExposeHeader>x-amz-request-id</ExposeHeader>
 <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

```
<CORSConfiguration>
 <CORSRule>
  <AllowedOrigin>http://www.example1.com</AllowedOrigin>
  <AllowedMethod>PUT</AllowedMethod>
  <AllowedMethod>POST</AllowedMethod>
  <AllowedMethod>DELETE</AllowedMethod>
  <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
  <AllowedOrigin>http://www.example2.com</AllowedOrigin>
  <AllowedMethod>PUT</AllowedMethod>
  <AllowedMethod>POST</AllowedMethod>
  <AllowedMethod>DELETE</AllowedMethod>
  <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
  <AllowedOrigin>*</AllowedOrigin>
  <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

EBS

EBS - Intro

EBS allows you to create storage volumes and attach them to EC2 instances. Once attached, you can create a file system on top of these volumes, run a database, or use them in any other way you would use block storage. EBS volumes are placed in a specific Availability Zone where they are automatically replicated to protect you from the failure of a single component. All EBS volume types offer durable snapshot capabilities and are designed for 99.999% availability. EBS provides a range of options that allow you to optimize storage performance and cost for your workload. These options are divided into two major categories:

	SSD (Best for workloads with small, random I/O operations)		HDD (Best for workloads with large, sequential I/O operations)	
Volume Type	General Purpose SSD (gp2)	Provisioned IOPS SSD (io1)	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Description	General purpose SSD volume that balances price and performance for a wide variety of workloads	Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads	Low-cost HDD volume designed for frequently accessed, throughput-intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
Use Cases	• Recommended for most workloads • System boot volumes • Virtual desktops • Low-latency interactive apps • Development and test environments	• Critical business applications that require sustained IOPS performance, or more than 16,000 IOPS or 250 MiB/s of throughput per volume • Large database workloads, such as: MongoDB, cassandra, Microsoft SQL Server, MySQL, PostgreSQL, Oracle	• Streaming workloads requiring consistent, fast throughput at a low price • Big data • Data warehouses • Log processing • Cannot be a boot volume	• Throughput-oriented storage for large volumes of data that is infrequently accessed • Scenarios where the lowest storage cost is important • Cannot be a boot volume
API Name	gp2	io1	st1	sc1
Volume Size	1 GiB - 16 TiB	4 GiB - 16 TiB	500 GiB - 16 TiB	500 GiB - 16 TiB
Max IOPS per Volume	16,000 (16 KiB I/O) *	64,000 (16 KiB I/O) †	500 (1 MiB I/O)	250 (1 MiB I/O)
Max Throughput per Volume	250 MiB/s *	1,000 MiB/s †	500 MiB/s	250 MiB/s
Max IOPS per Instance ††	80,000	80,000	80,000	80,000
Max Throughput per Instance ††	1,750 MiB/s	1,750 MiB/s	1,750 MiB/s	1,750 MiB/s
Dominant Performance Attribute	IOPS	IOPS	MiB/s	MiB/s

EBS Snapshots

EBS provides the ability to **save point-in-time snapshots of your volumes to S3**.
EBS Snapshots are stored incrementally. Snapshots can be used to instantiate multiple new volumes, expand the size of a volume, or move volumes across Availability Zones.
When a new volume is created, you may choose to create it based on an existing EBS snapshot.

- **Immediate access to EBS volume data** - After a volume is created from a snapshot, there is no need to wait for all of the data to transfer from Amazon S3 to your EBS volume before your attached instance can start accessing the volume. EBS Snapshots implement lazy loading, so that you can begin using them right away.
- **Resizing EBS volumes** - There are two methods that can be used to resize an EBS volume. If you create a new volume based on a snapshot, you can specify a larger size for the new volume. With the Elastic Volumes feature you can dynamically grow live volumes without the use of snapshots. Make certain that your file system and application supports resizing a device.
- **Sharing EBS Snapshots** - EBS Snapshots' shareability makes it easy for you to share data with your co-workers or others in the AWS community. Authorized users can create their own EBS volumes based on your EBS shared snapshots; your original snapshot remains intact. If you choose, you can also make your data available publicly to all AWS users.
- **Copying EBS Snapshots across AWS regions** - EBS's ability to copy snapshots across AWS regions makes it easier to leverage multiple AWS regions for geographical expansion, data center migration and disaster recovery. You can copy any snapshot accessible to you: snapshots you created; snapshots shared with you; and snapshots from the AWS Marketplace, VM Import/Export, and AWS Storage Gateway.

For an additional low, hourly fee, customers can launch certain EC2 instance types as **EBS-optimized instances**. EBS-optimized instances enable EC2 instances to fully use the IOPS provisioned on an EBS volume. EBS-optimized instances deliver dedicated throughput between EC2 and EBS, with options between 500 and 10,000 Megabits per second (Mbps) depending on the instance type used. The dedicated throughput minimizes contention between EBS I/O and other traffic from your EC2 instance, providing the best performance for your EBS volumes.

Automating the Amazon EBS Snapshot Lifecycle

You can use Amazon Data Lifecycle Manager (Amazon DLM) to automate the creation, retention, and deletion of snapshots taken to back up your Amazon EBS volumes. Automating snapshot management helps you to:

- Protect valuable data by enforcing a regular backup schedule.
- Retain backups as required by auditors or internal compliance.
- Reduce storage costs by deleting outdated backups.

Combined with the monitoring features of Amazon CloudWatch Events and AWS CloudTrail, Amazon DLM provides a complete backup solution for EBS volumes at no additional cost.

RAID Configuration on Linux

Configuration	Use	Advantages	Disadvantages
RAID 0 **(FAST I/O, Less fault tolerant**)	When I/O performance is more important than fault tolerance; for example, as in a heavily used database (where data replication is already set up separately).	I/O is distributed across the volumes in a stripe. If you add a volume, you get the straight addition of throughput.	Performance of the stripe is limited to the worst performing volume in the set. Loss of a single volume results in a complete data loss for the array.
RAID 1 **(SLOW I/O, Is fault tolerant**)	When fault tolerance is more important than I/O performance; for example, as in a critical application.	Safer from the standpoint of data durability.	Does not provide a write performance improvement; requires more Amazon EC2 to Amazon EBS bandwidth than non-RAID configurations because the data is written to multiple volumes simultaneously.

RAID 5 and RAID 6 are not recommended for Amazon EBS

Storage Gateway

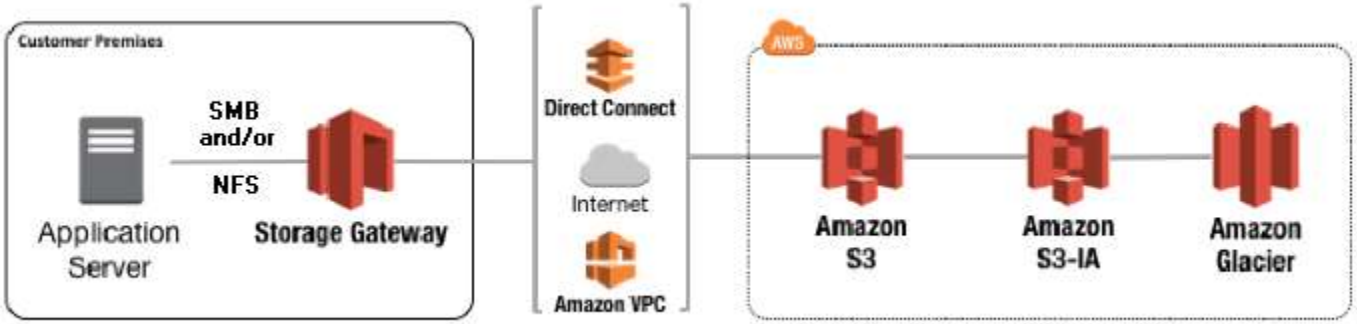
Storage Gateway - Intro

Run AWS Storage Gateway either on-premises as a VM appliance, as a hardware appliance, or in AWS as an EC2 instance.

AWS Storage Gateway connects an on-premises software appliance with cloud-based storage. Use the service to store data in the AWS for scalable and cost-effective storage that helps maintain data security. AWS Storage Gateway offers

1. **File-based** - A file gateway supports a file interface into S3 and combines a service and a virtual software appliance. By using this combination, you can store and retrieve objects in S3 using industry-standard file protocols such as NFS and SMB
2. **Volume-based** - A volume gateway provides cloud-backed storage volumes that you can mount as Internet Small Computer System Interface (iSCSI) devices from your on-premises application servers.
    a. **Cached volumes** – You store your data in Amazon S3 and retain a copy of frequently accessed data subsets locally.
    b. **Stored volumes** – If you need low-latency access to your entire dataset, first configure your on-premises gateway to store all your data locally. Then asynchronously back up point-in-time snapshots of this data to S3.
3. **Tape-based -** cost-effectively and durably archive backup data in GLACIER or DEEP_ARCHIVE

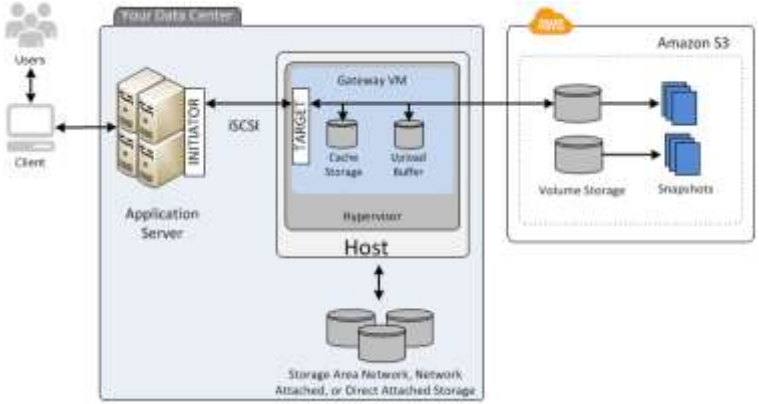
Storage Gateway - Architecture - File Gateways

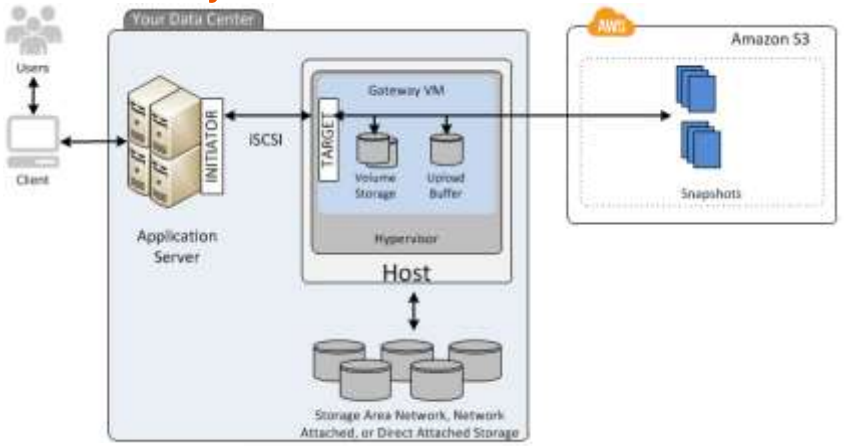


File Gateways:

- Start by downloading a VM image for the file gateway. Activate the file gateway from the AWS Console or through the Storage Gateway API.
- You can also create a file gateway using an EC2 image.
- After the file gateway is activated, you create and configure your file share and associate that share with your S3 bucket.
- Doing this makes the share accessible by clients using either the NFS or SMB protocol.
- Files written to a file share become objects in S3, with the path as the key.
- There is a one-to-one mapping between files and objects, and the gateway asynchronously updates the objects in S3 as you change the files.
- Existing objects in the bucket appear as files in the file system, and the key becomes the path. Objects are encrypted with SSE-S3.
- All data transfer is done through HTTPS.

Storage Gateway - Architecture - Volume Gateways





Cached Volumes:
Allocate disks on-premises for the VM. These on-premises disks serve the following purposes:

- **Disks for use by the gateway as cache storage** – As your applications write data to the storage volumes in AWS, the gateway first stores the data on the on-premises disks used for cache storage. Then the gateway uploads the data to Amazon S3.
- **Disks for use by the gateway as the upload buffer** – To prepare for upload to Amazon S3, your gateway also stores incoming data in a staging area, referred to as an upload buffer.

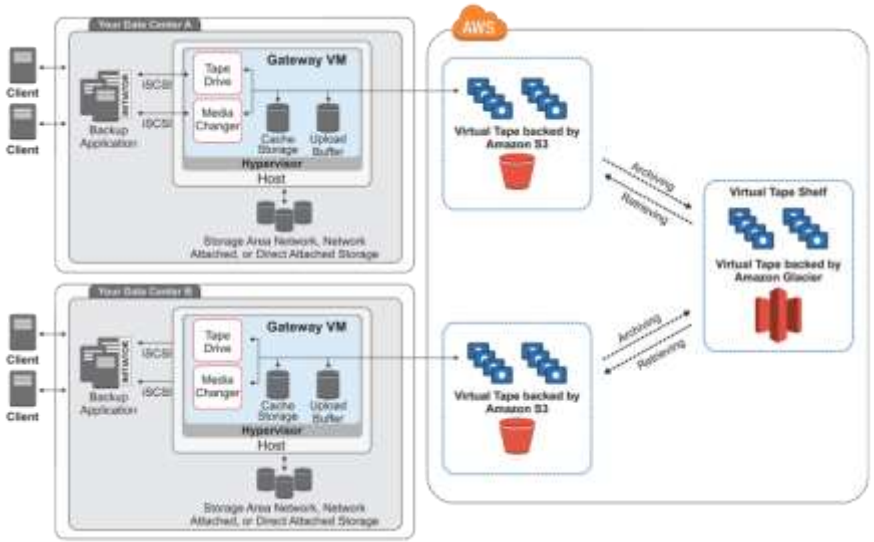
Stored Volumes:
After you install the AWS Storage Gateway software and activated it, you can create gateway storage volumes. You then map them to on-premises DAS or SAN disks. Mount these storage volumes to your on-premises application servers as iSCSI devices.

You can take point-in-time snapshots of gateway volumes that are made available in the form of Amazon EBS snapshots,

- These snapshots can be turned into either Storage Gateway Volumes or EBS Volumes.

To access your iSCSI volumes in AWS, you can take EBS snapshots which can be used to create EBS volumes.

Storage Gateway - Architecture - Tape Gateways



Tape Gateway offers a durable, cost-effective solution to archive your data in the AWS Cloud. With its virtual tape library (VTL) interface, you use your existing tape-based backup infrastructure to store data on virtual tape cartridges that you create on your tape gateway. Each tape gateway is preconfigured with a media changer and tape drives. These are available to your existing client backup applications as iSCSI devices. You add tape cartridges as you need to archive your data.