AWS Storage Gateway

File Gateway for Hybrid Architectures

Overview and Best Practices

April 2017

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Organizations are looking for ways to reduce their physical data center footprints, particularly for secondary file or on-demand workloads. However, bridging data between private data centers and the public cloud comes with a unique set of challenges. Traditional data center services rely on low-latency network attached storage (NAS) and storage area network (SAN) protocols to access storage locally. Cloud-native applications are generally optimized for API access to data in scalable and durable cloud object storage, such as Amazon Simple Storage Service (Amazon S3). This paper outlines the basic architecture and best practices for building hybrid environments using AWS Storage Gateway in a file gateway configuration to address key use cases, such as cloud tiering and hybrid cloud backup.

Introduction

Organizations are looking for ways to reduce their physical data center infrastructure. A great way to start is by moving secondary or tertiary workloads, such as long-term file retention and backup and recovery operations, to the cloud. In addition, organizations want to take advantage of the elasticity of cloud architectures and features to access and use their data in new on-demand ways that a traditional data center infrastructure can't support.

AWS Storage Gateway can provide low-latency Network File System (NFS) access to Amazon Simple Storage Service (Amazon S3) objects from on-premises applications, while offering simultaneous access from any Amazon S3 API-enabled application. The file gateway configuration of AWS Storage Gateway enables hybrid IT architectures in use cases such as tiered file storage, archiving, on-demand bursting of workloads, and backup to the AWS Cloud.

Individual files that are stored in Amazon S3 using the AWS file gateway are stored as independent objects. This provides high durability and low-cost, flexible storage with virtually infinite capacity. Files are written to Amazon S3 in their original format without any proprietary modification. This means that you can easily access data using applications and services that natively integrate with Amazon S3 buckets, such as Amazon EMR or Amazon Athena. It also allows storage management through native Amazon S3 features, such as lifecycle policies, analytics, and Cross-Region Replication (CRR).

A file gateway can communicate efficiently between private data centers and AWS and translate traditional NAS protocols to object storage API calls. This makes it an ideal component in a variety of use cases, including data migration, cloud tiering, and hybrid backup solutions.

AWS File Gateway Architecture

A file gateway provides a simple solution for presenting one or more Amazon S3 buckets and their objects as a mountable NFS to one or more clients on-premises.

The file gateway is deployed in the form of a virtual appliance that can run either in a VMware environment or in an Amazon Elastic Compute Cloud (Amazon EC2) instance in AWS. When the file gateway is deployed in a privately hosted VMware environment, it acts as a performance-optimized connection between NFS (v3.0 or v4.1) client systems in a private data center and Amazon S3 buckets hosted in a given AWS Region. The file gateway uses locally attached storage to provide a read/write cache to reduce latency for NFS clients in the same local area network (LAN) as the file gateway.

A "bucket share" consists of an NFS share hosted from a file gateway across a single Amazon S3 bucket. The file gateway virtual appliance currently supports up to 10 bucket shares.

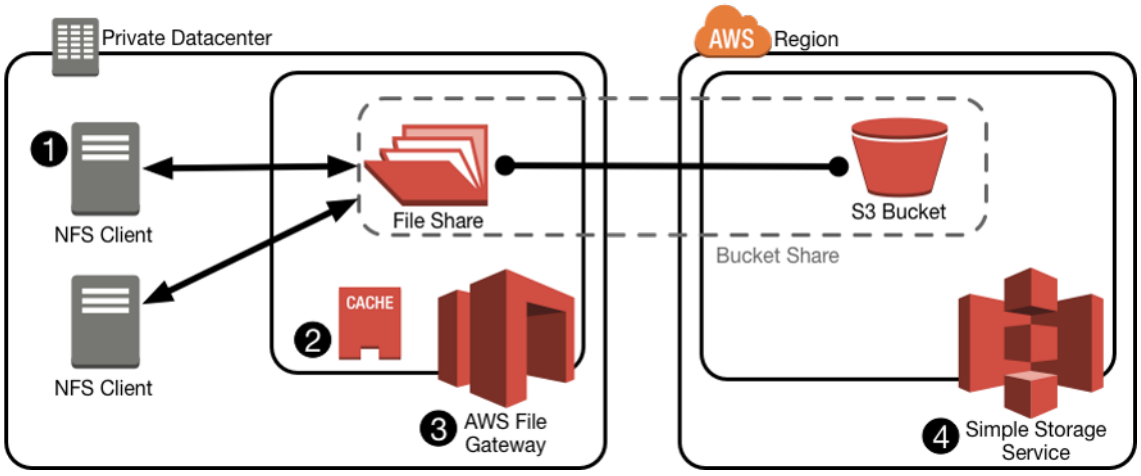

Figure 1: Basic file gateway architecture

Here are the components of the file gateway architecture shown in Figure 1:

1. NFS clients, which access objects as files from AWS Storage Gateway deployed as a file gateway
2. Expandable read/write cache for the AWS file gateway
3. File gateway virtual appliance
4. Amazon S3, which provides persistent object storage for all files that are written using the file gateway

File to Object Mapping

After you deploy, activate, and configure the file gateway, it presents one or more bucket shares that can be mounted by NFS v3 or v4.1 clients on the same LAN. Each share (or mount point) on the gateway is paired to a single bucket, and the contents of the bucket are available as files and folders in the share.

Writing an individual file to a share on the file gateway creates an identically named object in the associated bucket. All newly created objects are written to Amazon S3 Standard or Amazon S3 Standard – Infrequent Access (Standard – IA) storage classes, depending on the configuration of the share.

The Amazon S3 key name of a newly created object is identical to the full path of the file that is written to the mount point in AWS Storage Gateway.

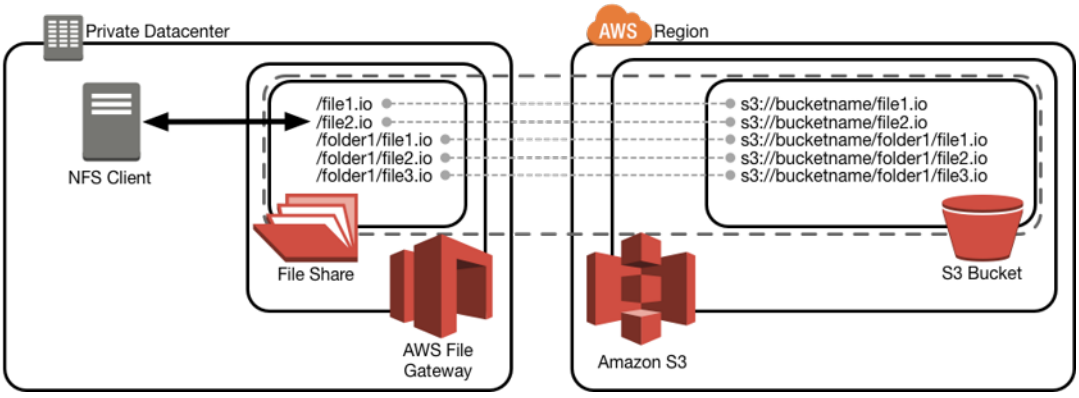

Figure 2: Files stored over NFS on the file gateway mapping to Amazon S3 objects

One difference between storing data in Amazon S3 versus a traditional file system is the way in which granular permissions and metadata are implemented and stored. Access to files that are stored directly in Amazon S3 is secured by policies that are stored in Amazon S3 and AWS Identity and Access Management (IAM). All other attributes, such as storage class and creation date, are stored in a given object's metadata. When you access a file over NFS, the file permissions, folder permissions, and attributes are stored in the file system.

To reliably persist NFS-based file permissions and attributes, the file gateway stores this information as part of Amazon S3 object metadata. If the permissions are changed on a file over NFS, the gateway modifies the metadata of the associated objects that are stored in Amazon S3 to reflect the changes. Custom default UNIX permissions are defined for all existing S3 objects within a bucket when a share is created from the AWS Management Console or using the file gateway API. This feature lets you create NFS-enabled shares from buckets with existing content without having to manually assign permissions after you create the share.

The following is an example of a file that is stored in a share bucket and is listed from a Linux-based client that is mounting the share bucket over NFS. The example shows that the file "file1.txt" has a modification date and standard UNIX file permissions.

```
[e2-user@host]$ ls -l /media/filegateway1/
total 1
-rw-rw-r-- 1 ec2-user ec2-user 36 Mar 15 22:49 file1.txt
[e2-user@host]$
```

The following example shows the output from the head-object on Amazon S3. It shows the same file from the perspective of the object that is stored in Amazon S3. Note that the permissions and time stamp in the previous example are stored durably as metadata for the object.

```
[e2-user@host]$ aws s3api head-object --bucket filegateway1 --
key file1.txt
{
    "AcceptRanges": "bytes",
    "ContentType": "application/octet-stream",
    "LastModified": "Wed, 15 Mar 2017 22:49:02 GMT",
    "ContentLength": 36,
    "VersionId": "93XCzHcBUHBSg2yP.8yKMHzxUumhovEC",
    "ETag": "\"0a7fb5dbb1ae1f6a13c6b4e4dcf54977-1\"",
    "ServerSideEncryption": "AES256",
    "Metadata": {
        "file-group": "500",
        "user-agent-id": "sgw-7619FB1F",
        "file-owner": "500",
        "aws-sgw":
"57c3c3e92a7781f868cb10020b33aa6b2859d58c868190661bcceae87f7b96f
1",
        "file-mtime": "1489618141421",
        "file-ctime": "1489618141421",
        "user-agent": "aws-storage-gateway",
        "file-permissions": "0664"
    }
}
[e2-user@host]$
```

Read/Write Operations and Local Cache

As part of the file gateway deployment, dedicated local storage is allocated to provide a read/write cache for all hosted share buckets. The read/write cache greatly improves response times for NFS operations in the private data center where the file gateway is hosted. The cache holds both recently written and recently read content and does not proactively evict data while the cache disk has free space. However, when the cache is full, AWS Storage Gateway will evict data based on a least recently used (LRU) algorithm. Recently accessed data is available for reads, and write operations are not impeded.

Read Operations (Read-Through Cache)

When an NFS client performs a read request, the file gateway first checks the local cache for the requested data. If the data is not in the cache, the gateway retrieves the data from Amazon S3 using Range GET requests to minimize data transferred over the Internet while repopulating the read cache on behalf of the client.

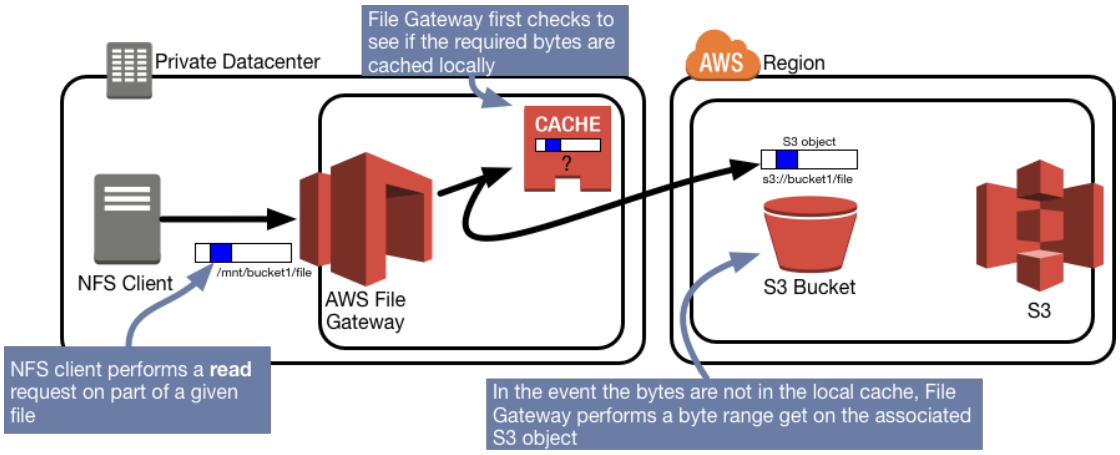

Figure 3: File gateway read operations

Write Operations (Write-Back Cache)

When a file is written to the file gateway over NFS, the gateway first commits the write to the local cache. At that point, it acknowledges the write success to the NFS client, which enables low latency on writes. After the write cache is populated, the file is put into the associated Amazon S3 bucket asynchronously to increase local performance of Internet transfers.

When an existing file is modified, the file gateway transfers only the newly written bytes to the associated Amazon S3 bucket. This uses Amazon S3 API calls to construct a new object from a previous version in combination with the newly uploaded bytes. This feature reduces the amount of data that is required to be transferred when NFS clients modify existing files within the file gateway.

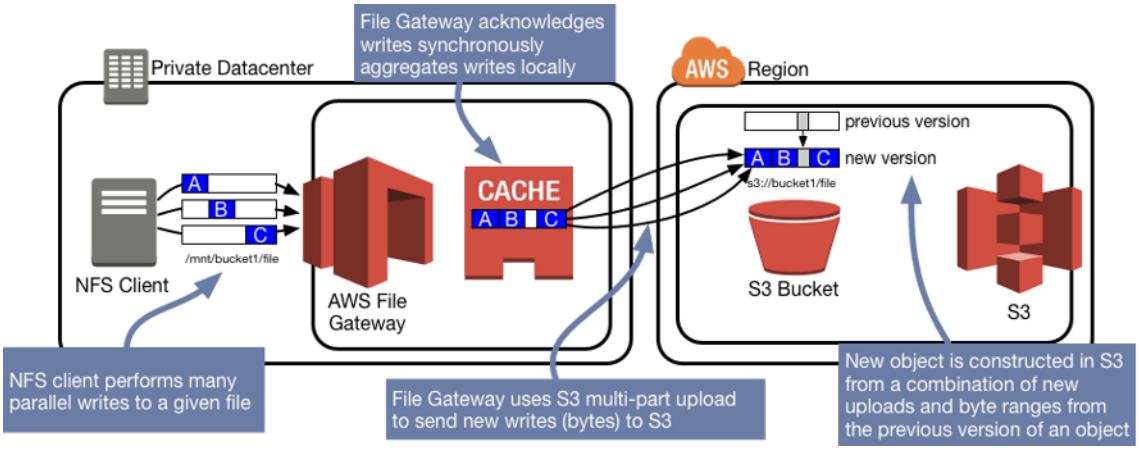

Figure 4: File gateway write operations

NFS Security Within a Local Area Network

When you create a mount point (share) on a deployed gateway, a single Amazon S3 bucket is selected to be the persistent object storage for files and their associated metadata. As part of the configuration of the mount point, default UNIX permissions are defined. These permissions are applied to all existing objects in the Amazon S3 bucket. This ensures that clients that access the mount point adhere to file and directory-level security for existing content.

Figure 5: Default share permissions settings in the file gateway console

In addition, you can help protect an entire mount point and its associated Amazon S3 content on the LAN by limiting mount access to individual hosts or a range of hosts. You can define this by using a Classless Inter-Domain Routing (CIDR) block or individual IP addresses.

Monitoring Cache and Traffic

The cache and Internet requirements that are associated with a given file gateway deployment can change over time as workloads or architectures evolve. To give you visibility into resource use, the file gateway provides statistical information in the form of Amazon CloudWatch metrics. The metrics cover cache consumption, cache hits/misses, data transfer, and read/write metrics.

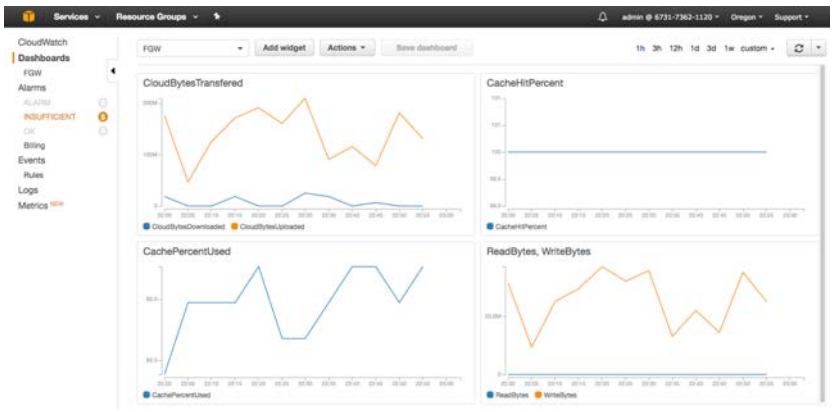

Figure 6: Amazon CloudWatch console with file gateway metrics

File Gateway Bucket Inventory

To reduce both latency and the number of Amazon S3 operations when performing list operations, the file gateway stores a local bucket inventory that contains a record of all recently listed objects. The bucket inventory is populated on-demand as NFS clients list parts of the NFS share for the first time. The file gateway updates inventory records only when the gateway itself modifies, deletes, or creates new objects on behalf of NFS clients. If objects are changed in an NFS bucket by a secondary gateway that is associated with the same Amazon S3 bucket or by any other Amazon S3 API call outside of the file gateway, the file gateway is not aware of those changes.

When Amazon S3 objects have to be modified outside of an NFS share and recognized by the file gateway (such as changes made by Amazon EMR or other AWS services), the bucket inventory must be refreshed using either the **RefreshCache** API call or **RefreshCache** AWS Command Line Interface (CLI) command.

RefreshCache re-inventories the existing records in a file gateway's bucket inventory. This reflects any changes to known objects to the NFS clients that access a given share.

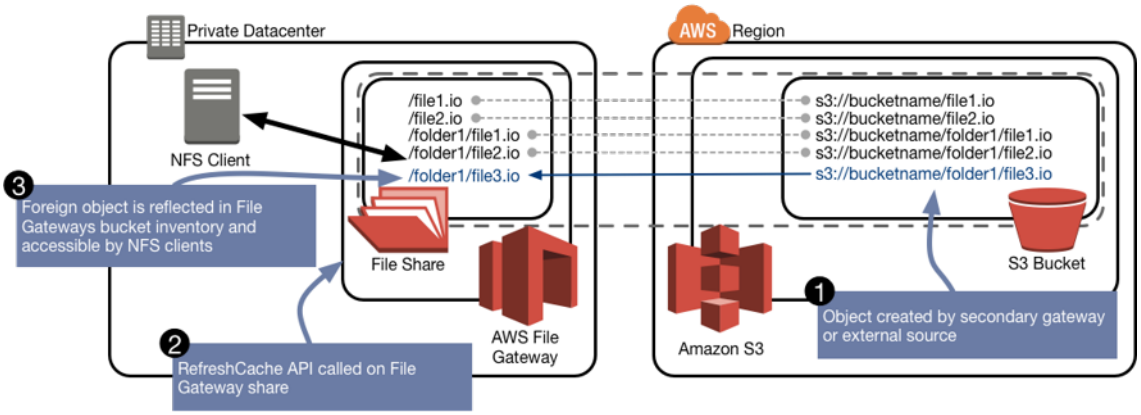

Figure 7: RefreshCache API called to re-inventory Amazon S3 bucket

Bucket Shares with Multiple Contributors

You can deploy more complex architectures that include more than one file gateway share associated with a single Amazon S3 bucket, or in scenarios where a single bucket is being modified by one or more file gateways in conjunction with other Amazon S3-enabled applications. In these more complex architectures, it is important to consider that there is no object locking or coherency across file gateways.

As file gateways are unaware of one another, you should be cautious when you design and deploy solutions that use more than one file gateway share with the same Amazon S3 bucket. File gateways that are associated with the same Amazon S3 bucket are aware of new changes to the content in the bucket only in the following circumstances:

1. A file gateway always recognizes changes it makes to the associated Amazon S3 bucket.
2. A file gateway recognizes changes made to objects by other file gateways when the affected objects are located in folders (or prefixes) that have not been queried by that particular file gateway.
3. A file gateway recognizes changes in an associated Amazon S3 bucket (bucket share) made by other contributors after the **RefreshCache** API is executed.

We highly recommend that you use the read-only mount option on a file gateway share when you deploy multiple gateways that have a common Amazon S3 bucket. Architectures that have only one writer and many readers are the simplest way to avoid write conflicts. If multiple writers are a mandatory requirement in a given architecture, the NFS clients that access each gateway must be tightly controlled to ensure that they don't write to the same objects in the shared Amazon S3 bucket.

When multiple file gateways are accessing the same objects in the same Amazon S3 bucket, you must call the **RefreshCache** API on file gateway shares that have to recognize changes made by other file gateways.

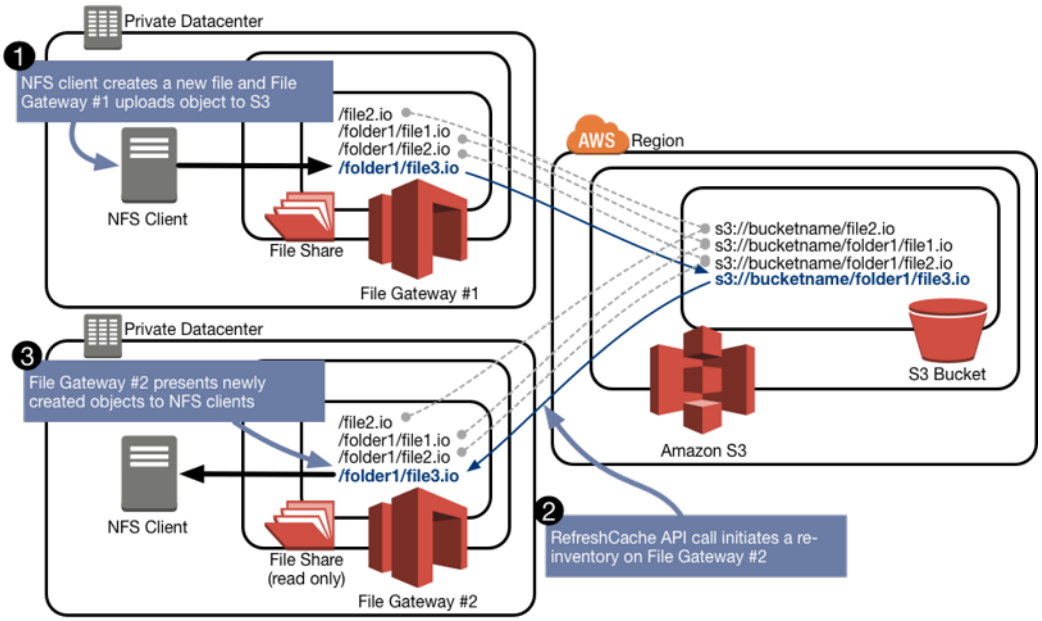

Figure 8: RefreshCache API makes objects created by file gateway #1 visible to file gateway #2

Amazon S3 and the File Gateway

The file gateway uses Amazon S3 buckets to provide storage for each mount point (share) that is created on an individual gateway. When you use Amazon S3 buckets, mount points provide limitless capacity, 99.999999999% durability on objects stored, and a consumption-based pricing model.

You are charged for data stored in Amazon S3 via AWS Storage Gateway based on the Region and storage class. A given mount point writes data directly to Amazon S3 Standard or Amazon S3 Standard – IA storage, depending on the initial configuration that you select when you create the mount point. Both of these storage classes provide equal durability. However, Amazon S3 Standard – IA has a different pricing model and lower availability (i.e., 99.9% compared with 99.99%), which makes it a good solution for less frequently accessed objects. The pricing for Amazon S3 Standard – IA is ideal for objects that exist for more than 30 days and are larger than 128 KB per object.

For details about price differences for Amazon S3 storage classes, visit the Amazon S3 Pricing page.[1]

Using Amazon S3 Object Lifecycle Management for Cost Optimization

Amazon S3 offers three different storage classes: Standard, Standard – Infrequent Access, and Amazon Glacier. You can create Amazon S3 lifecycle policies to automate transitioning objects though the different tiers or even to expire (delete objects) based on their creation date. Lifecycle transitions are supported according to the following table.

Object's initial location	Transition to Amazon S3 Standard	Transition to Amazon S3 Standard - IA	Transition to Amazon Glacier
Amazon S3 Standard	N/A	Yes	Yes
Amazon S3 Standard - IA	No	N/A	Yes
Amazon Glacier	No	No	N/A

You can apply lifecycle policies to an entire Amazon S3 bucket, which reflects a single mount point on a storage gateway, or you can apply them to a specific prefix that reflects a folder within a hosted mount point on a file gateway. The lifecycle policy transition condition is based on the creation date or optionally on the object tag key value pair. For more information about tagging, see Object Tagging in the *Amazon S3 Developer Guide*.[2]

You can use a lifecycle policy in its simplest implementation to move all objects in a given Amazon S3 bucket from Amazon S3 Standard to Amazon S3 Standard – IA, and finally to Amazon Glacier as the data ages. This means that files that are created by the file gateway that

are stored as objects in Amazon S3 buckets can be automatically transitioned to more economical storage classes as the content ages.

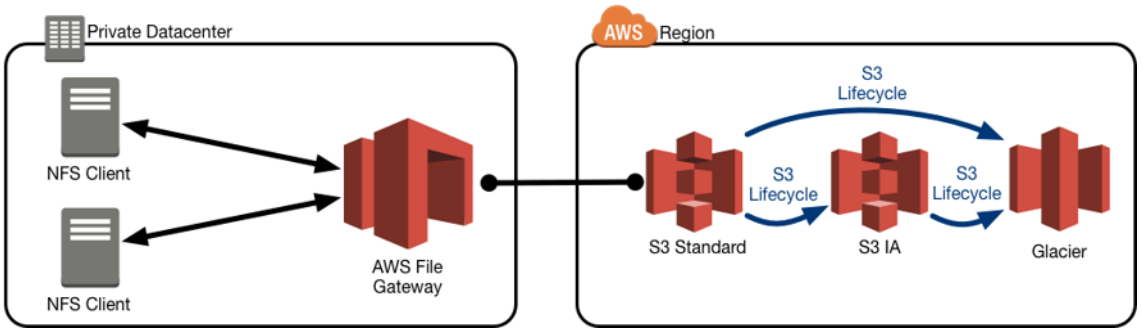

Figure 9: File gateway storing files as objects in Amazon S3 Standard and transitioning to Amazon S3 Standard – IA and Amazon Glacier

Transitioning Objects from Amazon S3 Standard to Amazon S3 Standard – IA and Amazon Glacier

Files that are migrated from Amazon S3 Standard to Amazon S3 Standard – IA using lifecycle policies are immediately available for NFS file read/write operations. Objects that are transitioned to Amazon Glacier are visible when you list NFS files on the file gateway. However, they are not readable unless you restore them to Amazon S3 Standard or to Amazon S3 Standard – IA using an API or the Amazon S3 console.

Trying to read files that are stored as objects in Amazon Glacier results in a read I/O error on the client that tries the read operation. For this reason, using lifecycle to transition files to Amazon Glacier objects is recommended only for file content that does not require immediate access from an NFS client in an AWS Storage Gateway environment.

Amazon S3 Object Replication Across AWS Regions

You can combine Amazon S3 Cross-Region Replication (CRR) with the file gateway architecture to store objects in two Amazon S3 buckets across two separate Regions. CRR is used for a variety of use cases, such as protection against human error, protection against malicious destruction, or to minimize latency to clients in a remote Region. Adding CRR to the file gateway architecture is just one example of how you can use native Amazon S3 tools and features in conjunction with the file gateway.

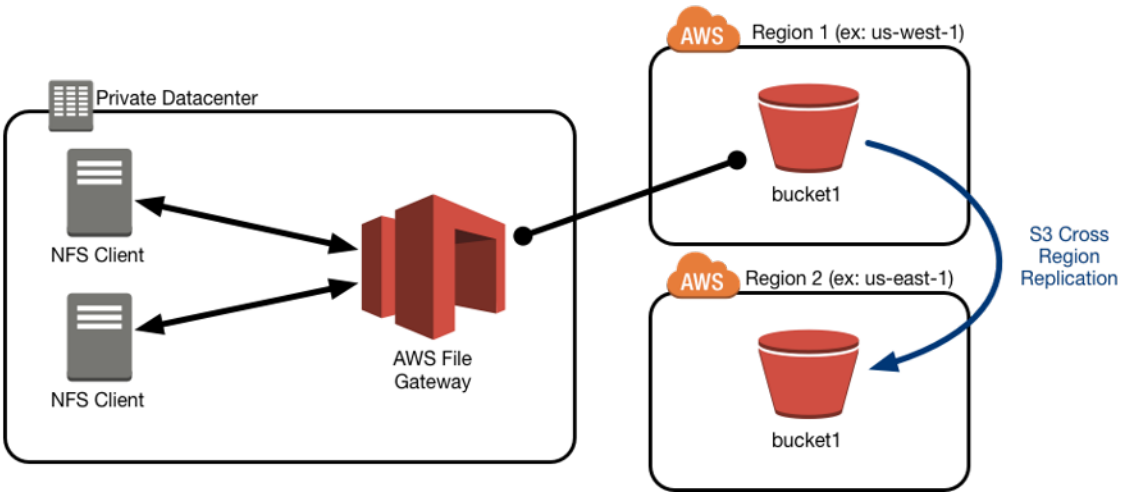

Figure 10: The file gateway in a private data center with CRR to duplicate objects across AWS Regions

AWS File Gateway Use Cases

The following scenarios demonstrate how a file gateway can be used in both cloud tiering and backup architectures.

Cloud Tiering

In on-premises environments where storage resources are reaching capacity, migrating colder data to the file gateway can extend the life span of existing storage on-premises and reduce the need to use capital expenditures on additional storage hardware and data center resources. When you add the file gateway to an existing storage environment, on-premises applications can take advantage of Amazon S3 storage durability, consumption-based pricing, and virtual infinite scale, while ensuring low-latency access to recently accessed data over NFS.

You can tier data using either native host OS tools or third-party tools that have NFS integration.

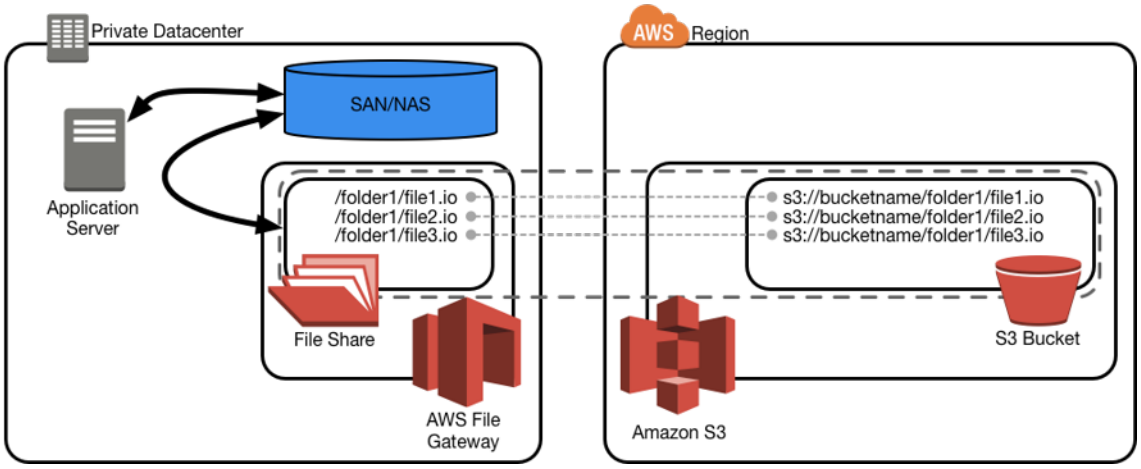

Figure 11: File gateway in a private data center providing Amazon S3 Standard or Amazon S3 Standard – IA as a complement to existing storage deployments

Hybrid Cloud Backup

The file gateway provides a low-latency NFS interface that creates Amazon S3 objects of up to 5 TiB in size, stored in a supported AWS Region. This makes it an ideal hybrid target for backup solutions that can use NFS. By using a mixture of Amazon S3 Standard, Amazon S3 Standard – IA, and Amazon Glacier backup, you can store data on low-cost, highly durable cloud storage and tier it to progressively lower-cost storage as the likelihood of restoration diminishes. Figure 12 shows an example architecture that assumes backups have to be retained for one year. After 30 days the likelihood of restoration becomes infrequent, and after 60 days it becomes extremely rare.

In this solution, Amazon S3 Standard is used as the initial location for backups for the first 30 days. The backup software or scripts write backups to the NFS share, preferably in the form of multi-megabyte size files or larger. Larger files offer better cost optimization in the end-to-end solution, including storage costs in Amazon S3 Standard – IA and Amazon Glacier, and lifecycle transition costs because fewer transitions are required.

After another 30 days, the backups are transitioned to Amazon Glacier. Here they are held until a full year has passed since they were first created, at which point they are deleted.

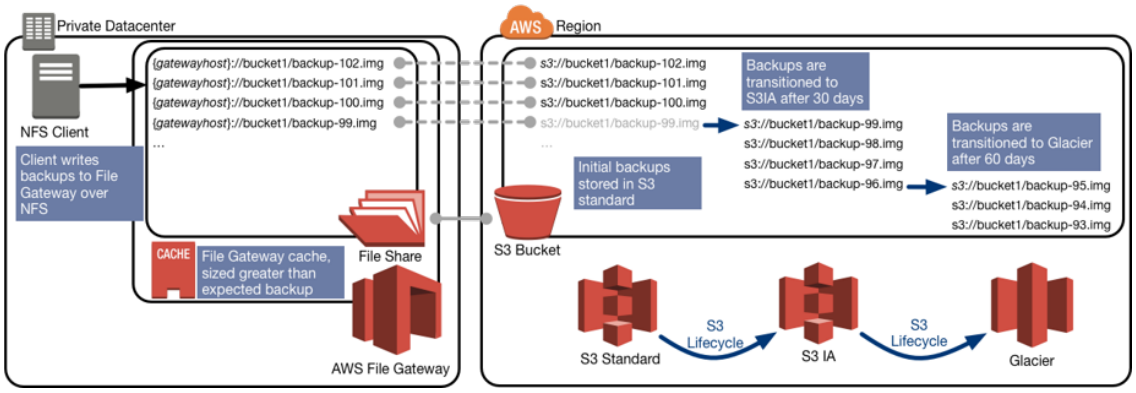

Figure 12: File gateway storing files as objects in Amazon S3 Standard and transitioning to Amazon S3 Standard – IA and Amazon Glacier

When you are sizing cache for the file gateway in this type of solution, it is important that you understand the backup process itself. One approach is to size the cache to be large enough to contain a complete full backup, which allows restores from that backup to come directly from the cache—much more quickly than over a wide-area network (WAN) link.

If the backup solution uses software that consolidates backup files by reading existing backups before writing ongoing backups, it's important to factor that into the sizing of cache also. This is because reading from the local cache during these types of operations reduces cost and increases overall performance of ongoing backup operations.

Conclusion

The file gateway configuration of AWS Storage Gateway provides a simple way to bridge data between private data centers and Amazon S3 storage. The file gateway can enable hybrid architectures for cloud migration, cloud tiering, and hybrid cloud backup.

The file gateway's ability to provide a translation layer between the NFS protocol and Amazon S3 APIs without obfuscation makes it ideal for architectures in which data must remain in its native format and be available both on-premises and in the AWS Cloud.

For more information about the AWS Storage Gateway service, see AWS Storage Gateway.[3]

Contributors

The following individuals and organizations contributed to this document:

- Peter Levett, Solutions Architect, AWS

- David Green, Solutions Architect, AWS

Further Reading

For additional information, see the following:

- AWS Storage Services Overview White Paper[4]
- Additional AWS Whitepapers[5]
- AWS Storage Gateway Documentation[6]
- Additional AWS Documentation[7]

Document Revisions

Date	Description
April 2017	Initial document creation

Notes

[1] https://aws.amazon.com/s3/pricing/

[2] http://docs.aws.amazon.com/AmazonS3/latest/dev/object-tagging.html

[3] https://aws.amazon.com/storagegateway

[4] https://d0.awsstatic.com/whitepapers/Storage/AWS Storage Services Whitepaper-v9.pdf

[5] https://aws.amazon.com/whitepapers/

[6] https://aws.amazon.com/documentation/storage-gateway/

[7] https://aws.amazon.com/documentation/