ECS Overview and Architecture

February 2022

H14071.21

White Paper

Abstract

This document provides a technical overview and design of the Dell ECS software-defined cloud-scale object storage platform.

Dell Technologies

Contents

Executive summary....4

Value of ECS....5

Architecture....7

Appliance hardware models....30

Network separation....35

Security....36

Data integrity and protection....42

Deployment....44

Storage protection overhead....56

Conclusion....58

Appendix: Technical support and resources....59

Executive summary

Introduction

Organizations require options for consuming public cloud services with the reliability and control of a private-cloud infrastructure. Dell ECS is a software-defined, cloud-scale, object storage platform that delivers S3, Atmos, CAS, Swift, NFSv3, and HDFS storage services on a single, modern platform.

With ECS, administrators can easily manage globally distributed storage infrastructure under a single global namespace that provides strong consistency across sites. ECS core components are layered for flexibility and resiliency. Each layer is abstracted and independently scalable with high availability.

Simple RESTful API access for storage services is being embraced by developers. Use of HTTP semantics like GET and PUT simplifies the application logic required when compared with traditional, but familiar, path-based file operations. In addition, ECS's underlying storage system is strongly consistent, which means it can guarantee an authoritative response. Applications that are required to guarantee authoritative delivery of data are able to do so without complex code logic by using ECS.

Audience

This paper is intended for anyone interested in understanding the value and architecture of ECS. It aims to provide context with links to additional information.

Scope

This document provides an overview of the Dell ECS object storage platform. It details the ECS design architecture and core components such as the storage services and data protection mechanisms.

This document focuses primarily on ECS architecture. It does not cover installation, administration, and upgrade procedures for ECS software or hardware. It also does not cover specifics on using and creating applications with the ECS APIs.

Updates to this document are done periodically and generally coincide with major releases or new features.

Revisions

Date	Description
December 2015	Initial release
May 2016	Updated for 2.2.1
September 2016	Updated for 3.0
August 2017	Updated for 3.1
March 2018	Updated for 3.2
September 2018	Updated for Gen3 Hardware
February 2019	Updated for 3.3
September 2019	Updated for 3.4
February 2020	Updated ECSDOC-628 changes

Date	Description
May 2020	Updated for 3.5
November 2020	Updated for 3.6
February 2021	Updated for 3.6.1
August 2021	Updated for 3.6.2 and compression mechanism
December 2021	Updated template
February 2022	Updated for 3.7

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Author: Jarvis Zhu

Note: For links to other documentation for this topic, see the ECS Info Hub.

Value of ECS

ECS provides significant value for enterprises and service providers seeking a platform architected to support rapid data growth. The main advantages and features of ECS that enable enterprises to globally manage and store distributed content at scale include:

Cloud Scale - ECS is an object storage platform for both traditional and next-gen workloads. ECS's software-defined layered architecture promotes limitless scalability. Feature highlights are:

- Globally distributed object infrastructure
- Exabyte+ scale without limits on storage pool, cluster or federated environment capacity
- No limits exist on the number of objects in a system, namespace or bucket
- Efficient at both small and large file workloads with no limits to object size

Flexible Deployment - ECS has unmatched flexibility with features such as:

- Appliance deployment
- Software-only deployment with support for certified or custom industry standard hardware
- Multiprotocol support: Object (S3, Swift, Atmos, CAS) and File (HDFS, NFSv3)
- Multiple workloads: Modern apps and traditional apps
- Secondary storage for Data Domain Cloud Tier and Isilon using CloudPools
- Non-disruptive upgrade paths to current generation ECS models

Enterprise Grade - ECS provides customers more control of their data assets with enterprise class storage in a secure and compliant system with features such as:

- Data-at-rest (D@RE) with key rotation and external key management.
- Encrypted inter-site communication
- Reporting, policy and event based record retention and platform hardening for SEC Rule 17a-4(f) compliance including advanced retention management such as litigation hold and min-max governance
- Compliance with Defense Information Systems Agency (DISA) Security Technical Implementation Guide (STIG) hardening guidelines
- Authentication, authorization and access controls with Active directory and LDAP
- Integration with monitoring and alerting infrastructure (SNMP traps and SYSLOG)
- Enhanced enterprise capabilities (multi-tenancy, capacity monitoring and alerting)

TCO Reduction - ECS can dramatically reduce Total Cost of Ownership (TCO) relative to both traditional storage and public cloud storage. It even offers a lower TCO than tape for long-term retention. Features include:

- Global namespace
- Small and large file performance
- Seamless Centera migration
- Fully compliant with Atmos REST
- Low management overhead
- Small data center footprint
- High storage utilization

The design of ECS is optimized for the following primary use cases:

- **Modern Applications** - ECS designed for modern development such as for next-gen web, mobile and cloud applications. Application development is simplified with strongly-consistent storage. Along with multi-site, simultaneous multi-user read/write access, as the ECS capacity changes and grows, developers never need to recode their apps.
- **Secondary Storage** - ECS is used as secondary storage to free up primary storage of infrequently accessed data, while also keeping it reasonably accessible. Examples are policy-based tiering products such as Data Domain Cloud Tier and Isilon CloudPools. GeoDrive, a Windows-based application, gives Windows systems direct access to ECS to store data.
- **Geo-Protected Archive** - ECS serves as a secure and affordable on-premises cloud for archival and long-term retention purposes. Using ECS as an archive tier can significantly reduce primary storage capacities. To allow for better storage efficiencies for cold archive use cases a 10+2 erasure coding (EC) scheme is available in addition to the default of 12+4.

- **Global Content Repository** - Unstructured content repositories containing data such as images and videos are often stored in high cost storage systems making it impossible for businesses to cost-effectively manage massive data growth. ECS enables consolidation of multiple storage systems into a single, globally accessible and efficient content repository.
- **Storage for Internet of Things** - The Internet of Things (IoT) offers a new revenue opportunity for businesses who can extract value from customer data. ECS offers an efficient IoT architecture for unstructured data collection at massive scale. With no limits on the number of objects, the size of objects or custom metadata, ECS is the ideal platform to store IoT data. ECS can also streamline some analytic workflows by allowing data to be analyzed directly on the ECS platform without requiring time consuming extract, transform and load (ETL) processes. Hadoop clusters can run queries using data stored on ECS by another protocol API such as S3 or NFS.
- **Video Surveillance Evidence Repository** - In contrast to IoT data, video surveillance data has a much smaller object storage count, but a much higher capacity footprint per file. While data authenticity is important, data retention is not as critical. ECS can be a low-cost landing area or secondary storage location for this data. Video management software can leverage the rich custom metadata capabilities for tagging files with important details like camera location, retention requirement and data protection requirement. Also, metadata can be used to set the file to a read-only status to ensure a chain of custody on the file.
- **Data lakes and Analytics** - Data and analytics have become a competitive differentiator and a primary source of value generation for organizations. However, transforming data into a valuable corporate asset is a complex topic that can easily entail the use of dozens of technologies, tools, and environments. ECS provide a set of services to help customer collecting, storing, governing, and analyzing data at any scale.

Architecture

Introduction

ECS is architected with a few core design principles, such as global namespace with strong consistency; scale-out capability, secure multi-tenancy; and superior performance for both small and large objects. ECS is built as a completely distributed system following the principle of cloud applications, where every function in the system is built as an independent layer. With this design, each layer is horizontally scalable across all nodes in the system. Resources are distributed across all nodes to increase availability and share the load.

This section will go in-depth into the ECS architecture and design of the software and hardware.

Architecture overview

ECS is deployed on a set of qualified industry standard hardware or as a turnkey storage appliance. The main components of ECS are the:

- **ECS Portal and Provisioning Services** - API-based WebUI and CLI for self-service, automation, reporting and management of ECS nodes. This layer also handles licensing, authentication, multi-tenancy and provisioning services such as namespace creation.
- **Data Services** - Services, tools and APIs to support object and file access to the system.
- **Storage Engine** - Core service responsible for storing and retrieving data, managing transactions, and protecting and replicating data locally and between sites.
- **Fabric** - Clustering service for health, configuration and upgrade management and alerting.
- **Infrastructure** - SUSE Linux Enterprise Server 12 for the base operating system in the turnkey appliance or qualified Linux operating systems for industry standard hardware configuration.
- **Hardware** - A turnkey appliance or qualified industry standard hardware.

The following figure shows a graphical view of these layers which are described in detail in the sections that follow.

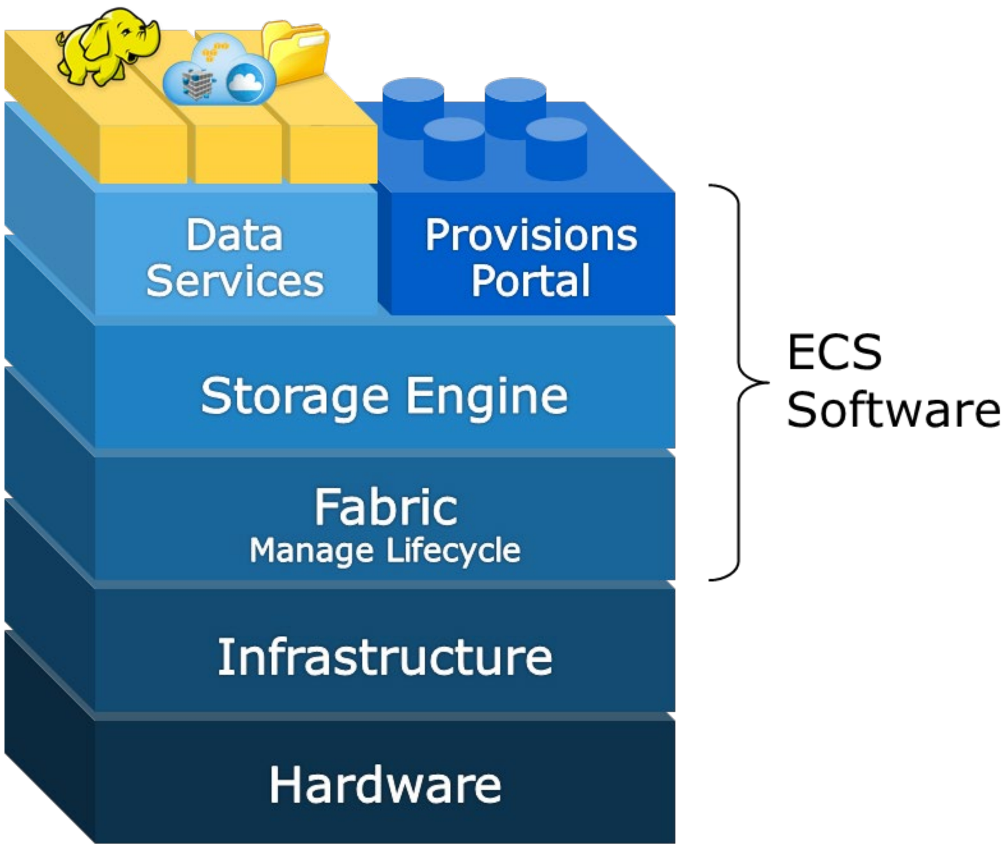


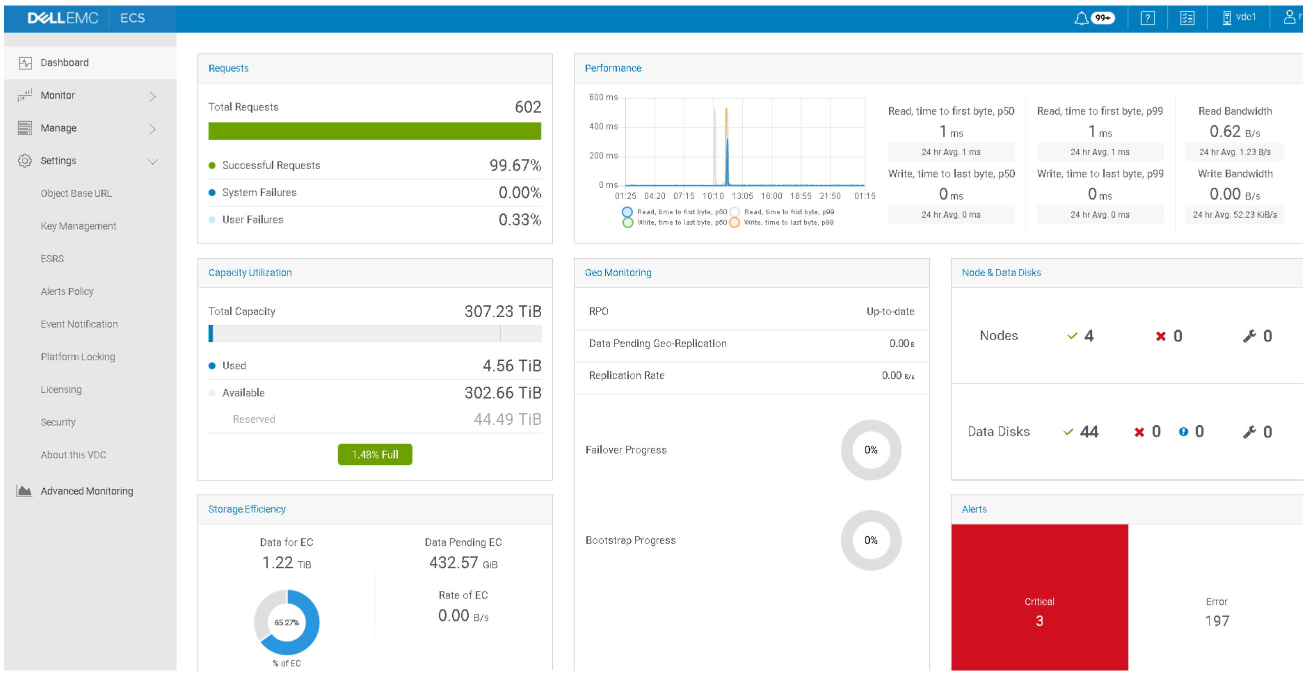
Figure 1. ECS architecture layers

ECS portal and provisioning services

Storage administrators manage ECS using the ECS Portal and provisioning services. ECS provides a web-based GUI (WebUI) to manage, license and provision ECS nodes. The portal has comprehensive reporting capabilities that include:

- Capacity utilization per site, storage pool, node and disk.
- Performance monitoring on latency, throughput, and replication progress.
- Diagnostic information, such as node and disk recovery status.

The ECS dashboard provides overall system-level health and performance information. This unified view enhances overall system visibility. Alerts notify users about critical events, such as capacity limits, quota limits, disk or node failures or software failures. ECS also provides a command-line interface to install, upgrade and monitor ECS. Access to nodes for command-line usage is done via SSH. The following figure shows the ECS dashboard:



Requests

Total Requests	602
Successful Requests	99.67%
System Failures	0.00%
User Failures	0.33%

Performance

Read, time to first byte, p50	Read, time to first byte, p99	Read Bandwidth
1 ms	1 ms	0.62 B/s
24 hr Avg. 1 ms	24 hr Avg. 1 ms	24 hr Avg. 1.23 B/s
Write, time to last byte, p50	Write, time to last byte, p99	Write Bandwidth
0 ms	0 ms	0.00 B/s
24 hr Avg. 0 ms	24 hr Avg. 0 ms	24 hr Avg. 52.23 KiB/s

Capacity Utilization

Total Capacity	307.23 TiB
Used	4.56 TiB
Available	302.66 TiB
Reserved	44.49 TiB

1.48% Full

Geo Monitoring

RPO	Up-to-date
Data Pending Geo-Replication	0.00 B
Replication Rate	0.00 B/s
Failover Progress	0%
Bootstrap Progress	0%

Node & Data Disks

Nodes	✓ 4	✖ 0		🔧 0
Data Disks	✓ 44	✖ 0	❗ 0	🔧 0

Storage Efficiency

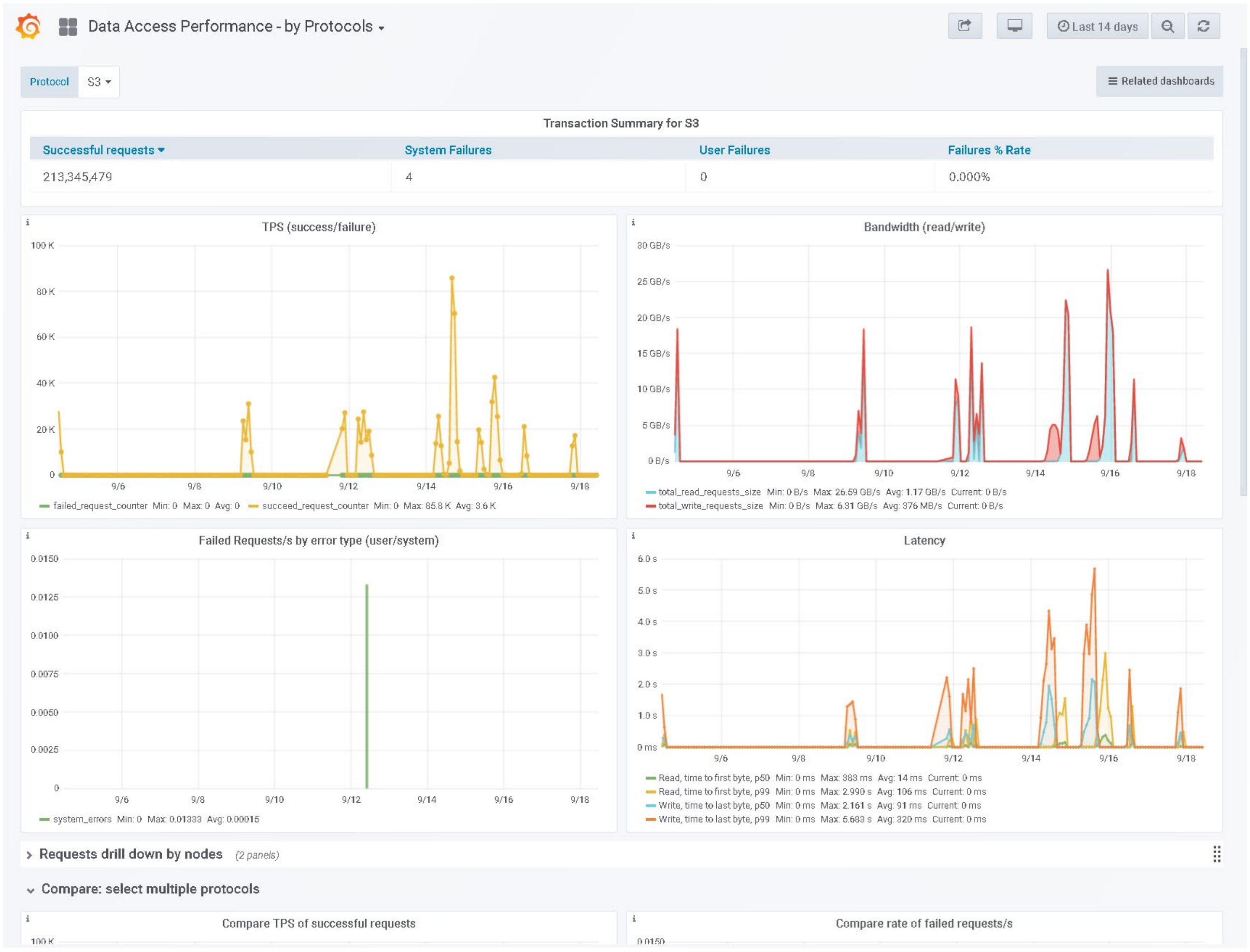
Data for EC	Data Pending EC
1.22 TiB	432.57 GiB
	Rate of EC
	0.00 B/s

Alerts

Critical	Error
3	197

Figure 2. ECS Web UI dashboard

Detailed performance reporting is available in the UI under the Advance Monitoring folder. The reports are displayed in a Grafana dashboard. There are filters available to drill into specified Namespaces, Protocols or Nodes. The following figure shows an example of an S3 protocol performance report:


Data Access Performance - by Protocols
Last 14 days
Protocol
S3
Related dashboards
TPS (success/failure)
Bandwidth (read/write)
Failed Requests/s by error type (user/system)
Latency
Requests drill down by nodes (2 panels)
Compare: select multiple protocols
Compare TPS of successful requests
Compare rate of failed requests/s

Transaction Summary for S3

Successful requests	System Failures	User Failures	Failures % Rate
213,345,479	4	0	0.000%

```
failed_request_counter  Min: 0  Max: 0  Avg: 0   succeed_request_counter  Min: 0  Max: 85.8 K  Avg: 3.6 K
total_read_requests_size  Min: 0 B/s  Max: 26.59 GB/s  Avg: 1.17 GB/s  Current: 0 B/s
total_write_requests_size  Min: 0 B/s  Max: 6.31 GB/s  Avg: 376 MB/s  Current: 0 B/s
system_errors  Min: 0  Max: 0.01333  Avg: 0.00015
Read, time to first byte, p50  Min: 0 ms  Max: 383 ms  Avg: 14 ms  Current: 0 ms
Read, time to first byte, p99  Min: 0 ms  Max: 2.990 s  Avg: 106 ms  Current: 0 ms
Write, time to last byte, p50  Min: 0 ms  Max: 2.161 s  Avg: 91 ms  Current: 0 ms
Write, time to last byte, p99  Min: 0 ms  Max: 5.683 s  Avg: 320 ms  Current: 0 ms
```

Figure 3. Advanced monitoring visualization using Grafana

ECS can also be managed using RESTful APIs. The management API allows users to administer ECS within their own tools, scripts and new or existing applications. The ECS web UI and command-line tools are built using the ECS REST Management APIs.

ECS supports the following event notification servers which can be set using the web UI, API or CLI:

- SNMP (Simple Network Management Protocol) servers
- Syslog servers

The *ECS Administrator's Guide* has more information and details on configuring notification services.

Data services

Standard object and file methods are used to access ECS storage services. For S3, Atmos and Swift, RESTful APIs over HTTP are used for access. For Content Addressable Storage (CAS), a proprietary access method/SDK is used. ECS natively supports all the NFSv3 procedures except for LINK. ECS buckets can now be accessed by S3a.

ECS provides multi-protocol access where data ingested through one protocol can be accessed through others. This means that data can be ingested through S3 and modified through NFSv3 or Swift, or vice versa. There are some exceptions to multi-protocol access due to protocol semantics and representations of protocol design. The following table highlights the access methods and which protocols interoperate.

Table 1. ECS supported data services and protocol interoperability

Protocol		Supported	Interoperability
Object	S3	Additional capabilities like Byte Range Updates and Rich ACLS	HDFS, NFS, Swift
	Atmos	Version 2.0	NFS (path-based objects only and not object ID style based)
	Swift	V2 APIs and Swift and Keystone v3 Authentication	HDFS, NFS, S3
	CAS	SDK v3.1.544 or later	N/A
File	HDFS	Hadoop 2.7 compatibility	S3, NFS, Swift
	NFS	NFSv3	S3, Swift, HDFS, Atmos (path-based objects only and not object ID style based)

Data services, which are also referred to as head services, are responsible for taking client requests, extracting required information, and passing it to the storage engine for further processing. All head services are combined to a single process, *dataheadsvc,* running inside the infrastructure layer. This process is further encapsulated within a Docker container named *object-main.* which runs on every node within ECS. Infrastructure covers Docker in more detail. ECS protocol service port requirements, such as port 9020 for S3 communication, are available in the latest *ECS Security Configuration Guide*.

Object

ECS supports S3, Atmos, Swift and CAS APIs for object access. Except for CAS, objects or data are written, retrieved, updated, and deleted via HTTP or HTTPS calls of GET, POST, PUT, DELETE and HEAD. For CAS, standard TCP communication and specific access methods and calls are used.

ECS provides a facility for metadata search for objects using a rich query language. This is a powerful feature of ECS that allows S3 object clients to search for objects within buckets using system and custom metadata. While search is possible using any metadata, by searching on metadata that has been specifically configured to be indexed in a bucket, ECS can return queries quicker, especially for buckets with billions of objects.

Metadata search with tokenization allows the customer to use metadata search to search for objects that have a specific metadata value within an array of metadata values. The method must be chosen when the bucket is created. It can be included as an option when creating the bucket through the S3 create bucket API, and include the `header x-emc-metadata-search-tokens: true` in the request.

Up to thirty user-defined metadata fields can be indexed per bucket. Metadata is specified at the time of bucket creation. Metadata search feature can be enabled on buckets with server-side encryption enabled; however, any indexed user metadata attribute utilized as a search key will not be encrypted.

Note: There is a performance impact when writing data in buckets configured to index metadata. The impact to operations increases as the number of indexed fields increases. Impact to performance needs careful consideration on choosing if to index metadata in a bucket, and if so, how many indexes to maintain.

For CAS objects, CAS query API provides similar ability to search for objects based on metadata that is maintained for CAS objects which does not need to be enabled explicitly.

For more information on ECS APIs and APIs for metadata search see the latest *ECS Data Access Guide*. For Atmos and S3 SDKs refer to the GitHub site Dell EMC Data Services SDK or Dell ECS. For CAS refer to the Centera Community site. Access to numerous examples, resources and assistance for developers can be found in the ECS Community.

Client applications such as S3 Browser and Cyberduck provide a way to quickly test or access data stored in ECS. ECS Test Drive is freely provided by Dell which allows access to a public facing ECS system for testing and development purposes. After registering for ECS Test Drive, REST endpoints are provided with user credentials for each of the object protocols. Anyone can use ECS Test drive to test their S3 API application.

Note: Only the number of metadata that can be indexed per bucket is limited to thirty in ECS. There is no limitation to the total number of custom metadata stored per object, only the number indexed for fast lookup.

HDFS

ECS can store Hadoop file system data. As a Hadoop-compatible file system, organizations can create big data repositories on ECS that Hadoop analytics can consume and process. The HDFS data service is compatible with Apache Hadoop 2.7, with support for fine-grained ACLs and extended filesystem attribute.

ECS has been validated and tested with Hortonworks (HDP 2.7). ECS also has support for services such as YARN, MapReduce, Pig, Hive/Hiveserver2, HBase, Zookeeper, Flume, Spark, and Sqoop.

Hadoop S3A support

ECS supports the Hadoop S3A client for storing Hadoop data. S3A is an open source connector for Hadoop, based on the official Amazon Web Services (AWS) SDK. It was created to address storage scaling and cost problems that many Hadoop admin were having with HDFS. Hadoop S3A connects Hadoop clusters to any S3 compatible object store whether in the public, hybrid, or on-premises cloud.

Note: S3A support is available on Hadoop 2.7 or later version.

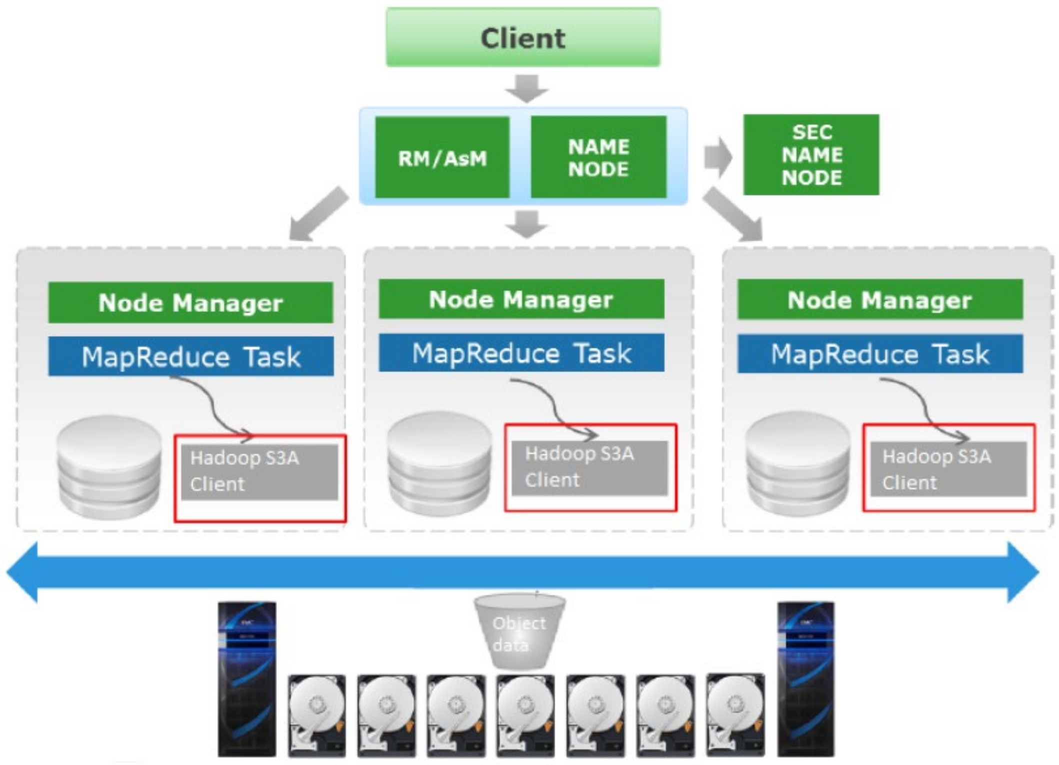


Figure 4. Hadoop and ECS architecture

As shown in the preceding figure, when the Hadoop cluster is set up on traditional HDFS, its S3A configuration points to the ECS Object data to do all the HDFS activity. On each Hadoop HDFS node, any traditional Hadoop component would use the Hadoop's S3A client to perform the HDFS activity.

Hadoop configuration analysis using ECS Service Console

The ECS Service Console (SC) can read and interpret your Hadoop configuration parameters with respect to connections to ECS for S3A. Also, SC provides a function, *Get_Hadoop_Config* that reads the Hadoop cluster configuration and checks S3A settings for typos, errors, and values. Contact ECS support team for assistance with installing ECS SC.

Privacera implementation with Hadoop S3A

Privacera is a third-party vendor that has implemented a Hadoop client-side agent and integration with Ambari for S3 (AWS and ECS) granular security. Although Privacera supports Cloudera Distribution of Hadoop (CDH), Cloudera (another third-party vendor) does not support Privacera on CDH.

Note: CDH users must use ECS IAM security services. If you want secure access to S3A without using ECS IAM, contact the support team.

See the latest *ECS Data Access Guide* for further information on S3A support

Hadoop S3A security

ECS IAM allows the Hadoop administrator to setup access policies to control access to S3A Hadoop data. Once the access policies are defined, there are two user access options for Hadoop administrators to configure:

- IAM Users/Groups
  - Create IAM groups that attach to policies
  - Create IAM users that are members of an IAM group
- SAML Assertions (Federated Users)
  - Create IAM roles that attach to policies
  - Configure CrossTrustRelationship between Identity Provider (AD FS) and ECS that map AD groups to IAM roles

ECS admin and Hadoop admin need to work together to pre-define appropriate policies. The fictional examples that follow outline three types of Hadoop users that we will create policies for. They are:

- **Hadoop Administrator -** do all operations, except create bucket and delete bucket
- **Hadoop Power User -** do all operations except create bucket, delete bucket and delete objects
- **Hadoop Read Only User -** only list and read objects

For more information about ECS IAM, see ECS IAM.

ECS HDFS client support

ECS has been integrated with Ambari, which allows you to easily deploy the ECS HDFS client jar file and specify ECS HDFS as the default filesystem in a Hadoop cluster. The jar file is installed on each node within a participating Hadoop cluster. ECS provides file system and storage functionality equivalent to what name and data nodes do in a Hadoop deployment. ECS streamlines the workflow of Hadoop by eliminating the need for migration of data to a local Hadoop DAS and/or creating a minimum of three copies. The following figure shows the ECS HDFS Client jar file installed on each Hadoop compute node and the general communication flow:

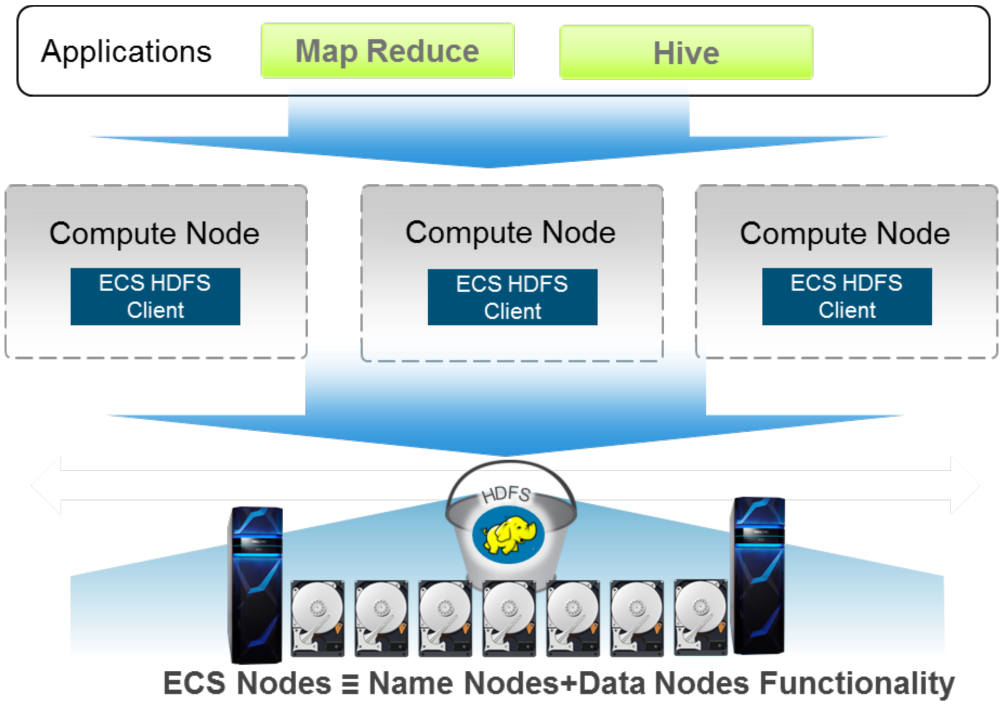


Figure 5. ECS serving as name and data nodes for a Hadoop cluster

Other enhancements added in ECS for HDFS include the following:

- **Proxy user authentication** - Impersonation for Hive, HBase, and Oozie.
- **Security** - Server-side ACL enforcement and addition of Hadoop superuser and superuser group as well as default group on buckets.

NFS

ECS includes native file support with NFSv3. The main features for the NFSv3 file data service include:

- **Global namespace** - File access from any node at any site.
- **Global locking** - In NFSv3 locking is **advisory only**. ECS supports compliant client implementations that allow for shared and exclusive, range-based and mandatory locks.
- **Multiprotocol access** - Access to data using different protocol methods.

NFS exports, permissions and user group mappings are created using the WebUI or API. NFSv3 compliant clients mount exports using namespace and bucket names. Here is a sample command to mount a bucket:

```
mount -t nfs -o vers=3 s3.dell.com:/namespace/bucket
```

To achieve client transparency during a node failure, a load balancer is recommended for this workflow.

ECS has tightly integrated the other NFS server implementations, such as *lockmgr*, *statd*, *nfsd*, and *mountd*, hence, these services are not dependent on the infrastructure layer (host operating system) to manage. NFSv3 support has the following features:

- No design limits on the number of files or directories.
- File write size can be up to 16TB.
- Ability to scale across up to 8 sites with a single global namespace/export.
- Support for Kerberos and AUTH_SYS authentication.

NFS file services process NFS requests coming from clients; however, data is stored as objects within ECS. An NFS file handle is mapped to an object id. Since the file is basically mapped to an object, NFS has features like the object data service, including:

- Quota management at the bucket level.
- Encryption at the object level.
- Write-Once-Read-Many (WORM) to the bucket level.
  - WORM is implemented using Auto Commit period during new bucket creation.
  - WORM is only applicable to non-compliant buckets.

Connectors and gateways

Several third-party software products have the capability to access ECS object storage. Independent software vendors (ISVs) such as Panzura, Ctera and Syncplicity create a layer of services that offer client access to ECS object storage via traditional protocols such as SMB/CIFS, NFS and iSCSI. Organizations can also access or upload data to ECS storage with the following Dell products:

- **Isilon CloudPools** - Policy-based tiering of data to ECS from Isilon.
- **Data Domain Cloud Tier** - Automated native tiering of deduplicated data to ECS from Data Domain for long-term retention. Data Domain Cloud Tier provides a secure and cost-effective solution to encrypt data in the cloud with a reduced storage footprint and network bandwidth.
- **GeoDrive** - ECS stub-based storage service for Microsoft® Windows® desktops and servers.

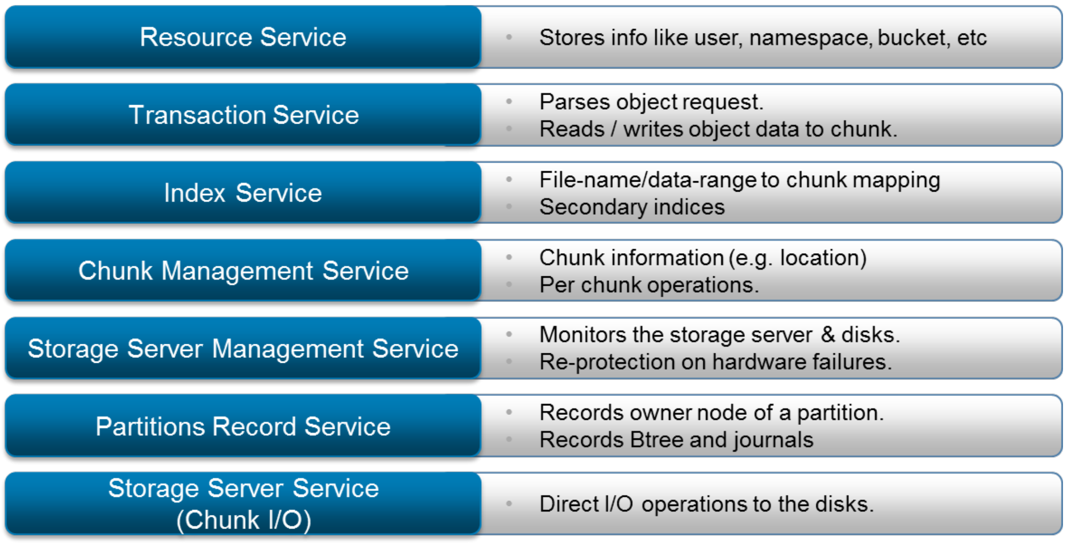
Storage engine

At the core of ECS is the storage engine. The storage engine layer contains the main components responsible for processing requests as well as storing, retrieving, protecting and replicating data.

This section describes the design principles and how data is represented and handled internally.

Storage services

The ECS storage engine includes the services shown in the following figure:



Resource Service	• Stores info like user, namespace, bucket, etc
Transaction Service	• Parses object request. • Reads / writes object data to chunk.
Index Service	• File-name/data-range to chunk mapping • Secondary indices
Chunk Management Service	• Chunk information (e.g. location) • Per chunk operations.
Storage Server Management Service	• Monitors the storage server & disks. • Re-protection on hardware failures.
Partitions Record Service	• Records owner node of a partition. • Records Btree and journals
Storage Server Service (Chunk I/O)	• Direct I/O operations to the disks.

Figure 6. Storage engine services

The services of the Storage Engine are encapsulated within a Docker container that runs on every ECS node to provide a distributed and shared service.

Data

The primary types of data stored in ECS can be summarized as follows:

- **Data** - Application- or user-level content stored such as an image. Data is used synonymously with object, file or content. Applications may store an unlimited amount of custom metadata with each object. The storage engine writes data and associated application-provided custom metadata together in a logical repository. Custom metadata is a robust feature of modern storage systems that provide further information or categorization of the data being stored. Custom metadata is formatted as key-value pairs and provided with write requests.
- **System metadata** - System information and attributes relating to user data and system resources. System metadata can be broadly categorized as follows:
  - **Identifiers and descriptors** - A set of attributes used internally to identify objects and their versions. Identifiers are either numeric ids or hash values which are not of use outside the ECS software context. Descriptors define information such as type of encoding.
  - **Encryption keys in encrypted format** - Data encryption keys are considered system metadata. They are stored in encrypted form inside the core directory table structure.
  - **Internal flags** - A set of indicators used to track if byte range updates or encryption are enabled, as well as to coordinate caching and deletion.
  - **Location information** - Attribute set with index and data location such as byte offsets.
  - **Timestamps** - Attribute set that tracks time such as for object create or update.
  - **Configuration/tenancy information** - Namespace and object access control.

Data and system metadata are written in *chunks* on ECS. An ECS chunk is a 128MB logical container of contiguous space. Each chunk can have data from different objects, as shown below in the following figure. ECS uses indexing to keep track of all the parts of an object that may be spread across different chunks and nodes.

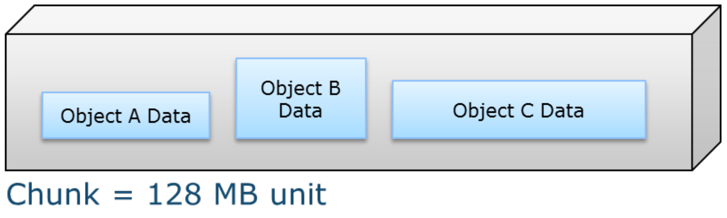


Figure 7. 128MB chunk storing data of three objects

Chunks are written in an append-only pattern. The append-only behavior means that an application's request to modify or update an existing object will not modify or delete the previously written data within a chunk, but rather the new modifications or updates will be written in a new chunk. Therefore, no locking is required for I/O and no cache invalidation is required. The append-only design also simplifies data versioning. Old versions of the data are maintained in previous chunks. If S3 versioning is enabled and an older version of the data is needed, it can be retrieved or restored to a previous version using the S3 REST API.

Data integrity and protection explains how data is protected at the chunk level.

Data management

ECS has a built-in snappy compression mechanism. The granularity is 2MB for small objects and 128MB for large objects. ECS employs a smart compression logic where it only compresses data that is compressible, saving resources from trying to compress already compressed or in-compressible data (such as encrypted data or video files). If more sophisticated compression is required, the ECS Java SDK supports client side ZIP and LZMA.

ECS uses a set of logical tables to store information relating to the objects. Key-value pairs are eventually stored on disk in a B+ tree for fast indexing of data locations. By storing the key-value pair in a balanced, searched tree like a B+ tree, the location of the data and metadata can be accessed quickly. ECS implements a two-level log-structured merge tree where there are two tree-like structures; a smaller tree is in memory (memory table) and the main B+ tree resides on disk. Lookup of key-value pairs occurs in memory first subsequently at the main B+ tree on disk if needed. Entries in these logical tables are first recorded in journal logs and these logs are written to disks in triple-mirrored chunks. The journals are used to track transactions not yet committed to the B+ tree. After each transaction is logged into a journal, the in-memory table is updated. Once the table in the memory becomes full or after a certain period, its merge sorted or dumped to B+ tree on disk. The number of journal chunks used by the system is insignificant when compared to B+ tree chunks. The following figure illustrates this process:

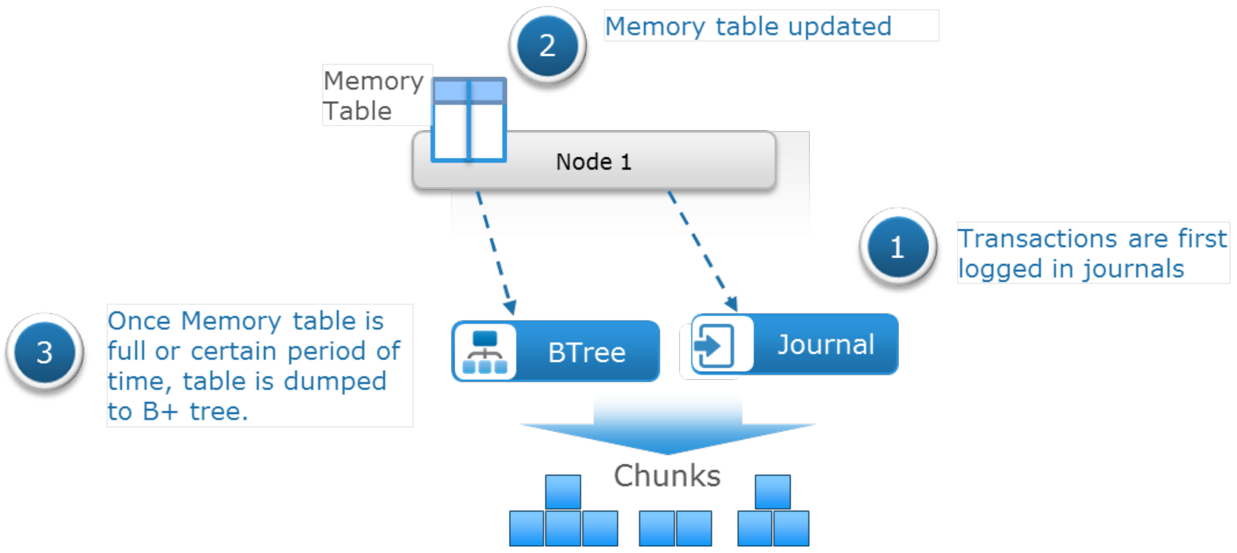

Memory table updated
Memory Table
Node 1
Transactions are first logged in journals
Once Memory table is full or certain period of time, table is dumped to B+ tree.
BTree
Journal
Chunks

Figure 8. Memory table dumped to B+ tree

The following table shows the information stored in the Object (OB) table. The OB table contains the names of objects and their chunk location at a certain offset and length within that chunk. In this table, the object name is the key to the index and the value is the chunk location. The index layer within the Storage Engine is responsible for the object name-to-chunk mapping.

Table 2. Object table entries

Object name	Chunk location
ImgA	C1:offset:length
FileB	• C2:offset:length • C3:offset:length

The chunk table (CT) records the location for each chunk, as detailed in the following table:

Table 3. Chunk table entries

Chunk ID	Location
C1	• Node1:Disk1:File1:Offset1:Length • Node2:Disk2:File1:Offset2:Length • Node3:Disk2:File6:Offset:Length

ECS was designed to be a distributed system such that storage and access of data are spread across all nodes. Tables used to manage object data and metadata grow over time as the storage is used and grows. The tables are divided into partitions and assigned to different nodes where each node becomes the owner of the partitions it is hosting for each of the tables. To get the location of a chunk, for example, the Partition Records table (PR) is queried for owner node which has knowledge of the chunk location. A basic PR table is illustrated in the following table:

Table 4. Partition records table entries

Partition ID	Owner
P1	Node 1
P2	Node 2
P3	Node 3

If a node goes down, other nodes take ownership of its partitions. The partitions are recreated by reading the B+ tree root and replaying the journals stored on disk. The following figure shows the failover of partition ownership:

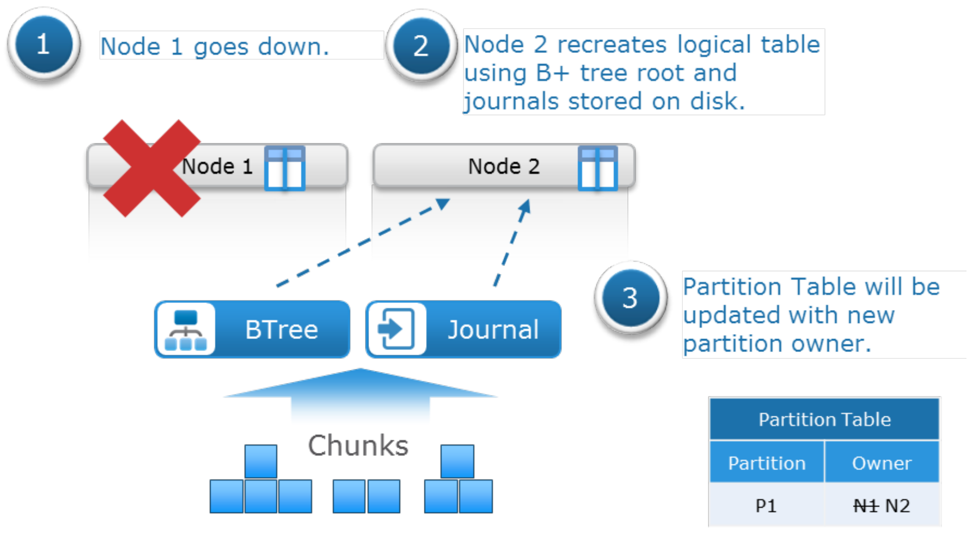

1
Node 1 goes down.
2
Node 2 recreates logical table using B+ tree root and journals stored on disk.
Node 1
Node 2
BTree
Journal
3
Partition Table will be updated with new partition owner.
Chunks
Partition Table
Partition
Owner
P1
~~N1~~ N2

Figure 9. Failover of partition ownership

Data flow

Storage services are available from any node. Data is protected by distributed EC segments across drives, nodes and racks. ECS runs a checksum function and stores the result with each write. If the first few bytes of data are compressible ECS will compress the data. With reads, data is decompressed, and its stored checksum is validated. Here is an example of a data flow for a write in five steps:

1. Client sends object create request to a node.
2. Node servicing the request writes the new object's data into a repo (short for repository) chunk.
3. On successful write to disk a PR transaction occurs to enter name and chunk location.
4. The partition owner records the transaction in journal logs.
5. Once the transaction has been recorded in the journal logs, an acknowledgement is sent to the client.

Figure 10 shows an example the data flow for a read for hard disk drive architecture like Gen2 and EX300, EX500 and EX3000.

1. A read object request is sent from client to Node 1.

2. Node 1 utilizes a hash function using the object name to determine which node is the partition owner of the logical table where this object information resides. In this example, Node 2 is owner and thus Node 2 will do a lookup in the logical tables to get location of chunk. In some cases, the lookup can occur on two different nodes, for instance when the location is not cached in logical tables of Node 2.
3. From the previous step, location of chunk is provided to Node 1 who will then issue a byte offset read request to the node that holds the data, Node 3 in this example, and will send data to Node 1.
4. Node 1 sends data to requesting client.

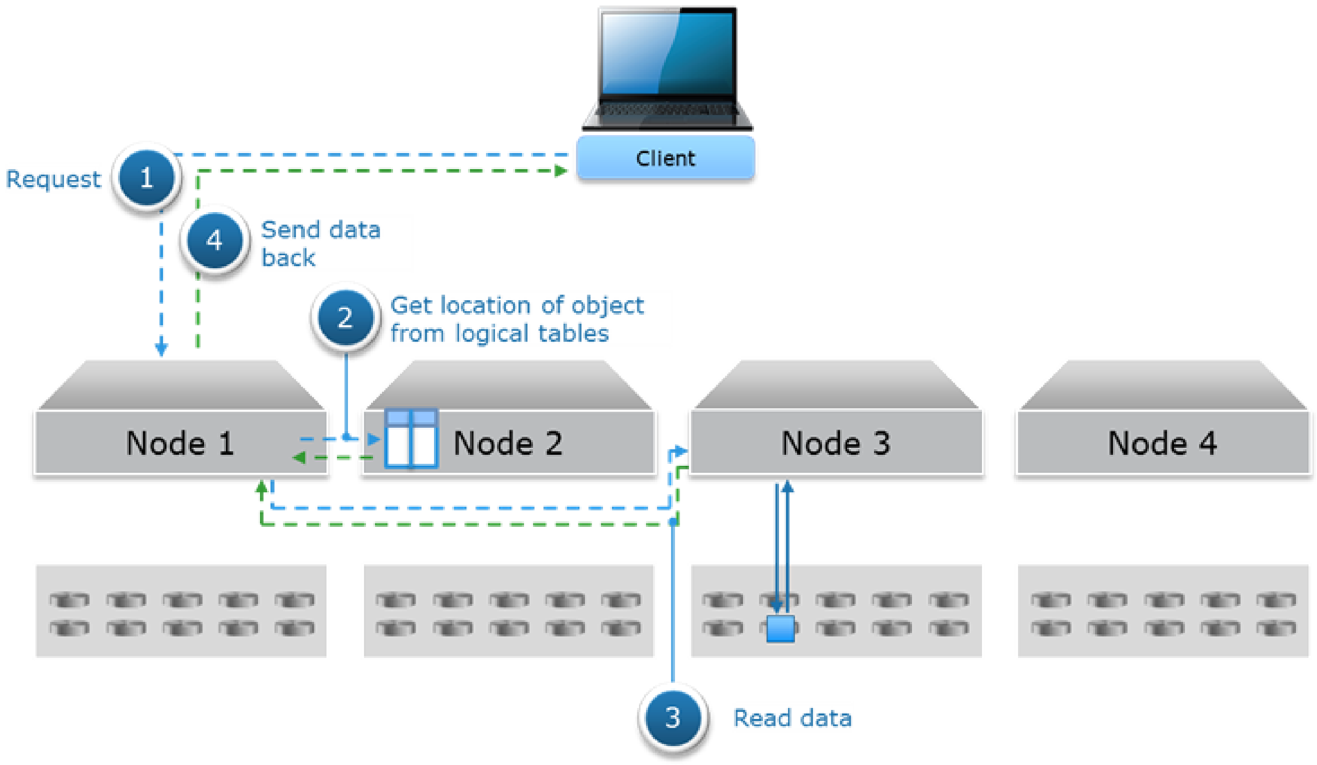


Figure 10. Read data flow for hard disk drive architecture

Figure 11 shows an example the data flow for a read for all flash architecture like EXF900.

1. A read object request is sent from client to Node 1.
2. Node 1 utilizes a hash function using the object name to determine which node is the partition owner of the logical table where this object information resides. In this example, Node 2 is owner and thus Node 2 will do a lookup in the logical tables to get location of chunk. In some cases, the lookup can occur on two different nodes, for instance when the location is not cached in logical tables of Node 2.
3. From the previous step, location of chunk is provided to Node 1 who will then read the data from Node 3 directly.
4. Node 1 sends data to requesting client.

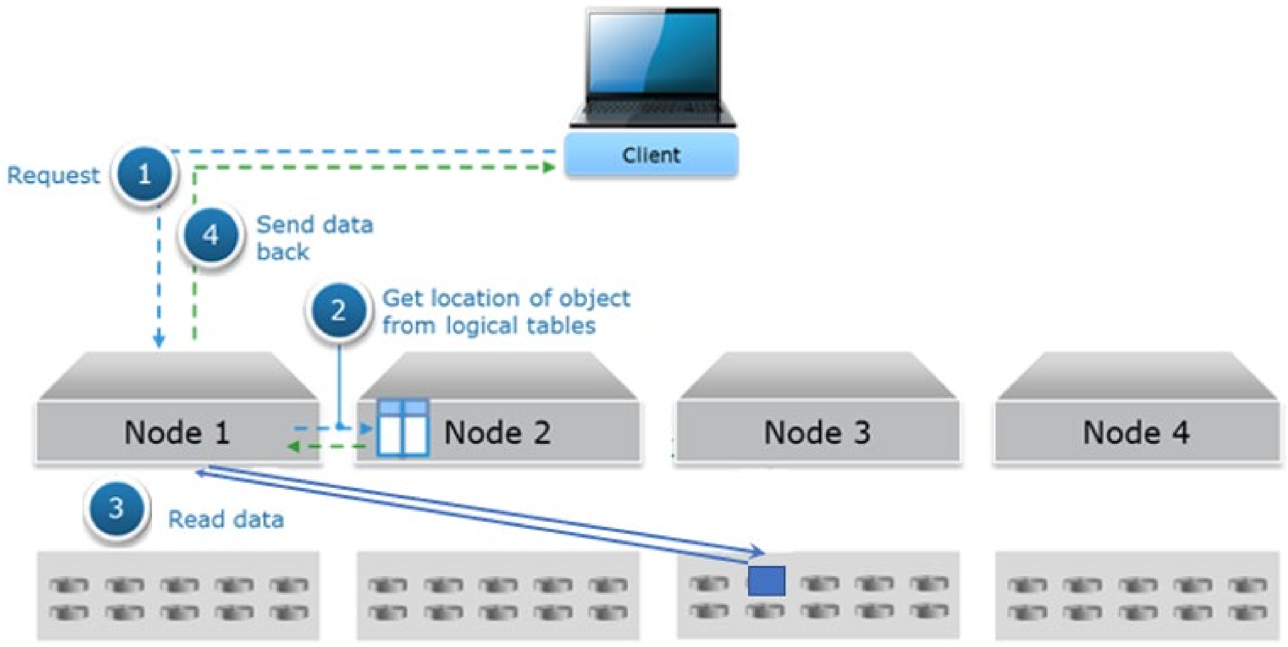


Figure 11. Read data flow for all-flash architecture

Note: In the all-flash architecture like EXF900, each node can read data from other node directly, other than the hard disk drive architecture that each node can only read the data store in themselves.

Write optimizations for file size

For smaller writes to storage ECS uses a method called *box-carting* to minimize impact to performance. Box-carting aggregates multiple smaller writes of 2MB or less in memory and writes them in a single disk operation. Box-carting limits the number of roundtrips to disk required process individual writes.

For writes of larger objects, nodes within ECS can process write requests for the same object simultaneously and take advantage simultaneous writes across multiple spindles in the ECS cluster. Thus, ECS can ingest and store small and large objects efficiently.

Space reclamation

Writing chunks in an append-only manner means that data is added or updated by first keeping the original written data in place and secondly by creating net new chunk segments which may or may not be included in the chunk container of the original object. The benefit of append-only data modification is an active/active data access model which is not hindered by file-locking issues of traditional filesystems. This being the case, as objects are updated or deleted, data in chunks becomes no longer referenced or needed. Two garbage collection methods used by ECS to reclaim space from discarded full chunks, or chunks containing a mixture of deleted and non-deleted object fragments which are no longer referenced, are:

- **Normal Garbage Collection** - When an entire chunk is garbage, reclaim space.

- **Partial Garbage Collection by Merge** - When a chunk is 2/3 garbage, reclaim the chunk by merging the valid parts of with other partially filled chunks to a new chunk, reclaim space.

Garbage collection has also been applied to the ECS CAS data services access API to clean up orphan blobs. Orphan blobs, which are unreferenced blobs identified in the CAS data stored on ECS, will be eligible for space reclamation via normal garbage collection methods.

SSD metadata caching

ECS metadata is stored in B-trees. Each B-tree may have entries in memory, journal transactions and on disk. For the system to have a complete picture of a particular B-tree all three locations are queried which often includes multiple look ups to disk.

To minimize latency for metadata lookups, an optional SSD-based cache mechanism has been implemented in ECS 3.5. The cache holds recently accessed B-tree pages. This means read operations on the latest B-trees will always hit the SSD-based cache and avoids trips to spinning disks.

Here are some highlights for the new SSD metadata caching feature:

- Improved system-wide read latency and TPS (Transactions Per Second) for small files
- One 960GB flash drive per node
- Net new nodes from manufacturing include the SSD drive as an option
- Existing field nodes—Gen3 and Gen2—can be upgraded via upgrade kits and self-service installation
- SSD drives can be added while ECS is online
- Improvement for small file analytics workloads which require fast reads of large data sets
- All nodes in a VDC must have SSDs to enable this feature

The ECS fabric detects when an SSD kit has been installed. This triggers the system to automatically initialize and begin using the new drive. The following figure shows SSD cache enabled:

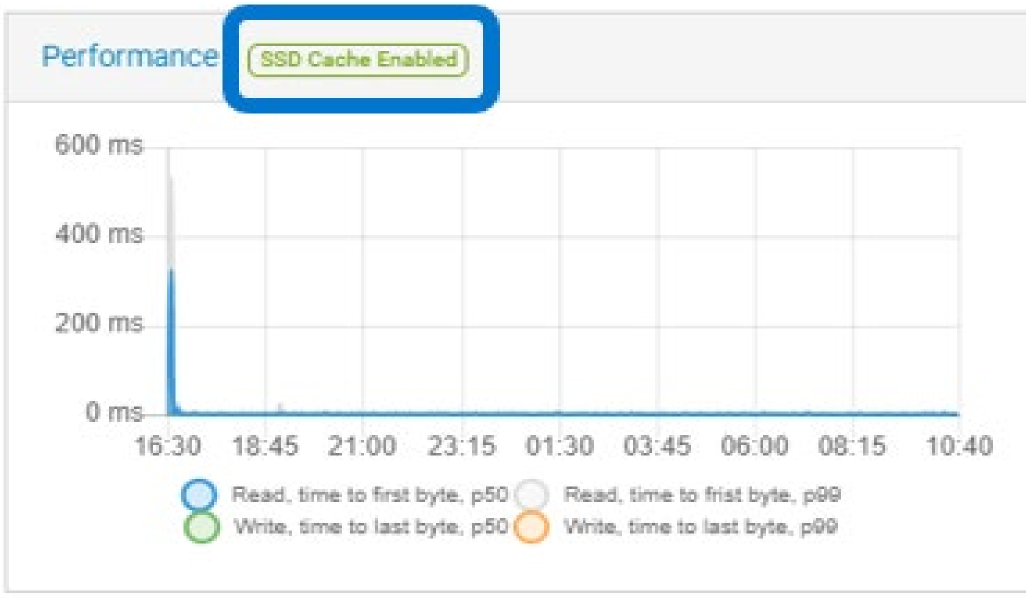

Performance
SSD Cache Enabled
600 ms
400 ms
200 ms
0 ms
16:30
18:45
21:00
23:15
01:30
03:45
06:00
08:15
10:40
Read, time to first byte, p50
Read, time to frist byte, p99
Write, time to last byte, p50
Write, time to last byte, p99

Figure 12. SSD cache enabled

SSD metadata caching improves small reads and bucket listing. As we tested in our lab, the listing performance improves 50% with 10MB objects. The read performance improves 35% with 10KB objects and 70% with 100KB objects.

Cloud DVR

ECS supports cloud Digital Video Recording (DVR) feature which addresses a legal copyright requirement for cable and satellite companies. The requirement is every unit of recording mapped to an object on ECS needs to be copied a predetermined number of times. The predetermined number of copies are known as fanout. The pre-determined number of copies (fan-out) is not really a requirement for redundancy or performance gain, but it's more of a legal copyright requirement for cable and satellite companies. ECS supports:

- Create fanout number of copies of object created in ECS
- Allow read of specific copy
- Allow delete of specific copy
- Allow delete of all copies
- Allow copy of specific copy
- Allow listing of copies
- Allow bucket listing of fanout objects

The cloud DVR feature can be enabled through Service Console. For the first time, you must enable the cloud DVR feature using Service Console. After enabling cloud DVR, by default, for all the new nodes cloud DVR is enabled.

Run the following command in Service Console to enable the cloud DVR feature:

```
service-console run Enable_CloudDVR
```

The cloud DVR feature support APIs and you can refer to the *ECS Data Access Guide* for more details.

Intermixing all flash array (AFA) and hard disk drive (HDD) nodes

With ECS 3.7, you can create a VDC that consists of both EXF 900 nodes and any* of the ECS HDD based nodes. This allows for a single point of management for all node types. EXF 900 and the HDD nodes must be their own Storage Pools. You have the flexibility to add different media type nodes to your existing VDC or create a new VDC.

- Fresh install with AFA/HDD intermixes mode
- Extend node and add storage pool to support AFA/HDD intermix mode
  - Extend HDD only VDC to co-existing VDC
  - Extend AFA only VDC to co-existing VDC

Note: * means EX300, EX500, EX3000, Gen2 U-series

ECS 3.7 does not support replication between AFA and HDD node, however you can leverage ECS Sync to move or copy data between the two storage pools.

Note: ECS doesn't support Replication Group (RG) containing both HDD and AFA storage pools.

In a Geo replication scenario, the replication group can only consist of the same disk type. It is better to add a new http header in Geo messages to indicate the type of request for app developer.

```
x-emc-drive-technology    #can be HDD or AFA
```

On the client side, try to send the request with x-emc-drive-technology to the endpoints with the same drive technology.

In an HDD&AFA intermix setup, it may impact performance negatively on front-end requests, if the customer deploys a load balancer in front of ECS but does not configure it properly.

The following are best practices for the load balancer configuration:

- If each load balancer is in charge of a specific type of storage pool, the client visits AFA bucket via AFA load balancer and HDD bucket via HDD load balancer.
- When a load balancer has multiple frontends, each frontend can forward requests to a specified type of storage pool.
- When a load balancer has one frontend, it can forward requests to a different type of storage pool based on the data in the request.

Note: ECS only supports S3 and forwards requests to the http port in an intermix node situation.

S3 select

S3 select, launched in version 3.7, enables applications to retrieve only a subset of data from an object by using simple SQL expressions. By using s3 select to retrieve only the data needed by your application, you can achieve drastic performance increases and network bandwidth savings.

Using the example of a 2GB csv object, without s3 select an application would have to download the entire 2GB object and then do the processing on that data. With s3 select, the application issues SQL select commands and potentially gets only a small subset of that data.

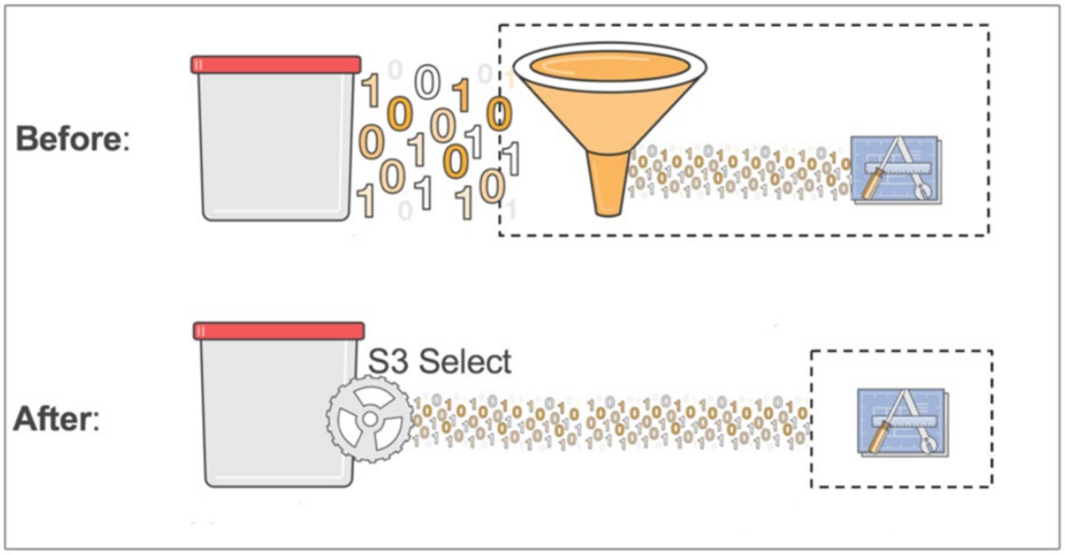


Figure 13. S3 select

S3 select can be used for objects in the csv, json, and parquet formats. It supports querying gzip/bzip2 compressed objects of these three file types.

S3 select is commonly used by query engines, like presto. A connector in presto can determine if a particular query can be sent directly to storage. For example, s3 select pushdown.

AWS compliant S3 performs partial reads of an Object. This offloads query and sort to ECS rather than using compute resources. This may provide a performance benefit for use cases where network bandwidth and or compute resources are a bottleneck.

Note: S3 select is not enabled by default. It is suggested to enable it with 192GB of memory on each node. Please contact your Dell support team if you need to enable this feature.

Fabric

The Fabric layer provides clustering, system health, software management, configuration management, upgrade capabilities and alerting. It is responsible for keeping services running and managing resources such as disks, containers and the network. It tracks and reacts to environment changes such as failure detection and provides alerts related to system health. The Fabric layer has the following components:

- **Node Agent** - Manages host resources (disks, network, containers, etc.) and system processes.
- **Lifecyle Manager** - Application lifecycle management which involves starting services, recovery, notification and failure detection.
- **Persistence Manager** - Coordinates and synchronizes the ECS distributed environment.

- **Registry** - Docker image store for ECS software.
- **Event Library** - Holds the set of events occurring on the system.
- **Hardware Manager** - Provides status, event information and provisioning of the hardware layer to higher level services. These services have been integrated to support commodity hardware.

Node agent

The node agent is a lightweight agent written in Java that runs natively on all ECS nodes. Its main duties include managing and controlling host resources (Docker containers, disks, the firewall, the network) and monitoring system processes. Examples of management include formatting and mounting disks, opening required ports, ensuring all processes are running, and determining public and private network interfaces. It has an event stream that provides ordered events to a lifecycle manager to indicate events occurring on the system. A Fabric CLI is useful to diagnose issues and look at overall system state.

Lifecycle manager

The lifecycle manager runs on a subset of three or five nodes and manages the lifecycle of applications running on nodes. Each lifecycle manager is responsible for tracking several nodes. Its main goal is to manage the entire lifecycle of the ECS application from boot to deployment, including failure detection, recovery, notification and migration. It looks at the node agent streams and drives the agent to handle the situation. When a node is down, it responds to failures or inconsistencies in the state of the node by restoring the system to a known good state. If a lifecycle manager instance is down, another one takes its place.

Registry

The registry contains the ECS Docker images used during installation, upgrade and node replacement. A Docker container called *fabric-registry* runs on one node within the ECS rack and holds the repository of ECS Docker images and information required for installations and upgrades. Although the registry is available on one node at a time, all Docker images are locally cached on every node, so any may serve the registry.

Event library

The event library is used within the Fabric layer to expose the lifecycle and node agent event streams. Events generated by the system are persisted onto shared memory and disk to provide historical information on the state and health of the ECS system. These ordered event streams can be used to restore the system to a specific state by replaying the ordered events stored. Some examples of events include node events such as started, stopped or degraded.

Hardware manager

The hardware manager is integrated to the Fabric Agent to support industry standard hardware. Its main purpose is to provide hardware specific status and event information, and provisioning of the hardware layer to higher level services within ECS.

Infrastructure

ECS appliance nodes currently run SUSE Linux Enterprise Server 12 for the infrastructure. For ECS software deployed on custom industry standard hardware the operating system can also be RedHat Enterprise Linux or CoreOS. Custom deployments are done via a formal request and validation process. Docker is installed on the infrastructure to deploy the encapsulated ECS layers. ECS software is written in Java so the Java Virtual Machine is installed as part of the infrastructure.

Docker

ECS runs on top of the operating system as a Java application and is encapsulated within several Docker containers. The containers are isolated but share the underlying operating system resources and hardware. Some parts of ECS software run on all nodes and some run on one or some nodes. The components running within a Docker container include:

- **object-main** - Contains the resources and processes relating to the data services, storage engine and the portal and provisioning services. Runs on every node in ECS.
- **fabric-lifecycle** - Contains the processes, information and resources required for system-level monitoring, configuration management and health management. An odd number of fabric-lifecycle instances will always be running. For example, there will be three instances running on a four-node system and five instances for an eight-node system. **fabric-zookeeper** - Centralized service for coordinating and synchronizing distributed processes, configuration information, groups and naming services. It is referred to as the persistence manager and runs on odd number of nodes, for instance, five in an eight-node system.
- **fabric-registry** - Registry of the ECS Docker images. Only one instance runs per ECS rack.

There are other processes and tools that run outside of a Docker container namely the Fabric node agent and hardware abstraction layer tools. The following figure provides an example of how ECS containers can be run on an eight-node deployment:

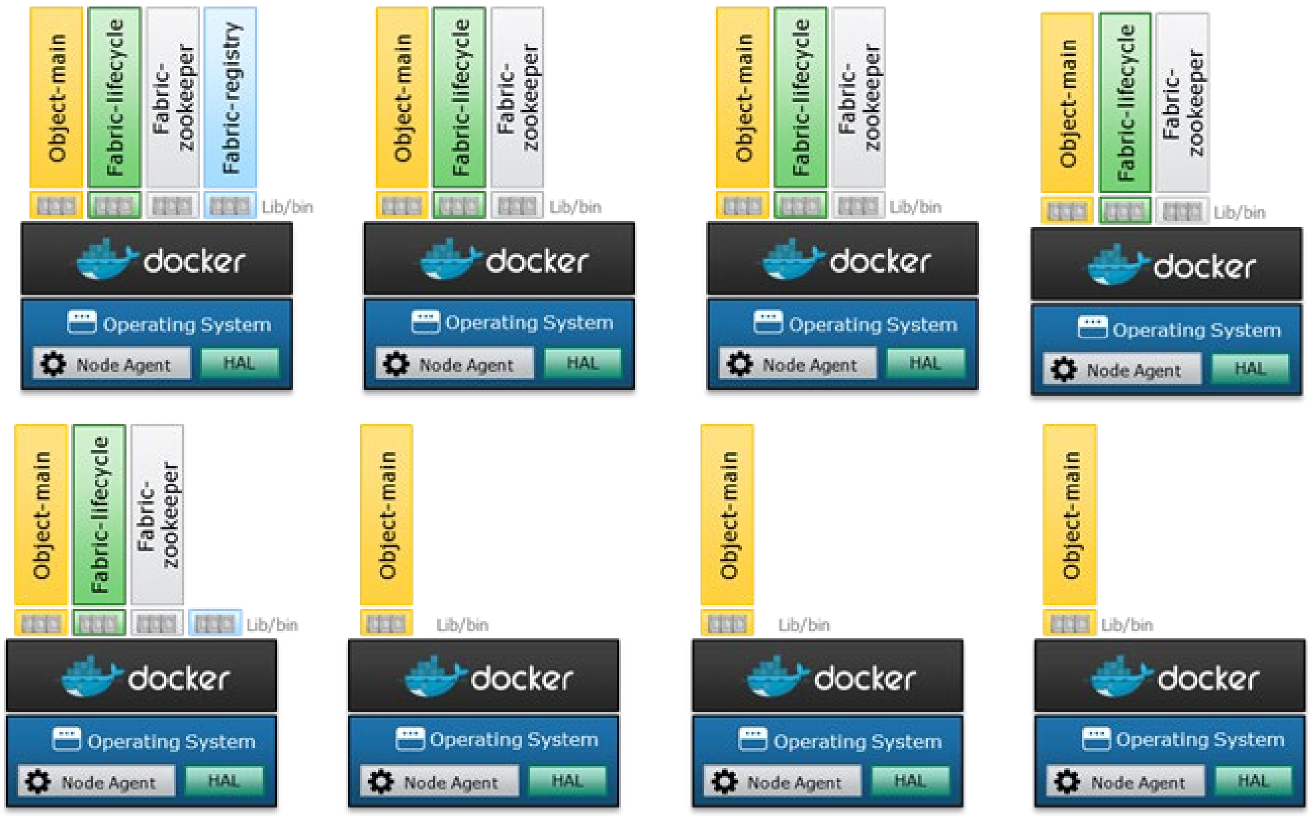
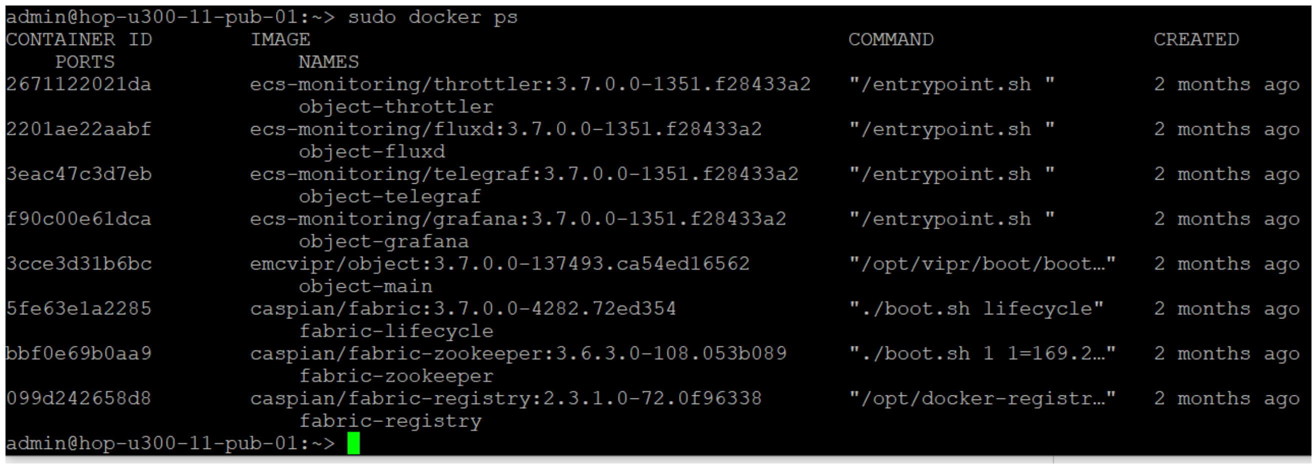


Figure 14. Docker containers and agents on eight node deployment example

The following figure shows command line output of the *docker ps* command on a node which shows the four containers used by ECS inside Docker. A listing is shown with all the object-related services available on the system.



```
admin@hop-u300-11-pub-01:~> sudo docker ps
CONTAINER ID        IMAGE                                              COMMAND                  CREATED
    PORTS               NAMES
2671122021da        ecs-monitoring/throttler:3.7.0.0-1351.f28433a2    "/entrypoint.sh "        2 months ago
                        object-throttler
2201ae22aabf        ecs-monitoring/fluxd:3.7.0.0-1351.f28433a2        "/entrypoint.sh "        2 months ago
                        object-fluxd
3eac47c3d7eb        ecs-monitoring/telegraf:3.7.0.0-1351.f28433a2     "/entrypoint.sh "        2 months ago
                        object-telegraf
f90c00e61dca        ecs-monitoring/grafana:3.7.0.0-1351.f28433a2      "/entrypoint.sh "        2 months ago
                        object-grafana
3cce3d31b6bc        emcvipr/object:3.7.0.0-137493.ca54ed16562         "/opt/vipr/boot/boot…"   2 months ago
                        object-main
5fe63e1a2285        caspian/fabric:3.7.0.0-4282.72ed354               "./boot.sh lifecycle"    2 months ago
                        fabric-lifecycle
bbf0e69b0aa9        caspian/fabric-zookeeper:3.6.3.0-108.053b089      "./boot.sh 1 1=169.2…"   2 months ago
                        fabric-zookeeper
099d242658d8        caspian/fabric-registry:2.3.1.0-72.0f96338        "/opt/docker-registr…"   2 months ago
                        fabric-registry
admin@hop-u300-11-pub-01:~>
```

Figure 15. Processes, resources, tools, and binaries in object-main container

Appliance hardware models

Introduction

Flexible entry points enable ECS to rapidly scale to petabytes and exabytes of data. With minimal business impact, an ECS solution can scale linearly in both capacity and performance by adding nodes and disks.

ECS appliance hardware models are characterized by hardware generation. The third-generation appliance series, known as the Gen3 or EX-Series, include three hardware models. A high-level overview of the EX-Series is provided in this section. For complete details refer to the *ECS EX-Series Hardware Guide*.

Information on the first and second generation ECS appliance hardware is available in the *Dell ECS D- and U-Series Hardware Guide*.

EX series

EX series appliance models are based on standard Dell servers and switches. The offerings in the series are:

- **EX300** - The EX300 has a starting raw capacity of 60 TB. They are the perfect storage platform for cloud-native apps and customer digital transformation initiatives. EX300 are ideal for modernizing Centera deployments. Most importantly, the EX300 can scale cost-effectively to larger capacities. It provides 12 drives per node and 1TB, 2TB, 4TB, 8TB, 16TB disk options (all same in the node)
- **EX500** - The EX500 is the latest edition appliance which aims to provide economy with density. With options for 12 or 24 drives, 2TB, 4TB, 8TB, 12TB and 16TB disk options (all same in the node). Cluster range from 480TB to 6.1PB per rack. This series provides a versatile option for midsize enterprises looking to support modern application and/or deep archive use cases.
- **EX3000** - The EX3000 has a maximum capacity of 11.5 PB of raw storage per rack, 30 to 90 drives per node, 12TB or 16TB disks and can grow into exabytes across several sites, providing a deep and scalable data center solution that is ideal for workloads with larger data footprints. These nodes are available in two different configurations known as EX3000S and EX3000D. The EX3000S is a single-node, and the EX3000D is a dual-node chassis. These high-density nodes are hot disk-swappable. They start with a minimum of thirty disks per node. Thirty drives per ECS node is the point around which the gains in performance by adding more drives diminishes. With thirty or more drives in each node as a minimum, the performance expectations are similar across every EX3000 node regardless of drive count.
- **EXF900** – The EXF900 is an all flash object storage solution of hyper-converged nodes for low latency and high IOPs ECS deployments. With options for 12 or 24 drives, 3.84TB, 7.68TB, and 15.36TB NVMe SSD drive options. This platform starts at 230TB RAW minimum configuration and scales to 2.8PB RAW per rack. The following figure shows a node of EXF900:

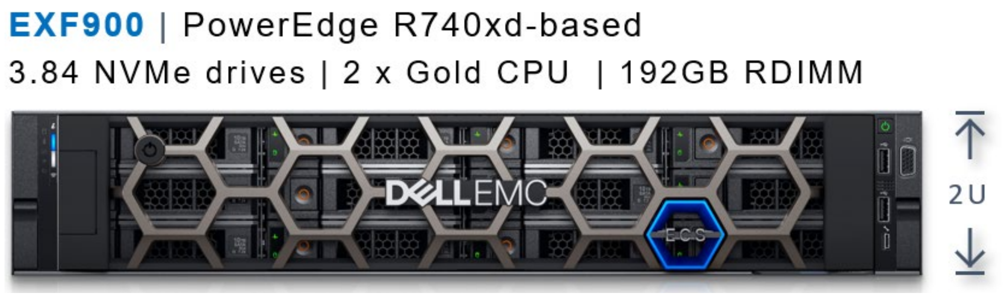

EXF900 | PowerEdge R740xd-based
3.84 NVMe drives | 2 x Gold CPU | 192GB RDIMM
DELLEMC
ECS
2U

Figure 16. EXF900 node

Note: SSD Read Cache feature does not apply for EXF900; Cloud DVR is not supported on EXF900; Tech Refresh is not supported with EXF900.

The EX-Series starting capacity options allow customers to begin an ECS deployment with only the capacity needed, and to easily grow as needs change in the future. Refer to the *ECS Appliance Specification Sheet* for more details on the EX-Series appliances which also details the previous Gen2 U- and D- series appliances.

Post deployment updates to EX-Series nodes are not supported. These include:

- Changing the CPU.
- Adjusting memory capacity.
- Upgrading hard drive size.

Appliance networking

Starting with the release of the EX-Series appliances, a redundant pair of dedicated back-end management switches are used. By moving to new appliance switch gear, ECS is now able to adopt a front- and back-end switching mode of configuration.

The EX300, EX500, and EX3000 appliances all use the Dell S5148F for the front-end pair of switches and for the pair of back-end switches. The EXF900 appliance use the Dell S5248F for the front-end pair of switches and for the pair of back-end switches and S5232F for the aggregation back-end switch. Note that customers have the option of using their own front-end switches instead of the Dell switches.

S5148F – front-end public switches

Two optional Dell S5148F 25GbE 1U ethernet switches can be obtained for network connection, or the customer can provide their own 10 or 25GbE HA pair for the front-end connectivity. The public switches are often referred to as *hare* and *rabbit* or just the front-end.

Caution: It is required to have connections from the customer's network to both front-end switches (rabbit and hare) in order to maintain the high availability architecture of the ECS appliance. If the customer chooses not to connect to their network in the required HA manner, there is no guarantee of high data availability for the use of this product.

These switches provide 48 ports of 25GbE SFP28 and 6 ports of 100GbE QSFP28. More details of these two port types are:

- SFP28 is an enhanced version of SFP+

- SFP+ supports up to 16Gb/s, SFP28 supports up to 28Gb/s
- Same form factor
- Backwards compatible to SFP+ modules

- QSFP28 is an enhanced version of QSFP+
  - QSFP+ supports up to 4 lanes of 16Gb/s, QSFP28 supports up to 4 lanes of 28Gb/s
    - QSFP+ aggregated lanes to obtain 40Gb/s Ethernet
    - QSFP28 aggregated lanes to obtain 100Gb/s Ethernet
  - Same form factor
  - Backwards compatible to QSFP+ modules
  - Can be broken out into 4 individual lanes of SFP28

Note: Two 100GbE LAG cables are provide with Dell S5148F 25GbE public switches. Organizations providing their own public switches must supply required LAG, SFPs or external connection cables.

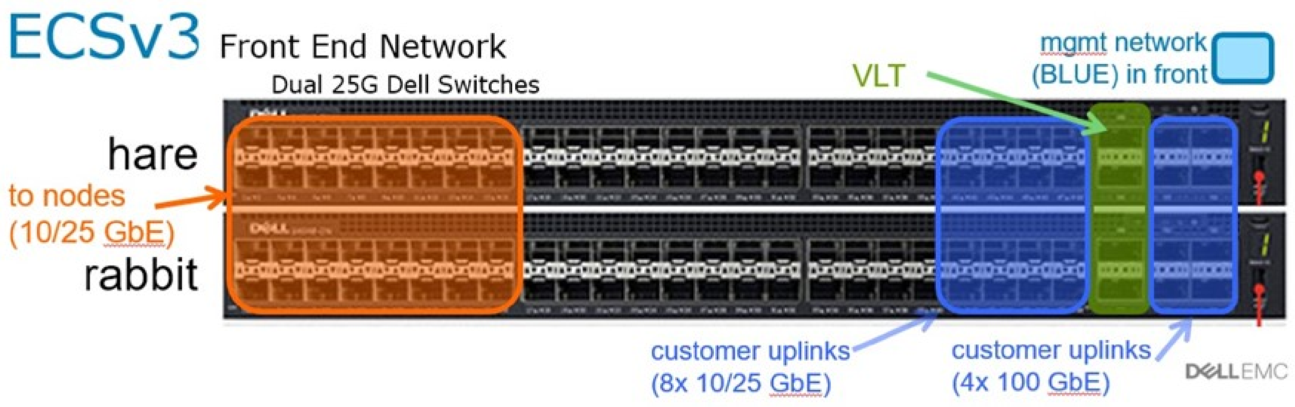

ECSv3
Front End Network
Dual 25G Dell Switches
VLT
mgmt network (BLUE) in front
hare
to nodes (10/25 GbE)
rabbit
customer uplinks (8x 10/25 GbE)
customer uplinks (4x 100 GbE)
DELLEMC

Figure 17. Front-end network switch port designation and usage

The preceding figure provides a visual representation of how ports are intended to be used to enable ECS node traffic as well as customer uplink ports. This is standard across all implementations.

S5148F – back-end private switches

Both required Dell S5148F 25GbE 1U Ethernet switches with 48 25GbE SFP ports and 6 100GbE uplink ports are included in each ECS rack. These are often referred to as *fox* and *hound* or back-end switches and are responsible for the management network. In future ECS releases, the back-end switches will also provide network separation for replication traffic. The main purpose of the private network is for remote management and console, PXE booting for the install manager and to enable rack and cluster-wide management and provisioning. The following figure shows a front view of two Dell 25GbE switches.

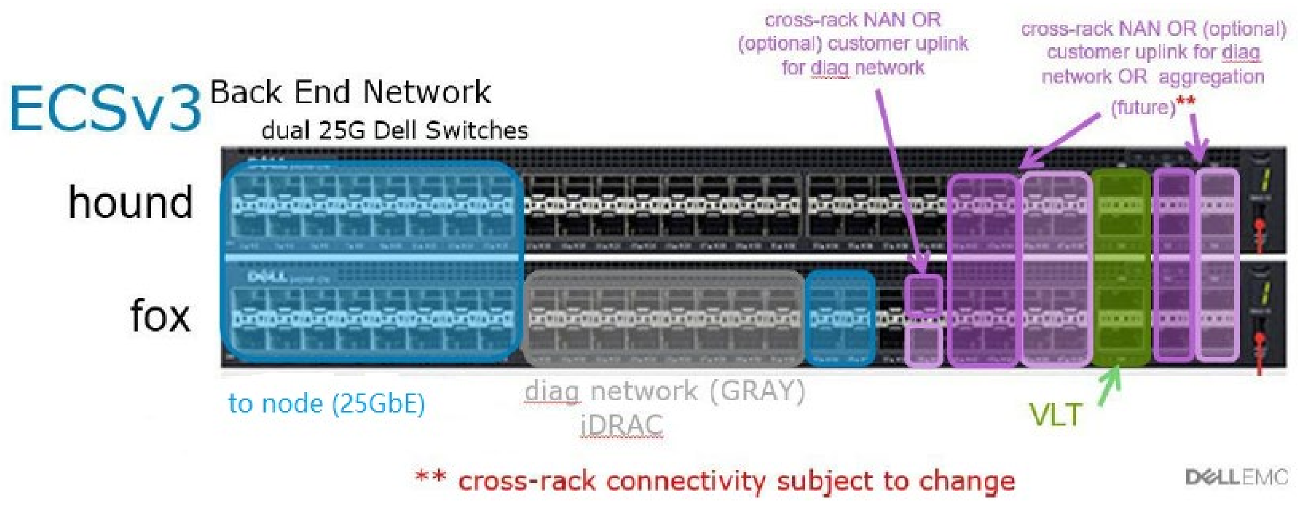

ECSv3
Back End Network
dual 25G Dell Switches
hound
fox
cross-rack NAN OR (optional) customer uplink for diag network
cross-rack NAN OR (optional) customer uplink for diag network OR aggregation (future)**
to node (25GbE)
diag network (GRAY)
iDRAC
VLT
** cross-rack connectivity subject to change
DELL EMC

Figure 18. Back-end network switch port designation and usage

The preceding figure provides a visual representation of how ports are intended to be used to enable ECS management traffic and diagnostic ports. These port allocations are standard across all implementations. Possible future use ports are noted in purple; however, this usage is subject to change in the future.

S5248F – front-end public switches

Dell offers an optional HA pair of front-end 25 GbE S5248F switches for customer network connection to the rack. It has two 200GbE (QSFP28-DD) virtual link trunking (VLT) cables per HA pair. These switches are called the Hare and the Rabbit switches. The following figure shows a visual representation of how ports are intended to be used to enable ECS node traffic as well as customer uplink ports.

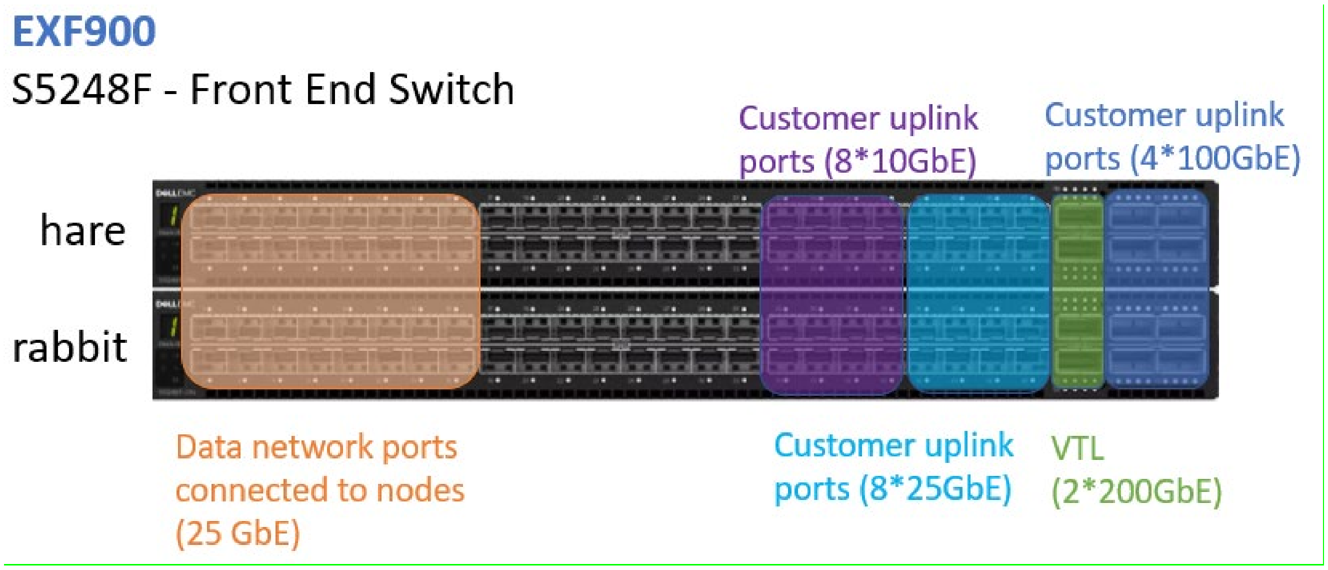

EXF900
S5248F - Front End Switch
Customer uplink ports (8*10GbE)
Customer uplink ports (4*100GbE)
hare
rabbit
Data network ports connected to nodes (25 GbE)
Customer uplink ports (8*25GbE)
VTL (2*200GbE)

Figure 19. Front-end network switch port designation and usage

S5248F – back-end private switches

Dell provides two 25 GbE S5248F back-end switches with two 200GbE (QSFP28-DD) VLT cables. These switches are referred to as the Hound and Fox switches. All iDRAC cables from nodes and all front-end switch management cable connections route to the Fox switch. The following figure provides a visual representation of how ports are intended to be used to enable ECS management traffic and diagnostic ports. These port allocations are standard across all implementations.

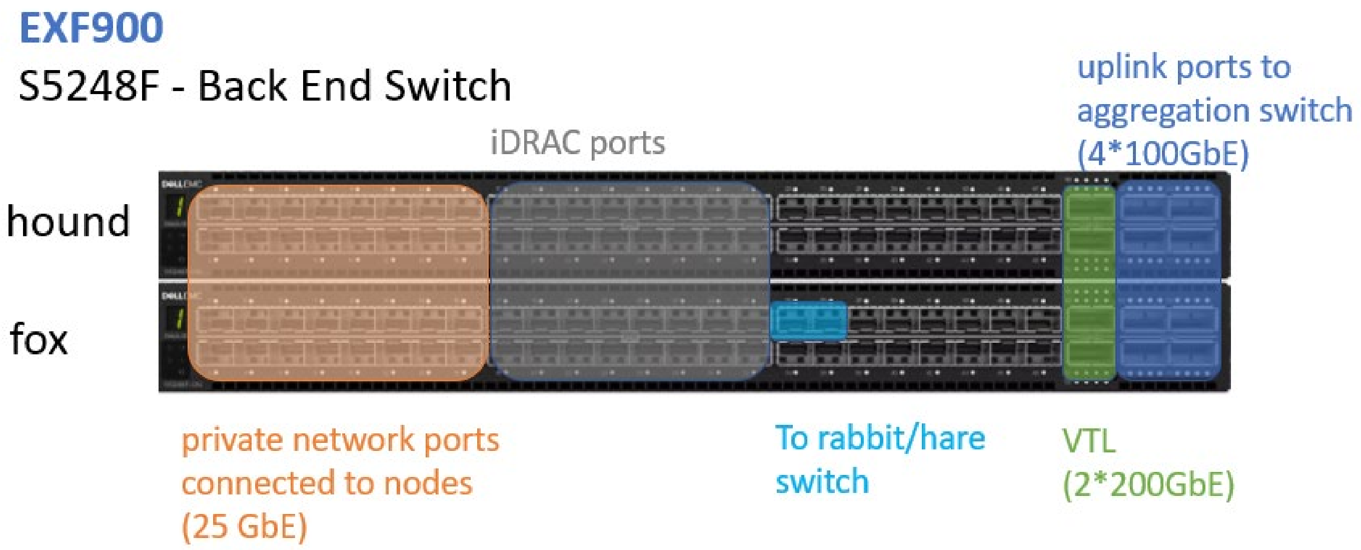

EXF900
S5248F - Back End Switch
iDRAC ports
uplink ports to aggregation switch (4*100GbE)
hound
fox
private network ports connected to nodes (25 GbE)
To rabbit/hare switch
VTL (2*200GbE)

Figure 20. Back-end network switch port designation and usage

S5232 – aggregation switch

Dell provides two 100GbE S5232F back-end aggregation switches (AGG1 and AGG2) with four 100GbE VLT cables. These switches are referred to as the Falcon and Eagle switches. In the following figure, all labeled ports indicate the port designations. This configuration allows to connect to 7 racks of EXF900 nodes.

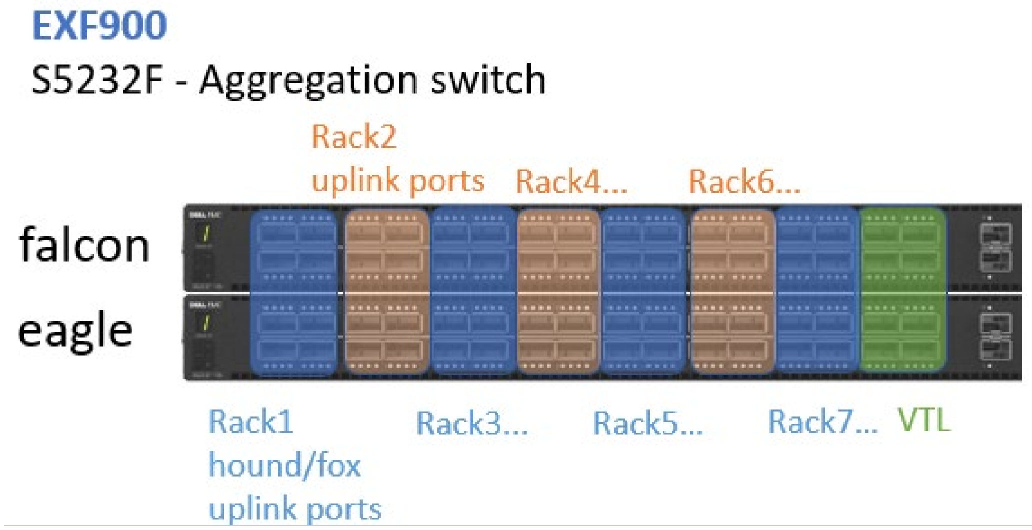


Figure 21. Aggregation switch port designation and usage

For more information about the networking and cabling, please refer to the *ECS EX Series Hardware Guide.*

Network separation

ECS supports separating different types of network traffic for security and performance isolation. The types of traffic that can be separated include:

- Management
- Replication
- Data

There is a mode of operation called the *network separation mode*. In this mode each node can be configured at the operating system level with up to three IP addresses, or logical networks, for each of the different types of traffic. This feature has been designed to provide the flexibility of either creating three separate logical networks for management, replication and data, or combining them to either create two logical networks, for instance management and replication traffic is in one logical network and data traffic in another logical network. A second logical data network for CAS-only traffic can be configured, allowing separation of CAS traffic from other types of data traffic like S3.

ECS implementation of network separation requires each logical network traffic to be associated with services and ports. For instance, the ECS portal services communicate via ports 80 or 443, so these ports and services will be tied to the management logical network. A second data network can be configured; however, it is for CAS traffic only. The following table highlights the services fixed to a type of logical network. For a complete list of services associated with ports, refer to the latest *ECS Security Configuration Guide*.

Table 5. Services to logical network mapping

Services	Logical network	Identifier
WebUI and API, SSH, DNS, NTP, AD, SMTP	Management	public.mgmt
Client data	Data	public.data
	CAS-only data	public.data2
Replication data	Replication	public.repl
SRS (Dell Secure Remote Services)	Based on network SRS Gateway is attached	public.data or public.mgmt

Note: ECS 3.6 allows S3 data access on both data (default) and data2 network (though S3 is not enabled by default on data2). To enable S3 data access on data2 network, public.data is required and please contact ECS remote support.

Network separation is achievable logically using different IP addresses, virtually using different VLANs or physically using different cables. The *setrackinfo* command is used to configure IP addresses and VLANs. Switch-level or client-side VLAN configuration is the customer's responsibility. For physical network separation, customers need to submit a Request for Product Qualification (RPQ) by contacting Dell Global Business Service. For more information on network separation refer to the *ECS Networking and Best Practices* white paper that provides a high-level view of network separation.

Security

Introduction

ECS security is implemented at the administration, transport and data levels. User and administrator authentication are achieved via Active Directory, LDAP methods, Keystone or directly within the ECS portal. Data level security is done via HTTPS for data in motion and/or server-side encryption for data-at-rest.

Authentication

ECS supports Active Directory, LDAP, Keystone and IAM authentication methods to provide access to manage and configure ECS; however, limitations exist as shown in the following table. For more information on security, see the latest *ECS Security Configuration Guide*.

Table 6. Supported authentication methods

Authentication method	Supported
Active Directory	• AD group support for management users • AD group support for object user self-provisioning methods using self-service keys via API • Multi-domain is supported
LDAP	• Management users may individually authenticate via LDAP • LDAP Groups are NOT supported for management users • LDAP is supported for Object Users (Self-service keys via API) • Multi-domain is supported.
Keystone	• RBAC policies not yet supported. • No support for un-scoped tokens • No support of multiple Keystone Servers per ECS system
IAM	• Delivers identity federation and single sign-on (SSO) via SAML 2.0 standards • Available through the S3 protocol only

Data services authentication

Object access using RESTful APIs is secured over HTTPS (TLS v1.2). Incoming requests are authenticated using defined methods such as Hash-based Message Authentication Code (HBAC), Kerberos or token authentication. The following table presents the different methods used for each protocol.

Table 7. Data services authentication

Protocol		Authentication methods
Object	S3	V2 (HMAC-SHA1), V4 (HMAC-SHA256)
	Swift	Token - Keystone v2 and v3 (scoped, UUID, PKI tokens), SWAuth v1
	Atmos	HMAC-SHA1
	CAS	Secret Key PEA file

Protocol		Authentication methods
File	HDFS	Kerberos
	NFS	Kerberos, AUTH_SYS

Data-at-rest encryption (D@RE)

Compliance requirements often mandate the use of encryption to protect data written on disks. In ECS encryption can be enabled at the namespace and bucket levels. Key features of ECS D@RE include:

- Native low touch encryption at rest - easily enabled, simple configuration
- CIPHERs (AES-256 CTR) used
- RSA Public Key Encryption with 2048bit length
- External Key Management (EKM) cluster-level support:
    - Gemalto SafeNet
    - IBM Security Key Lifecycle Manager
- Key rotation
- S3 encryption semantics support using HTTP headers such as *x-amz-server-side-encryption*
- FIPS 140-2 compliance with US government cryptographic security standards

Note: FIPS 140-2 mode enforces the use of approved-only algorithms within D@RE; FIPS 140-2 compliance is only for the D@RE module, not the entire ECS product.

ECS uses a key hierarchy to encrypt and decrypt data. The native key manager stores a private key common to all nodes to decrypt the primary key. With EKM configuration, the primary key is provided by the EKM. EKM provided keys reside in memory only on ECS. They are never stored in persistent storage within ECS.

In a geo-replicated environment, when a new ECS system joins an existing federation, the primary key is extracted using the public-private key of the existing system and encrypted using the new public-private key pair generated from the new system that joined the federation. From this point on, the primary key is global and known to both systems within the federation. When using EKM, all federated systems retrieve the primary key from the key management system.

Key rotation

ECS supports changing encryption keys. This can be done periodically to limit the amount of data protected by a specific set of Key Encryption Keys (KEK) or in response to a potential leak or compromise. A Rotation KEK Record is used in combination with other parent keys to create virtual wrapping keys for protecting Data Encryption Keys (DEK) and namespace KEKs.

Rotation keys are natively generated or supplied and maintained by an EKM. ECS uses the current Rotation Key to create virtual wrapping keys to protect any DEK or KEK regardless of whether key management is done natively or externally.

During writes ECS wraps the randomly generated DEK using a virtual wrapping key created using the bucket and active rotation key.

As part of the rotation of keys, ECS re-wraps all namespace KEK records with a new virtual primary KEK created from new rotation key, the associated secret context and the active primary key. This is done to protect access to data protected by the previous rotation keys.

Using an EKM affects the read/write path for encrypted objects. Rotation of keys allow for extra data protection by using virtual wrapping keys for DEKs and namespace KEKs. The virtual wrapping keys are not persisted and are derived from two independent hierarchies of persisted keys. With the use of EKM, then the rotation key is not stored in ECS and adds further to the security of data. We mainly add new KEK records and update active ids but never delete anything.

Additional points to consider regarding key rotation on ECS are:

- The process of rotating keys only changes the current rotation key. The existing primary, namespace and bucket keys do not change during the key rotation process.
- Namespace or bucket level key rotation is not supported, however, the scope of rotation is at cluster level, so all new system encrypted objects will be affected.
- Existing data is not re-encrypted because of rotating keys.
- ECS does not support the rotation of keys during outages.
  - TSO during rotation: Key rotation task suspended until the system comes out of TSO.
  - PSO is in progress. ECS must come out of a PSO before key rotation is enabled. If a PSO happens during rotation, the rotation will fail immediately.
- Bucket encryption is not required to do object encryption via S3.
- Indexed client object metadata utilized as a search key are not encrypted.

See the latest *ECS Security Configuration Guide* for further information on D@RE, EKM and key rotation.

ECS IAM

ECS Identify and Access Management (IAM) enables you to control and secure access to the ECS S3 resources. This functionality ensures that each access request to an ECS resource is identified, authenticated, and authorized. ECS IAM allows admin to add users, roles, and groups. Admin can also restrict the access by adding policies to the ECS IAM entities.

Note: ECS IAM is for use with S3 only. It is not enabled for CAS or filesystem enabled buckets.

ECS IAM consists of the following components

- **Account Management** - enables you to manage IAM identities within each namespace such as users, groups, and roles
- **Access Management** - access is managed by creating policies and attaching them to IAM identities or resources

- **Identity Federation** - identity is be established and authenticated by SAML (Security Assertion Markup Language). After the identity is established you will use the Secure Token Service to obtain temporary credentials that will be used to access the resource
- **Secure Token Service** - enables you to request temporary credentials for cross account access to resources and also for users who are authenticated using SAML authentication from an enterprise identity provider or directory service

By using IAM, you can control who are authenticated and authorized to use ECS resources by creating and managing:

- **Users** - IAM user represents a person or application in the namespace that can interact with ECS resources
- **Groups** - IAM group is a collection of IAM users. Use groups to specify permissions for a collection of IAM users
- **Roles** - IAM Role is an identity that could be assumed by anyone who requires the role. A role is similar to a user, an identity with permission policies that determine what the identity can and cannot do
- **Policies** - IAM policy is a document in JSON format, which defines the permissions for a role. Assign and attach policies to IAM Users, IAM Groups, and IAM Roles.
- **SAML provider**- SAML is an open standard for exchanging authentication and authorization data between an identity provider and a service provider. SAML provider in ECS is used to establish trust between a SAML-compatible Identity Provider (IdP) and ECS

Each ECS system is allotted with an ECS IAM account. This account supports multiple namespaces and has related IAM entities that are defined in its namespace.

- Individual namespaces support in managing account using the ECS IAM entities such as users, roles, and groups.
- Policies, permissions, Access Control List (ACL) that are associated with the ECS IAM entities, and the ECS S3 resources support in managing access to the ECS IAM features.
- ECS IAM supports cross-account access using Security Assertion Markup Language (SAML) and roles.
- ECS IAM supports Amazon Web Services (AWS) Access Key to access IAM and S3 in ECS.

See the latest *ECS Security Guide* for further information on ECS IAM

Object tagging

Object tagging allows categorization of objects by assigning tags to the individual objects. A single object can have multiple tags that are associated with it, enabling multidimensional categorization.

A tag could describe some sort of sensitive information like a health record, or you can tag an object to a certain product that can be categorized as confidential. Tagging is a sub-resource of an object that has a life-cycle integrated with object operations. You can add tags to new objects when you upload them or add tags to existing objects. It is acceptable to use tags to label objects containing confidential data, such as personally identifiable information (PII) or protected health information (PHI). The tags must not contain any confidential information, as tags can be viewed without having the actual read permission to an object.

Additional information about object tagging

This section provides information about object tagging in IAM, object tagging with bucket policies, handling object tagging during TSO/PSO, and object tagging during object lifecycle management. Here are additional considerations:

- Object tagging in IAM

  The key function of object tagging as categorization system comes when it is integrated with IAM polices. This allows admin to configure specific user permissions. For example, admin can add a policy that allows everyone to access objects with a specified tag or you can configure and grant permissions to users, who can manage the tags on specific objects. The other key aspect with object tagging is how and where the tags are persisted. This is important because it has a direct impact on various aspects of the system.

- Object tagging with bucket policies

  Object tagging allows you to categorize the objects, additionally tagging gets integrated with various policies. Lifecycle management policy allows you to configure at a bucket level. Earlier versions of ECS supports Expiration, Abort Incomplete Uploads, and Deletion of Expired Object Tagging Delete Marker. The filter could include multiple conditions including a tag-based condition. Each tag in the filter condition must match the key and the value.

- Object tagging during TSO/PSO

  Object tagging is another entry set in system metadata; no special handling is required during TSO/PSO. There is a set limit on the number of tags that are allowed to be associated with each object, size of system metadata along with object tagging is well within the memory limits.

- Object tagging during object lifecycle management

  Object tagging is part of system metadata and handled simultaneously with system metadata handling, during lifecycle management. The Expiration Logic and Lifecycle Delete Scanner requires to understand tag-based policies. Object tags enable fine-grained object lifecycle management in which you can specify a tag-based filter, in addition to a key name prefix, in a lifecycle rule.

See the latest *ECS Security Configuration Guide* for further information on ECS object tagging.

Object lock

Dell ECS object lock protects object versions from accidental or malicious deletion such as a ransomware attack. It does this by allowing object versions to enter a Write Once Read Many (WORM) state, where access is restricted based on attributes set on the object version.

Object lock is designed to meet compliance requirements such as SEC 17a4(f), FINRA Rule 4511(c), and CFTC Rule 17.

For the best application compatibility, object lock is compatible with the capabilities of Amazon S3 object lock.

Object lock overview

Object lock prevents object version deletion during a user-defined retention period. Immutable S3 objects are protected using object- or bucket-level configuration of WORM and retention attributes. The retention policy is defined using the S3 API or bucket-level defaults (also set using the S3 API). Objects are locked for the duration of the retention period, and legal hold scenarios are also supported.

There are two lock types for object lock:

- Retention period -- Specifies a fixed period of time during which an object version remains locked. During this period, your object version 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. Legal holds are independent from retention periods.

There are two modes for the retention period:

- Governance mode -- users can't overwrite or delete an object version or alter its lock settings unless they have special permissions. With governance mode, you protect objects against being deleted by most users, but you can still grant some users permission to alter the retention settings or delete the object if necessary. You can also use governance mode to test retention-period settings before creating a compliance-mode retention period.
- Compliance mode -- a protected object version can't be overwritten or deleted by any user, including the root user in your account. When an object is locked in compliance mode, its retention mode can't be changed, and its retention period can't be shortened. Compliance mode helps ensure that an object version can't be overwritten or deleted for the duration of the retention period.

Object lock requires the use of versioned buckets, enabling object lock on a bucket automatically enables versioning. Once object lock is enabled, it is not possible to disable it or suspend versioning for the bucket. Object locks apply to individual object versions only, different versions of a single object can have different retention modes and periods.

An object can still be deleted, but the version still exists and is locked. Retention period can be placed on an object explicitly, or implicitly through a bucket default setting. Placing a default retention setting on a bucket doesn't place any retention settings on objects that already exist in the bucket. Changing a bucket's default retention period doesn't change the existing retention period for any objects in that bucket. In compliance mode, locks can't be removed, decreased, or downgraded to governance mode. In governance mode, lock can be removed, bypassed, and elevated to compliance mode with the appropriate assigned privilege.

Object lock and lifecycle

Objects under lock are protected from lifecycle deletions.

Lifecycle logic is made difficult due to variety of behavior of different locks. From lifecycle point of view there are locks without a date, locks with date that can be extended, and locks with date that can be decreased.

- For compliance mode, the retain until date can't be decreased, but can be increased:
- For governance mode, the lock date can increase, decrease, or get removed.
- For legal hold, the lock is indefinite.

Object lock condition keys

Access control using IAM policies is an important part of the object lock functionality. The s3:BypassGovernanceRetention permission is important since it is required to delete a WORM-protected object in governance mode (it is not effective for compliance mode). IAM policy conditions have been defined in the following table to support object lock:

Table 8. Object lock condition keys

Condition key	Description
s3:object-lock-legal-hold	Enables enforcement of the specified object legal hold status
s3:object-lock-mode	Enables enforcement of the specified object retention mode
s3:object-lock-retain-until-date	Enables enforcement of a specific retain-until-date
s3:object-lock-remaining-retention-days	Enables enforcement of an object relative to the remaining retention days

Note: Object lock requires ECS ADO and FS (File System) disabled on bucket in ECS 3.6.2 version； Object lock is only supported by S3 API, no UI workflows in ECS 3.6.2 version; It only works with IAM, not legacy accounts.

See the latest *ECS Data Access Guide* for further information on ECS object lock.

Data integrity and protection

Overview

For data integrity, ECS utilizes checksums. Checksums are created during write operations and are stored with the data. On reads checksums are calculated and compared with the stored version. A background task scans verifies checksum information proactively.

For data protection, ECS utilizes triple mirroring for journal chunks and separate EC schemes for *repo* (user repository data) and *btree* (B+ tree) chunks.

Erasure coding provides enhanced data protection from a disk, node and rack failure in a storage efficient fashion compared with conventional protection schemes. The ECS storage engine implements the Reed Solomon error correction using two schemes:

- 12+4 (Default) - Chunk is broken into 12 data segments. 4 coding (parity) segments are created.
- 10+2 (Cold archive) - Chunk is broken in to 10 data segments. 2 coding segments are created.

Using the default of 12+4, the resulting 16 segments are dispersed across nodes at the local site. The data and coding segments of each chunk are equally distributed across nodes in the cluster. For example, with 8 nodes, each node has 2 segments (out of 16 total). The storage engine can reconstruct a chunk from any 12 of the 16 segments.

ECS requires a minimum of six nodes for the cold archive option, in which a 10+2 scheme is used instead of 12+4. EC stops when the number of nodes goes below the minimum required for the EC scheme.

When a chunk is full or after a set period, it’s sealed, parity is calculated, and the coding segments are written to disks across the fault domain. Chunk data remains as a single copy that consists of 16 segments (12 data, 4 code) dispersed throughout the cluster. ECS only uses the code segments for chunk reconstruction when a failure occurs.

When the underlying infrastructure of a VDC changes at the node or rack level, the Fabric layers detects the change and triggers a rebalance scanner as a background task. The scanner calculates the best layout for EC segments across fault domains for each chunk using the new topology. If the new layout provides better protection than the existing layout, ECS re-distributes EC segments in a background task. This task has minimal impact on system performance; however, there will be an increase in inter-node traffic during rebalancing. Balancing of the logical table partitions onto the new nodes also occurs and newly created journal and B+ tree chunks are evenly allocated on old and new nodes going forward. Redistribution enhances local protection by leveraging all of the resources within the infrastructure.

Note: It is recommended not to wait until the storage platform is completely full before adding drives or nodes. A reasonable storage utilization threshold is 70% taking consideration daily ingest rate and expected order, delivery and integration time of added drives/nodes.

Compliance

To meet corporate and industry compliance requirements (SEC Rule 17a-4(f)) for storage of data, ECS implemented the following:

- **Platform hardening** - Hardening addresses security vulnerabilities in ECS such as platform lockdown to disable access to nodes or cluster, all non-essential ports (for example, *ftpd, sshd*) are closed, full audit logging for sudo commands and support for SRS (Dell Secure Remote services) to shut down remote access to nodes.
- **Compliance reporting** - A system agent reports system’s compliance status such as *Good* indicating compliance or *Bad* indicating non-compliance.
- **Policy-based record retention and rules** - Ability to limit changes to records or data under retention using policies, time-period and rules.

- **Advanced Retention Management (ARM)** - To meet Centera compliance requirements a set of retention rules were defined for CAS only.
    - **Event based retention** - Enables retention periods that start when specified event occurs.
    - **Litigation hold** - Enables temporary deletion prevention of data subject to legal action.
    - **Min/max governor** - Per bucket setting for minimum and maximum default retention period.

Compliance is enabled at the namespace level. Retention periods are configured at the bucket level. Compliance requirements certify the platform, and it is because of this that; the compliance feature is only available for ECS running on appliance hardware. For information on enabling and configuring compliance in ECS please refer to the current *ECS Data Access Guide* and the most recent *ECS Administrator’s Guide*.

Deployment

ECS can be deployed as a single or multiple site instance. The building blocks of an ECS deployment include:

- **Virtual Data Center (VDC)** - A cluster, also generally referred to as a site or geographically distinct region, made up of a set of ECS infrastructure managed by a single fabric instance.
- **Storage Pool (SP)** - SPs can be thought of as a subset of nodes and their associated storage belonging to a VDC. A node can belong to only one SP. EC is set at the SP level with either a 12+4 or 10+2 scheme. A SP can be used as a tool for physically separating data between clients or groups of clients accessing storage on ECS.
- **Replication Group (RG)** - RGs define where SP content is protected and locations from which data can accessed. An RG with a single member site is sometimes called a local RG. Data is always protected locally where it is written against disk, node and rack failures. RGs with two or more sites are often called global RGs. Global RGs span up to 8 VDCs and protect against disk, node, rack and site failures. A VDC can belong to multiple RGs.
- **Namespace** - A namespace is conceptually the same as a tenant in ECS. A key characteristic of a namespace is that users from one namespace cannot access objects in another namespace.
- **Buckets** - Buckets are containers for objects created in a namespace and sometimes considered a logical container for sub-tenants. In S3, containers are called buckets and this term has been adopted by ECS. In Atmos, the equivalent of a bucket is a subtenant; in Swift, the equivalent of a bucket is a container, and for CAS, a bucket is a CAS pool. Buckets are global resources in ECS. Each bucket is created in a namespace and each namespace is created in an RG.

ECS leverages the following infrastructure systems:

- **DNS** - (required) Forward and reverse lookups required for each ECS node.
- **NTP** - (required) Network Time Protocol server.

- **SMTP** - (optional) Simple Mail Transfer Protocol Server for sending alerts and reporting.
- **DHCP** - (optional) Required if assigning IP addresses via DHCP.
- **Authentication Providers** - (optional) ECS administrators can be authenticated using Active Directory and LDAP groups. Object users can be authenticated using Keystone. Authentication providers are not required for ECS. ECS has local user management functionality built-in however do note that users created locally are not replicated between VDCs.
- **Load Balancer** - (required if workflow dictates, otherwise optional) Client load should be distributed across nodes to effectively utilize all resources available in the system. If a dedicated load balancer appliance or service is needed to manage the load across ECS nodes, it should be considered a requirement. Developers writing applications using the ECS S3 SDK can take advantage of its built-in load balancer functionality. Sophisticated load balancers may take additional factors into account, such as a server's reported load, response times, up/down status, number of active connections and geographic location. The customer is responsible for managing client traffic and determining access requirements. Regardless of method there are a few basic options that are generally considered including manual IP allocation, DNS Round Robin, Client-Side Load Balancing, Load Balancer Appliances and Geographic Load Balancers. The following are brief descriptions of each of those methods:
    - **Manual IP allocation** - IP addresses are manually distributed to applications. This is generally not recommended as it may not distribute load or provide fault-tolerance.
    - **DNS round-robin** - A DNS entry is created that includes all node IP addresses. Clients query DNS to resolve fully-qualified domain names for ECS services and are answered with the IP addresses of a random node. This may provide some pseudo-load balancing. This method may not provide fault-tolerance because often manual intervention is used to remove IP addresses of failed nodes from DNS. Time to live (TTL) issues may be encountered with this method. Some DNS server implementations may cache DNS lookups for a period such that clients connecting in a close timeframe may bind to the same IP address, reducing the amount of load distribution to the data nodes. Using DNS for distributing traffic in a round-robin fashion is not recommended.
    - **Load balancing** - Load balancers are the most common approach to distributing client load. Clients can send traffic to a load balancer which receives and forwards it on to a healthy ECS node. Proactive health checks or connection state are used to verify each node’s availability to service requests. Unavailable nodes are removed from use until they pass a health check. Offloading CPU-intensive SSL processing can be used to free up those resource on ECS.
    - **Geographic load balancing** - Geographic load balancing leverages DNS to route lookups to an appliance like the Riverbed SteelApp, for example, which use Geo-IP or another mechanism to determine the best site to route the client to.

Single site deployment

During a single site, or single cluster initial deployment, nodes are first added into a SP. SPs are logical containers of physical nodes. SP configuration involves selecting the required minimum number of available nodes and choosing the default 12+4 or cold archive 10+2 EC scheme. Critical alert levels may be set during SP configuration initially and in the future; however, EC schema cannot be changed after SP initialization. The first SP created is designated as the system SP and is used to store system metadata. The system SP cannot be deleted.

Clusters generally contain one or two SPs, as shown in the following figure—one for each EC schema; however, if an organization requires physical separation of data, additional SPs are used to implement boundaries.

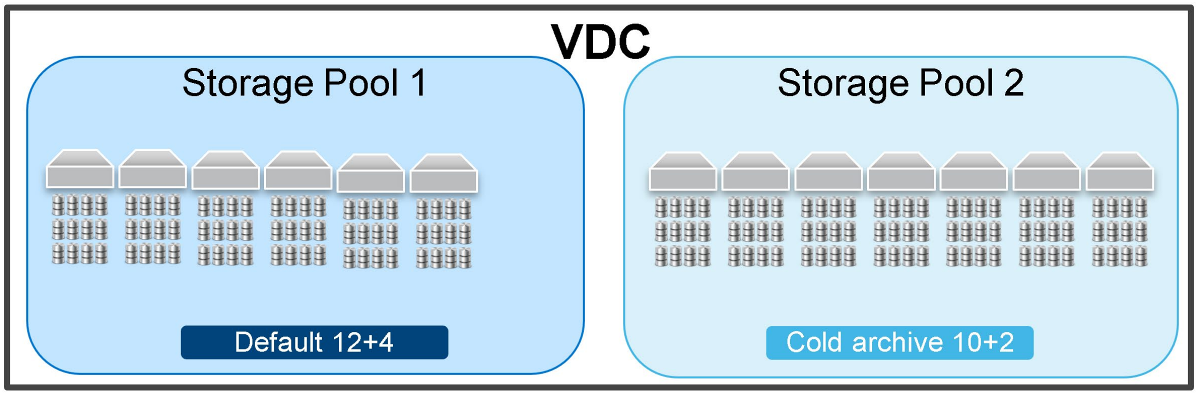


Figure 22. VDC with two storage pools, each configured with different EC scheme

After the initialization of the first SP, a VDC can be created. VDC configuration involves designating replication and management endpoints. Note that although system SP initialization is required prior to VDC creation, VDC configuration does not assign SPs but rather the IP addresses of nodes.

After a VDC is created, RGs are configured. RGs are global resources with configuration involving designating at least one VDC, itself, in the single- or initial-site setup, along with one of the VDC's SPs. An RG with a single VDC member protects data locally at the disk, node and rack level. The next section expands on RGs to include multisite deployments.

Namespaces are global resources created and assigned to an RG. At the namespace level retention policies, quotas, compliance and namespace administrators are defined. Access During Outage (ADO) can be configured at the namespace level which is covered in the next section. Generally, it is at the namespace level where tenants are organized. Tenants may be an application instance or team, user, business group or any other grouping that makes sense to the organization.

Buckets are global resources that can span multiple sites. Bucket creation involves assigning it to a namespace and an RG. The bucket level is where ownership and file or CAS access is enabled. The following figure shows one SP in a VDC with a namespace containing two buckets:

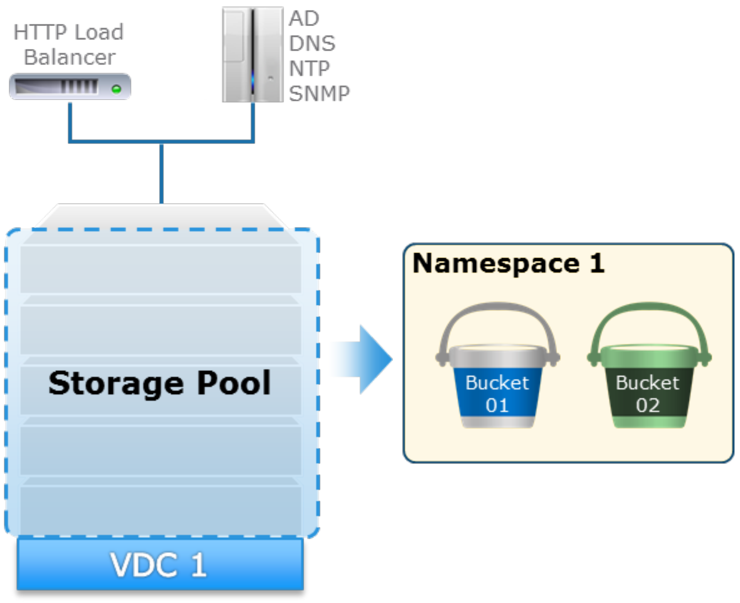


Figure 23. Single site deployment example

Multisite deployment

A multisite deployment, also referred to as a federated environment or federated ECS, may span across up to eight VDCs. Data is replicated in ECS at the chunk level. Nodes participating in an RG send their local data asynchronously to one or all other sites. Data is encrypted using AES256 before it is sent across the WAN via HTTP. Key benefits recognized when federating multiple VDCs are:

- Consolidation of multi-VDC management efforts into a single logical resource
- Site-level protection in addition to locally to the node-, disk- and rack-level
- Geographically distributed access to storage in an everywhere-active strongly-consistent manner

This section on multisite deployment describes features specific to federated ECS such as:

- **Data consistency** - By default ECS provides a strongly-consistent storage service.
- **Replication Groups** - Global containers used to designate protection and access boundaries.
- **Geo-caching** - Optimization for remote-site access workflows in multisite deployments.
- **ADO** - Client access behavior during temporary site outage (TSO).

Data consistency

ECS is a strongly consistent system that uses ownership to maintain an authoritative version of each namespace, bucket and object. Ownership is assigned to the VDC where the namespace, bucket or object is created. For example, if a namespace, NS1, is created at VDC1, VDC1 owns NS1 and responsible for maintaining the authoritative version of buckets inside NS1. If a bucket, B1, is created at VDC2 inside NS1, VDC2 owns B1 and responsible for maintaining the authoritative version of the bucket contents as well as each object's owner VDC. Similarly, if an object, O1, is created inside B1 at

VDC3, VDC3 owns O1 and is responsible for maintaining the authoritative version of O1 and associated metadata.

The resiliency of multisite data protection comes at the expense of increased storage protection overhead and WAN bandwidth consumption. Index queries are required when an object is accessed or updated from a site that does not own the object. Similarly index lookups across the WAN are also required to retrieve information such as an authoritative list of buckets in a namespace or objects in a bucket, owned by a remote site.

Understanding how ECS uses ownership to authoritatively track data at the namespace, bucket and object level helps administrators and application owners make decisions in configuring their environment for access.

Active replication group

During RG creation a *Replicate to All Sites* setting is available which is either left off, by default, or can be toggled on which enables this feature. Replicating data to all sites means that data written individually to each VDC is replicated to all other RG member VDCs. For example, a federated X-number-of-sites ECS instance with an active RG configured to replicate data to all sites will result in X times protection overhead, or X * 1.33 (or 1.2 in cold archive EC) total data protection overhead. Replicating to all sites may make sense especially for smaller data sets where local access is important. Leaving this setting off means that all data written to each VDC will be replicated to one other VDC. The primary site, where on object is created, and the site storing the replicate copy, each protect the data locally using the EC schema assigned to the local SP. That is, that only the original data is replicated across the WAN and not any associated EC coding segments.

Data stored in an active RG is accessible to clients via any available RG member VDC. Figure 23 shows an example of a federated ECS built using VDC1, VDC2 and VDC3. Two RGs are shown, RG1 has a single member, VDC1, and RG2 has all three VDCs as members. Three buckets are shown, B1, B2 and B3.

In this example:

- Clients accessing VDC1 have access to all buckets
- Clients accessing VDC2 and VDC3 have access only to buckets B2 and B3.

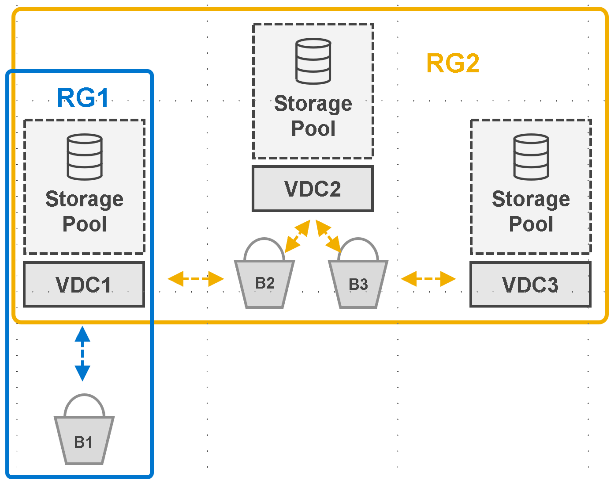


Figure 24. Bucket-level access by site with single site and multisite replication groups

Passive replication group

A passive RG has three member VDCs. Two of the VDCs are designated as active and are accessible to clients. The third VDC is designated passive and used as a replication target only. The passive site is used for recovery purposes only and does not allow for direct client access. Benefits of geo-passive replication are:

- Decrease in storage protection overhead by increasing the potential for XOR operations
- Administrator-level control of the location used for replicate-only storage

The following figure shows an example of a geo-passive configuration whereby VDC 1 and VDC 2 are primary (source) sites that both replicate their data (chunks) to the replication target, VDC 3:

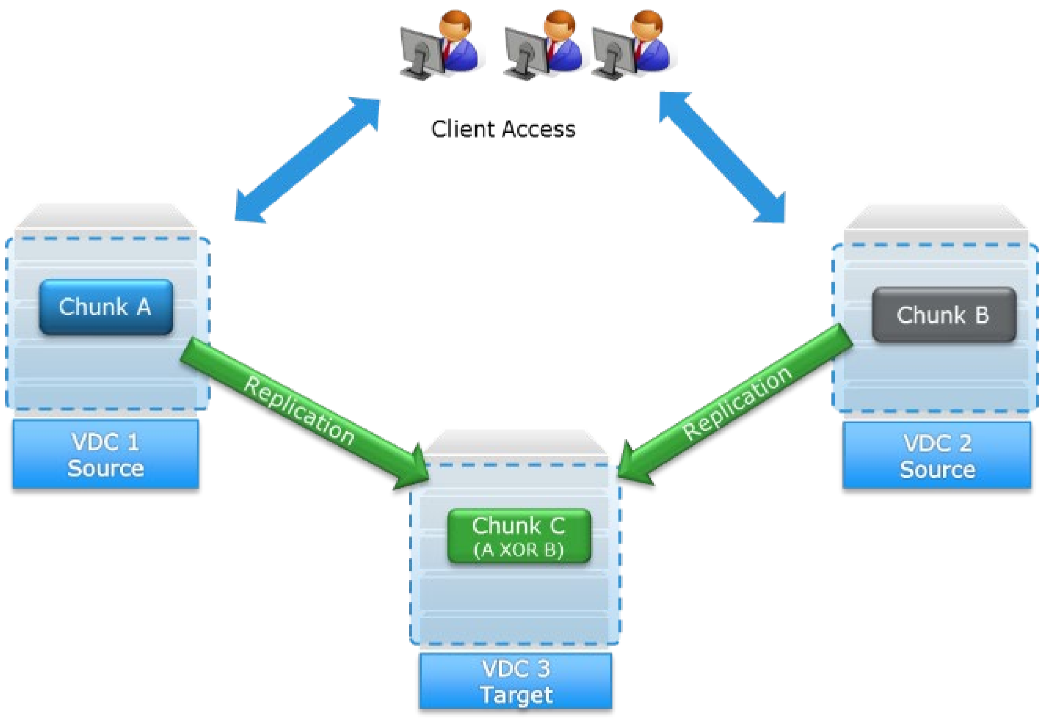


Figure 25. Client access and replication paths for geo-passive replication group

Multisite access to strongly-consistent data is accomplished using namespace, bucket and object ownership across RG member sites. Inter-site across-the-WAN index queries are required when API access originates from a VDC that doesn’t own the required logical construct(s). WAN lookups are used to determine the authoritative version of data. Thus, if an object created in Site 1 is read from Site 2, a WAN lookup is required to query the object’s owner VDC, Site 1, to verify if the object’s data that has been replicated to Site 2 is the latest version of the data. If Site 2 doesn’t have the latest version, it fetches the necessary data from Site1; otherwise, it uses the data previously replicated to it. This is illustrated in the following figure:

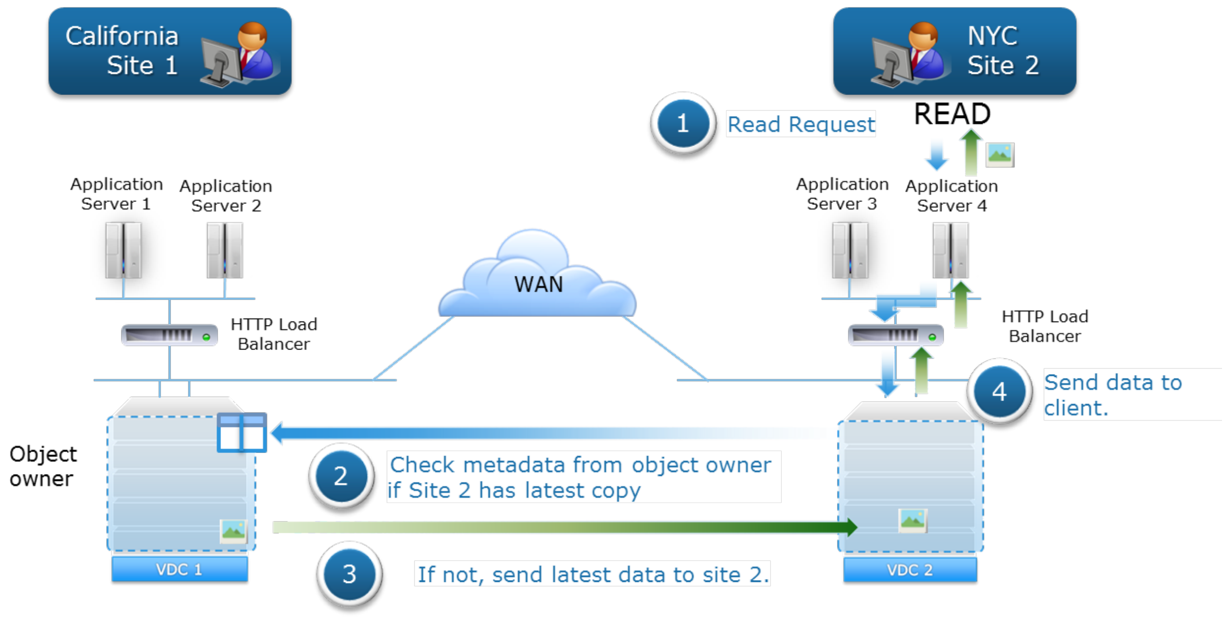


Figure 26. Read request to non-owner VDC triggers WAN lookup to object-owner VDC

Figure 26 shows the data flow of writes in a geo-replicated environment in which two sites are updating the same object. In this example, Site 1 initially created and owns the object. The object has been erasure-coded and the related journal transactions written to disk at Site 1. The data flow for an update to the object received at Site 2 is as follows:

1. Site 2 first writes the data locally.
2. Site 2 synchronously updates the metadata (journal write) with the object owner, Site 1, and waits for acknowledgement of metadata update from Site 1.
3. Site 1 acknowledges the metadata write to Site 2.
4. Site 2 acknowledges the write to the client.

Note: Site 2 asynchronously replicates the data Site 1, the object owner site, as usual. If the data must be served from Site 1 before it is replicated to it from Site 2, Site 1 will retrieve the data directly from Site 2.

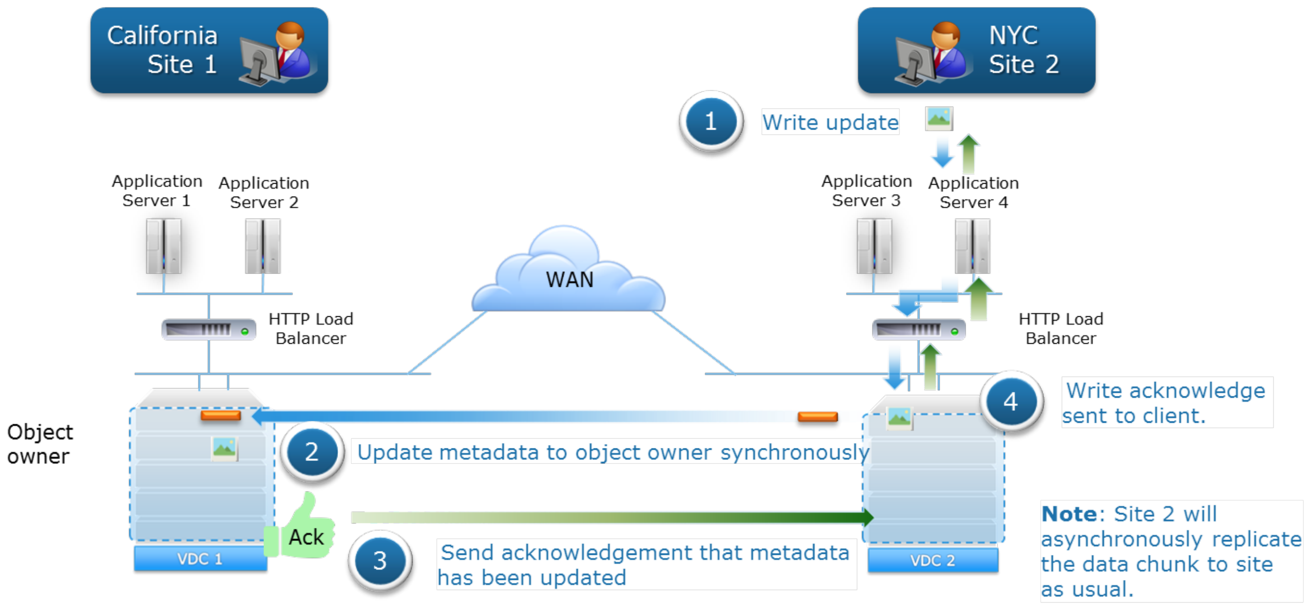

California Site 1
NYC Site 2
1 Write update
Application Server 1
Application Server 2
Application Server 3
Application Server 4
WAN
HTTP Load Balancer
HTTP Load Balancer
Object owner
2 Update metadata to object owner synchronously
4 Write acknowledge sent to client.
Ack
3 Send acknowledgement that metadata has been updated
VDC 1
VDC 2
Note: Site 2 will asynchronously replicate the data chunk to site as usual.

Figure 27. Update of same object data flow in geo-replicated environment

In both read and write scenarios in a geo-replicated environment, there is latency involved in reading and updating the metadata and retrieving data from the object-owner site.

Note: Starting from ECS 3.4, you can remove a VDC from a replication group (RG) in a multi VDC federation without affecting the VDC or other RGs associated with the VDC. Removing VDC from RG no longer initiates PSO (permanent site outage). Removing a VDC from RG initiates recovery.

See the latest *ECS Administrator Guide* for further information on Replication Group.

Geo-caching remote data

ECS optimizes response times for accessing data stored on remote sites by locally caching objects read across the WAN. This is can be useful for multi-site access patterns where data is often fetched from a remote, or non-owner site. Consider a geo-replicated environment with three sites, VDC1, VDC2 and VDC3, where an object is written to VDC1 and the replicate copy of the object is stored at VDC2. In this scenario, to service a read request received at VDC3, for the object created at VDC1 and replicated to VDC2, the object data must be sent to VDC3 from either VDC1 or VDC2. Geo-caching frequently accessed remote data helps reduce response times. A Least Recently Used algorithm is used for caching. Geo-cache size is adjusted when hardware infrastructure such as disks, nodes and racks are added to a geo-replicated SP.

Behavior during site outage

Temporary site outage (TSO) generally refers to either a failure of WAN connectivity or of an entire site such as during a natural disaster. ECS uses heartbeat mechanisms to detect and handle temporary site failures. Client access and API-operation availability at the namespace, bucket and object levels during a TSO is governed the following ADO options set at the namespace and bucket level:

- **Off (default)** - Strong consistency is maintained during a temporary outage.
- **On** - Eventually consistent access is allowed during a temporary site outage.

Data consistency during a TSO is implemented at the bucket level. Configuration is set at the namespace level, which sets the default ADO setting in place for ADO during new bucket creation. and can be overridden at new bucket creation; meaning TSO can be configured for some buckets and not for others.

Access during outage (ADO) not enabled

By default, ADO is not enabled, and strong consistency is maintained. All client API requests where authoritative namespace, bucket or object data is required but temporarily unavailable will fail. Object operations of read, create, update and delete as well as list buckets not owned by an online site, will fail. Also, operations of create and edit of bucket, user and namespace will also fail.

As previously mentioned, the initial site owner of bucket, namespace and an object, is the site where the resource was first created. During a TSO, certain operations may fail if the site owner of resource is not accessible. Highlights of operations permitted or not permitted during a temporary site outage include:

- Creation, deletion, and update of buckets, namespaces, object users, authentication providers, RGs and NFS user and group mappings are not allowed from any site.
- Listing buckets within a namespace is allowed if the namespace owner site is available.

HDFS/NFS enables buckets that are owned by the inaccessible site are read-only.

ADO enabled

In an ADO-enabled bucket, during a TSO, the storage service provides eventually consistent responses. In this scenario reads and optionally writes from a secondary (non-owner) site are accepted and honored. Further, a write to a secondary site during a TSO causes the secondary site to take ownership of the object. This allows each VDC to continue to read and write objects from buckets in a shared namespace. Finally, the new version of the object becomes the authoritative version of the object during post-TSO reconciliation even if another application updates the object on the owner VDC.

Although many object operations continue during a network outage, certain operations are not be permitted, such as creating new buckets, namespaces, or users. When network connectivity between two VDCs is restored, the heartbeat mechanism automatically detects connectivity, restores service and reconciles objects from the two VDCs. If the same object is updated on both VDC A and VDC B, the copy on the non-owner VDC is the authoritative copy. So, if an object that is owned by VDC B is updated on both VDC A and VDC B during synchronization, the copy on VDC A will be the authoritative copy that is kept, and the other copy will be un-referenced and available for space reclamation.

When more than two VDCs are part of an RG, and if network connectivity is interrupted between one VDC and the other two, then write/update/ownership operations continue just as they would with two VDCS; however, the process for responding to read requests is more complex, as described below.

If an application requests an object that is owned by a VDC that is not reachable, ECS sends the request to the VDC with the secondary copy of the object. However, the secondary site copy might have been subject to a data contraction operation, which is an XOR between two different data sets that produces a new data set. Therefore, the secondary site VDC must first retrieve the chunks of the object included in the original XOR operation and it must XOR those chunks with the recovery copy. This operation will return the contents of the chunk originally stored on the failed VDC. The chunks from the recovered object can then be reassembled and returned. When the chunks are reconstructed, they are also cached so that the VDC can respond more quickly to subsequent requests. Note reconstruction is time consuming. More VDCs in an RG imply more chunks that must be retrieved from other VDC's, and hence reconstructing the object takes longer.

If a disaster occurs, an entire VDC can become unrecoverable. ECS treats the unrecoverable VDC as a temporary site failure. If the failure is permanent, the system administrator must permanently failover the VDC from the federation to initiate fail over processing, which initiates resynchronization and re-protection of the objects stored on the failed VDC. The recovery tasks run as a background process. You can review the recovery progress in the ECS Portal.

An additional bucket option is available for *read-only (RO)* ADO which ensures object ownership never changes and removes the chance of conflicts otherwise caused by object updates on both the failed and online sites during a temporary site outage. The disadvantage of RO ADO is that during a temporary site outage no new objects can be created and no existing objects in the bucket can be updated until after all sites are back online. The RO ADO option is available during bucket creation only, it can't be modified afterwards. By default, this option is disabled.

Failure tolerance

ECS is designed to tolerate a range of equipment failure situations using a number of fault domains. The range of failure conditions spans a varying scope including:

- Single hard drive failure in a single node
- Multiple hard drive failure in a single node
- Multiple nodes with single hard drive failure
- Multiple nodes with multiple hard drive failures
- Single node failure
- Multiple node failure
- Loss of communication to one replicated VDC
- Loss of one entire replicated VDC

In either a single site, dual-site, or geo-replicated configuration, the impact of the failure depends on the quantity and type of components affected. However, at each level, ECS provides mechanisms to defend against the impact of component failures. Many of these mechanisms have already been discussed in this paper but are reviewed here and in Figure 27 to show how they are applied to the solution. These include:

- Disk failure
    - EC segments or replica copies from the same chunk are not stored on the same disk
    - Checksum calculation on write and read operations
    - Background consistency checker re-verifying checksums

- Node failure
  - Distribute segments or replica copies of a chunk equally across nodes in a VDC
  - ECS Fabric keeps services running and manages resources such as disks and network.
  - Partition records and tables protected by partition ownership failover from node to node.
- Rack failure within VDC
  - Distribute segments of replica copies of a chunk equally across racks in a VDC.
  - One fabric registry instance runs in each rack and can be restarted on any other node in the same rack should the node fail.

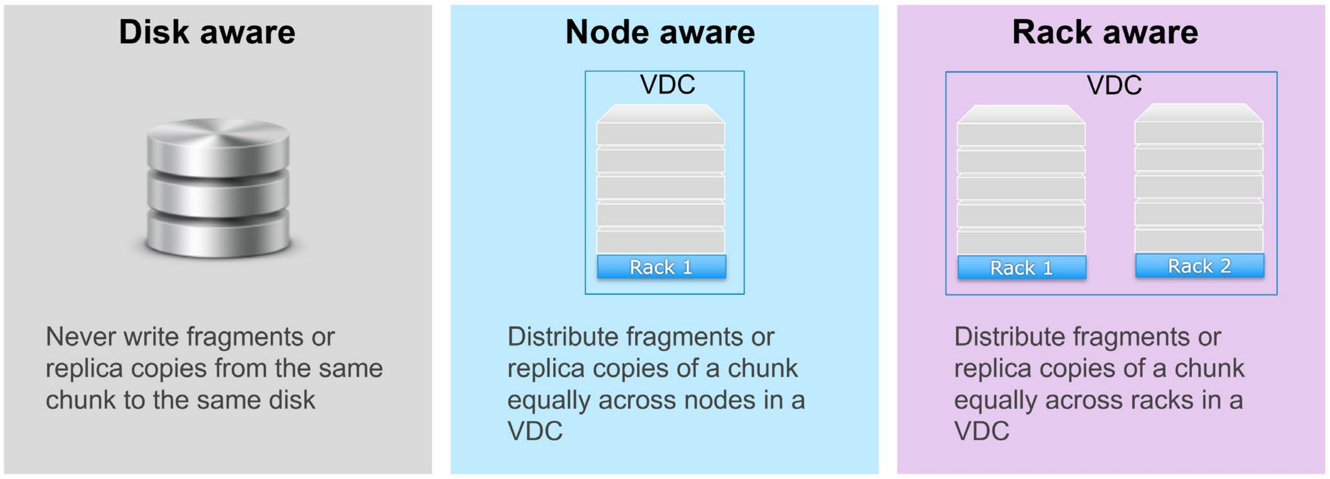

Disk aware
Never write fragments or replica copies from the same chunk to the same disk
Node aware
VDC
Rack 1
Distribute fragments or replica copies of a chunk equally across nodes in a VDC
Rack aware
VDC
Rack 1
Rack 2
Distribute fragments or replica copies of a chunk equally across racks in a VDC

Figure 28. Protection mechanisms at the disk, node, and rack levels

The following table defines the type and number of component failures that each EC scheme protects against per basic rack configuration. The table highlights the importance of considering the impact of protective failure domains on overall data and service availability in terms of number of nodes required at each EC scheme.

Table 9. Erasure code protection across failure domains

EC scheme	# nodes in VDC	# chunk fragments per node	EC data protected against…
12+4 Default	5 or less	4	• Loss of up to four disks or • Loss of one node
	6 or 7	3	• Loss of up to four disks or • Loss of one node and one disk from a second node
	8 or more	2	• Loss of up to four disks or • Loss of two nodes or • Loss of one node and two disks

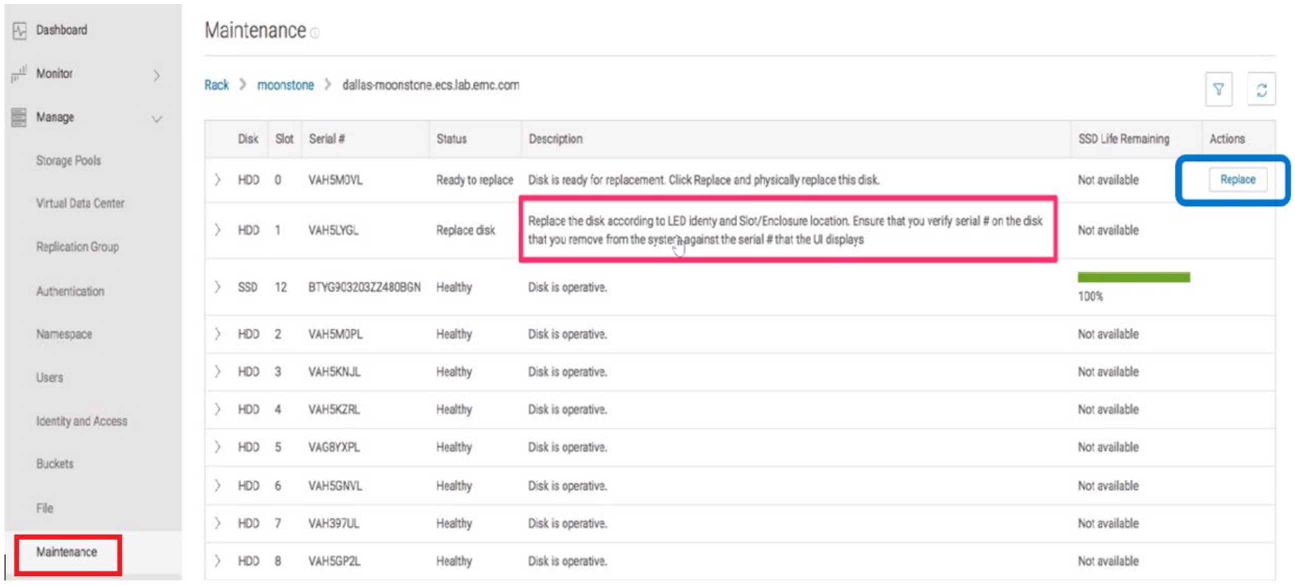
EC scheme	# nodes in VDC	# chunk fragments per node	EC data protected against…
	16 or more	1	• Loss of four nodes or • Loss of three nodes and disks from one additional node or • Loss of two nodes and disks from up to two different nodes or • Loss of one node and disks from up to three different nodes or • Loss of four disks from four different nodes
10+2 Cold Storage	11 or less	2	• Loss of up to two disks or • Loss of one node
	12 or more	1	• Loss of any number of disks from two different nodes or • Loss of two nodes

Disk replacement automation

Beginning in ECS 3.5 customers can replace failed disks with Dell services using an intuitive ECS portal (Web UI) workflow. The feature provides:

- Do-it-yourself resolution to drive failures
- Accelerated time to break fix
- Operational flexibility and TCO savings

The maintenance page in the ECS portal provides admin visibility to all disks in each node. When a drive fails the system automatically initiates recovery. All types of resources on drive are recovered and when the drive is ready to be removed from the node, ECS portal will display the replace button, as shown in the following figure:



Maintenance

Rack > moonstone > dallas-moonstone.ecs.lab.emc.com

Disk	Slot	Serial #	Status	Description	SSD Life Remaining	Actions
HDD	0	VAH5M0VL	Ready to replace	Disk is ready for replacement. Click Replace and physically replace this disk.	Not available	Replace
HDD	1	VAH5LYGL	Replace disk	Replace the disk according to LED identy and Slot/Enclosure location. Ensure that you verify serial # on the disk that you remove from the system against the serial # that the UI displays	Not available	
SSD	12	BTYG903203ZZ480BGN	Healthy	Disk is operative.	100%	
HDD	2	VAH5M0PL	Healthy	Disk is operative.	Not available	
HDD	3	VAH5KNJL	Healthy	Disk is operative.	Not available	
HDD	4	VAH5KZRL	Healthy	Disk is operative.	Not available	
HDD	5	VAG8YXPL	Healthy	Disk is operative.	Not available	
HDD	6	VAH5GNVL	Healthy	Disk is operative.	Not available	
HDD	7	VAH397UL	Healthy	Disk is operative.	Not available	
HDD	8	VAH5GP2L	Healthy	Disk is operative.	Not available	

Figure 29. Disk replacement automation

Note: Only one drive can be replaced at a time. This is to avoid replacement of the wrong drive.

Tech refresh

Tech refresh is a Dell Professional Services directed engagement available beginning in ECS 3.5 to non-disruptively remove older hardware nodes from ECS clusters using the embedded software feature. It is an efficient and low resource consuming operation that can be precisely throttled. This feature reduces the overhead previously associated with decommissioning of ECS hardware.

Tech refresh includes three parts:

- **Node extend**: Adding Gen3 nodes to existing cluster
- **Resource migration**: Move all resources from existing nodes to Gen 3 nodes
- **Node evacuation**: Clean up old nodes and remove them from the cluster

Professional Services should be involved during tech refresh maintenance. See the latest *ECS Tech Refresh Guide* for further information on Tech Refresh.

Storage protection overhead

Each VDC member in an RG is responsible for its own EC protection of data at the local level. That is, data is replicated but not any related coding segments. Although EC is more storage efficient than other forms of protection, such as full copy drive mirroring, it does incur an inherent storage cost overhead at the local level. However, when it is required to have secondary copies replicated offsite as well as all sites having access to data when a single site becomes unavailable, the storage costs become more extensive than when using traditional site-to-site data copying protection methods. This is especially true when unique data is distributed across three or more sites.

ECS provides a mechanism in which storage protection overhead efficiency can increase as three or more sites are federated. In a two-VDC replicated environment ECS replicates chunks from the primary, or owner VDC to a remote site to provide high availability and resiliency. There is no way to evade the 100% cost of protection overhead of a full copy of data in a two-site federated ECS deployment.

Now, consider three VDC's in a multi-site environment, VDC1, VDC2 and VDC3, where each VDC has unique data replicated to it from each of the other VDC's. VDC2 and VDC3 may send a copy of their data to VDC1 for protection. VDC1 would therefore have its own original data, plus replicate data from VDC2 and VDC3. This means that VDC1 would be storing 3X the amount of data written at its own site.

In this situation ECS can perform an XOR operation of VDC2 and VDC3 data locally stored at VDC1. This mathematical operation compares equal quantities of unique data, chunks, and renders a result in new chunk which contain enough characteristics of the two original data chunks to make it possible to restore either of the two original sets. So, where previously there were three unique sets of data chunks stored on VDC1, consuming 3X the available capacity, there is now only two - the original local data set, and the XOR reduced protection copies.

In this same scenario, if VDC3 becomes unavailable, ECS can reconstruct VDC3 data chunks by using chunk copies recalled from VDC2 and the (C1 $\oplus$ C2) data from VDC3 stored locally at VDC1. This principle applies to all three sites participating in the RG and is dependent upon each of the three VDC's having unique data sets. The following figure shows an XOR calculation with two sites replicating to a third site.

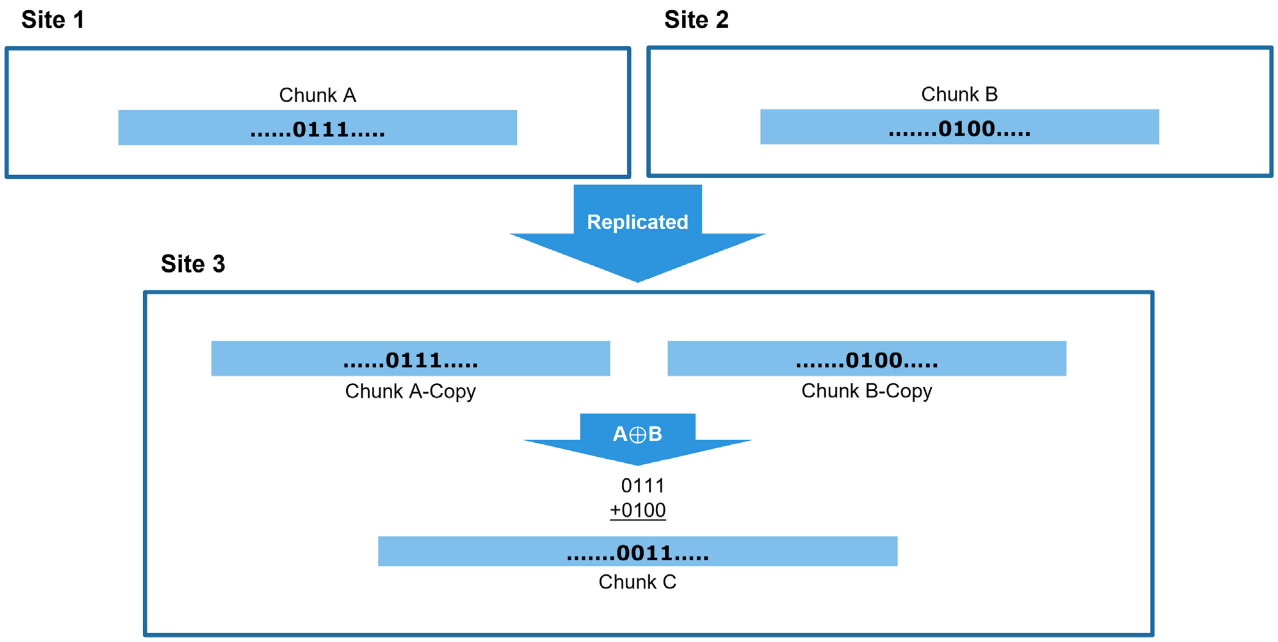


Figure 30. XOR data protection efficiency

If business service level agreements require optimum read access speeds even in the event of a full site failure, then the replicate to all sites setting forces ECS to revert to full copies of replicated data to be stored at all sites. Expectedly, this drives up the storage costs in proportion to the number of VDCs participating in the RG. Therefore a 3-site configuration would revert to 3X storage protection overhead. Replicate to All Sites setting is available during RG creation and cannot be toggled back and forth.

As the number of federated sites increases, the XOR optimization is more efficient in reducing the storage protection overhead due to replication. The following table provides information on the storage protection overhead based on the number of sites for normal EC of 12+4 and cold archive EC of 10+2, illustrating how ECS can become more storage efficient as more sites are linked.

Note: To lower replicated data overhead across three, and up to eight sites, unique data must be written relatively equally at each site. By writing data in equal amounts across sites, each site will have a similar number of replica chunks. Similar numbers of replica chunks at each site lead to similar number of XOR operations that can occur at each site. Maximum multisite storage efficiency is gained by reducing the maximum number of replica chunks stored by using XOR.

Table 10. Storage protection overhead

# sites in RG	12+4 EC	10+2 EC
1	1.33	1.2
2	2.67	2.4
3	2.00	1.8
4	1.77	1.6

# sites in RG	12+4 EC	10+2 EC
5	1.67	1.5
6	1.60	1.44
7	1.55	1.40
8 (Max # sites in RG)	1.52	1.37

Conclusion

Organizations are facing ever-increasing amounts of data and storage costs, particularly in the public cloud space. The ECS scale-out and geo-distributed architecture delivers an on-premises cloud platform that scales to exabytes of data with a *Total Cost of Ownership* that's significantly less than public cloud storage. ECS is a great solution because of its versatility, hyper-scalability, powerful features, and use of commodity hardware.

Appendix: Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical documents and videos provide expertise that helps to ensure customer success on Dell storage platforms.