Technical White Paper

Dell EMC ECS: Data-at-Rest Encryption

Abstract

Dell EMC ECS provides support for Data at Rest Encryption (D@RE) which provides simple, low-touch, server-side encryption. This feature helps enterprises and service providers protect sensitive data on storage media.

August 2021

Revisions

Date	Description
August 2021	Initial release

Acknowledgments

Author: Jarvis Zhu

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

This document may contain certain words that are not consistent with Dell's current language guidelines. Dell plans to update the document over subsequent future releases to revise these words accordingly.

Table of contents

Executive summary

Compliance requirements often mandate the use of encryption to protect data written on disks. Dell EMC ECS supports Data at Rest Encryption (D@RE) that can be enabled at the namespace and bucket levels. To support D@RE, ECS maintains a hierarchy of encryption keys where a parent key in the hierarchy is used to protect a child key. Before ECS version 3.3, these keys were natively managed by ECS across the geo-federated environment. ECS version 3.3 adds support for certain external key-management solutions that are compliant with the Key Management Interoperability Protocol (KMIP).

1 Data-at-rest encryption

Data at Rest Encryption (D@RE) is simple, low-touch, server-side encryption. It helps enterprises and service providers protect sensitive data on storage media. It also encrypts data inline before storing it on ECS disks or drives. This encryption helps prevent users from acquiring sensitive data from discarded or stolen media. It is a required feature in many financial and healthcare use cases that must adhere to regulatory compliance.

D@RE has the following features:

- ECS 3.6.1 supports FIPS 140-2 mode by default only for the DARE module and it is Level 1 compliant using an AES 256-bit encryption algorithm.
- ECS uses RSA BSAFE Crypto-J JSAFE and JCE software module version 6.2.5 for data encryption that is based on the AES256 algorithm.
- Enabled through the ECS Portal or ECS REST Management APIs.
- Can be applied at the namespace and bucket level with transitivity.
- Not all buckets or objects must be encrypted within a specific namespace.
- Supports Amazon S3 ServerSide Encryption (SSE) constructs which enable for object encryption and user-supplied keys.
- Each namespace, bucket, and object have an associated key that is autogenerated at creation.
- Keys are separated between namespaces.
- All user data are encrypted inline before being stored on ECS commodity drives.
- There is no limit on the number of namespaces and buckets that can be encrypted.

The ECS D@RE implementation encrypts all customer data. The data that resides on the storage system includes any user metadata (applicable to S3 and Swift users) associated with objects. System metadata such as timestamps, object location information, access control lists, object, and bucket names are not in the scope of data-at-rest encryption. Names of objects and buckets are excluded from D@RE because their encryption impacts indexing of the data.

ECS software distribution is available in two forms: one with D@RE and one without. EMC recommends that all users have the ECS software with D@RE except where D@RE is not lawful. For customers who should have access to D@RE, the ECS license file includes D@RE. When this license is applied to the appropriate ECS software distribution, the feature is initialized and available.

Note: Some countries (such as China and Russia) do not permit software with strong encryption. For those countries, ECS is packaged without D@RE.

2 Key management

ECS maintains a hierarchy of encryption keys where a parent key in the hierarchy is used to protect a child key, to support D@RE. ECS maintains a set of encryption keys in the system. This includes Data Encryption keys (DEKs) and Key Encryption Keys (KEKs). In order to support industry standard practices, we need to support rotation of keys which is the changing of keys to protect new incoming data. This can be done periodically to limit the amount of data protected by the set of KEKs or in response to a potential leak or compromise. The key rotation is available for both native and external key management.

ECS supports the following two ways for managing keys in the federation:

- Native Key Management
- External Key Management

2.1 Native key management

Native encryption key management uses a hierarchy as shown in Figure 1. Each VDC is associated with a public-private key pair which is generated during initial setup. The pair is used to protect natively generated master keys and the resource table (RT) data encryption key. The virtual master key is a key that is derived using HMAC SHA 256 from master key and RT data encryption key. The virtual master key is used to protect all natively generated rotation keys. The rotation key in turn is used to derive virtual master and bucket keys. Each namespace in ECS has an associated namespace key, which is randomly generated and protected by the virtual master key. This virtual key is never persisted to disks. All buckets in a namespace have an associated randomly generated bucket key which is protected by the corresponding parent namespace key. Data for each object is encrypted using a randomly generated object key, which is then protected using a virtual bucket key. The virtual bucket key is derived from bucket and rotation key using HMAC SHA 256 and is never saved to disks.

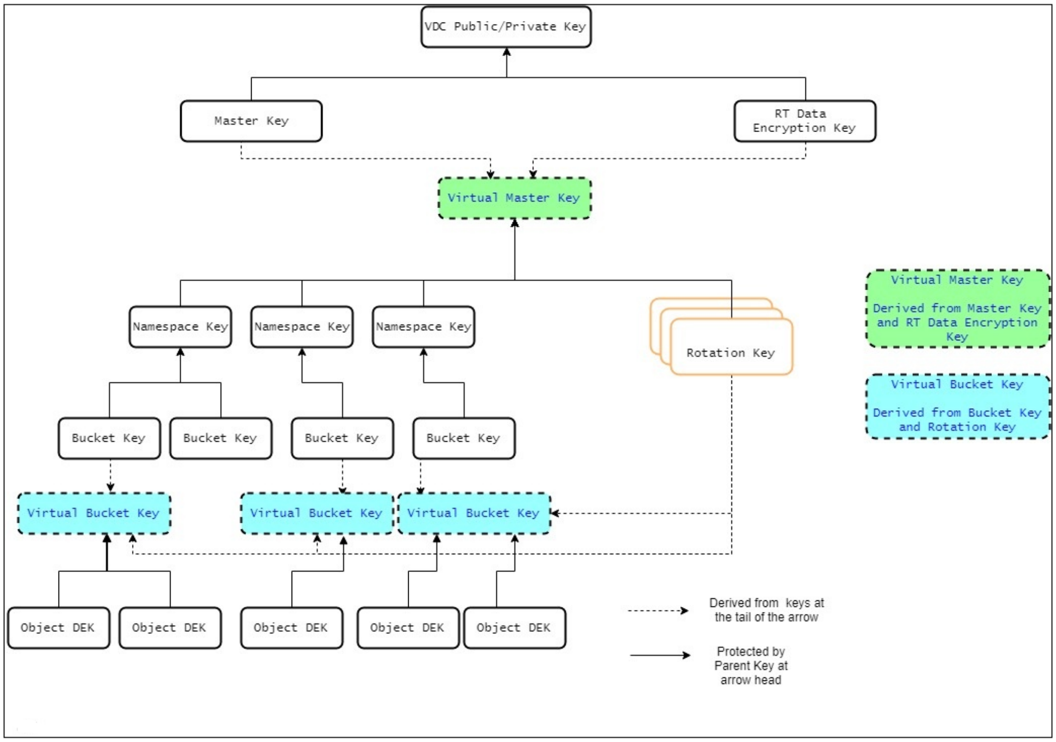


Figure 1 Native key management key hierarchy

Figure 2 shows the location of natively managed keys in ECS systems. The master key and RT data encryption key are wrapped with the respective VDC public key using RSA/ECB/PKCS1padding and stored in the resource table (RT). The resource table is another distributed key-value store implementation that spans across all VDCs in the federation and uses the commodity disks. There is one instance each of wrapped master and RT data encryption keys per VDC in the RT. A VDC attempts to get the master or RT data encryption key, reads the encrypted data corresponding to the VDC, and uses the private key to decrypt and retrieve the keys.

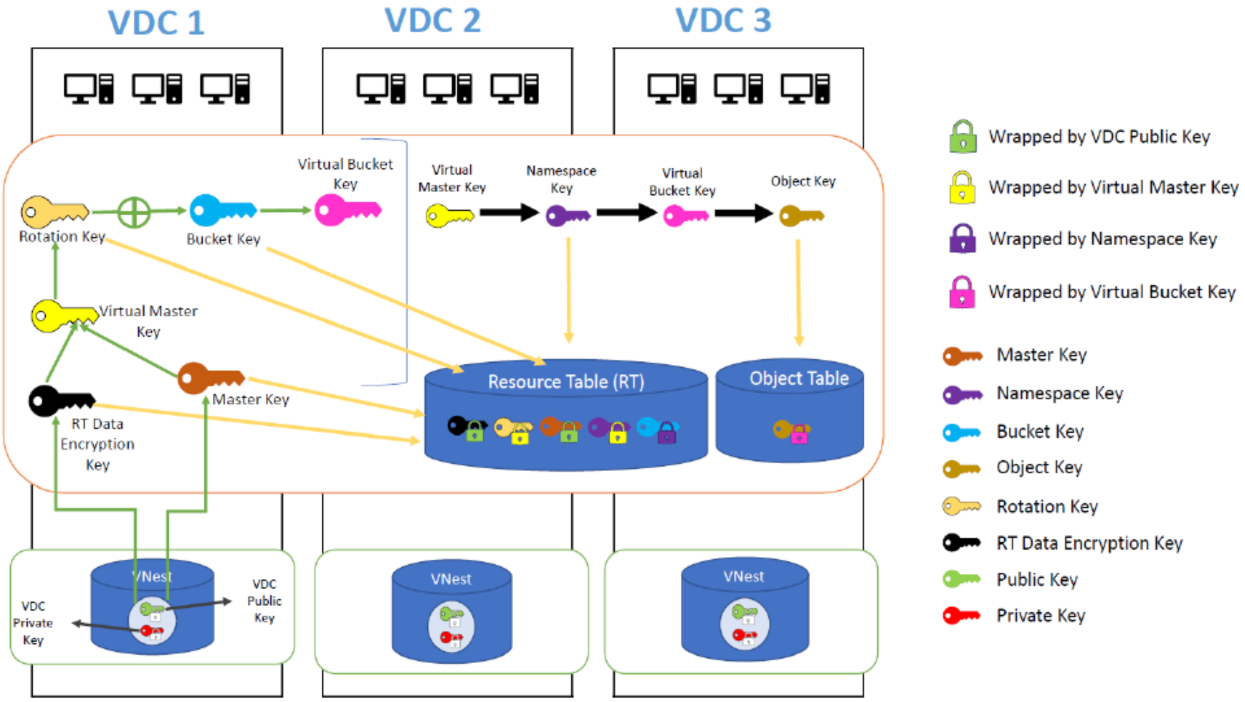


Figure 2 Storage locations of natively managed keys

The rotation key and namespace key are wrapped using AESKeyWrapRFC5649 by virtual master key. The bucket is wrapped using AESKeyWrapRFC5649 by namespace key. These wrapped keys are also stored in the Resource table. Object keys are wrapped by the virtual bucket key using AESKeyWrapRFC5649 and stored in Object table. The object table is another instance of a distributed key-value store that spans across all VDCs in the federation. The public or private key pair of a VDC is stored obfuscated in VNest on system disks. VNest is an ECS distributed key-value store implementation that spans across nodes in a single VDC.

When keys are persisted or retrieved, only the encrypted data is transported across nodes of a VDC or across VDCs. Decryption of key happens locally at each service that needs a particular key. Table 1 provides a brief explanation of the various keys that are used in native key management.

Note: Obfuscation (listed in the table) is a method in which the key is made unclear or unintelligible so that the key cannot be decoded or interpreted easily.

Table 1 ECS native management keys

Key name	Key type	Protected by	Description	Storage
VDC Public/Private Key	Key Pair	Obfuscation	Per VDC generated key pair using RSA 2048	Stored as obfuscated in VNest, a VDC wide KV store on node system disks.
Master Key	KEK	VDC Public Key	Randomly generated AES 256-bit key that is used with the RT Data Encryption key to create a Virtual Master key that protects rotation and namespace keys.	Unique per ECS Federation and is stored and wrapped using each VDC Public Key in the Resource Table (RT). The RT is a KV-store across all VDCs in the federation. A new Master key is generated every time a user requests a key rotation.
RT Data Encryption Key	KEK	VDC Public Key	Randomly generated AES 256-bit key that is used to create a Virtual Master Key.	Unique per ECS Federation and is stored as a wrapped key using each VDC Public Key in the Resource Table (RT). The RT is a KV-store across all VDCs in the federation.

Key name	Key type	Protected by	Description	Storage
Rotation Key	KEK	Virtual Master Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key that is used to create a Virtual Bucket key for wrapping object keys.	New rotation key that is generated and stored in an RT every time a user requests a key rotation.
Virtual Master Key	KEK	Does not require protection since it is not stored.	Is computed from the Master and RT Data Encryption Keys as needed.	This key is never persisted to disk.
Namespace Key	KEK	Virtual Master Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key per namespace that is used to wrap all bucket keys belonging to buckets in the namespace.	Unique per namespace and is stored as a wrapped key using the Virtual Master Key in the Resource Table.
Bucket Key	KEK	Namespace Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key per bucket that is used with the Rotation Key to generate a Virtual Bucket Key that is used to wrap all object keys.	Unique per bucket and is stored as a wrapped key using the namespace key in the Resource Table.
Virtual Bucket Key	KEK	Does not require protection since it is not stored.	Is computed from the Bucket and Rotation keys when needed.	This key is never persisted to disk.
Object Data Encryption Key	DEK	Virtual Bucket Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key per object that is used to encrypt object data.	Object Key is wrapped using the Virtual Bucket KEK in the Object Table (KV-store) in commodity disks.

2.2 External key management

ECS versions 3.3 and later support External Key Management (EKM) using external key managers that are Key Management Interoperability Protocol version 1.4 (KMIP v1.4) compliant. ECS delegates the storage and protection of top-level Key Encrypting Key (KEK), the master key to the external EKM. ECS 3.3 and later versions support Safenet KeySecure (Gemalto Safenet) and ECS 3.4 supports the IBM SKLM 3.0 (Security Key Lifecycle Manager). ECS 3.6 supports Safenet KeySecure 8.11 with Client Certificate Authentication only.

External key management also uses a hierarchy as shown in Figure 3. EKM flow is as native key management except that master is created in EKM. Like native key management, the Master key is used to derive the Virtual master key. Each namespace in ECS associates to a namespace key, and a virtual master key protects it. Virtual master key is a key that is derived from master key and RT data encryption key, and the virtual master key is never persisted to disks. All buckets within a namespace associates to a bucket key, and the corresponding namespace key protects the bucket key. Data for each object is encrypted using a unique object key, which is protected using a virtual bucket key. The Virtual bucket key is derived from bucket and rotation key and is not saved to disks.

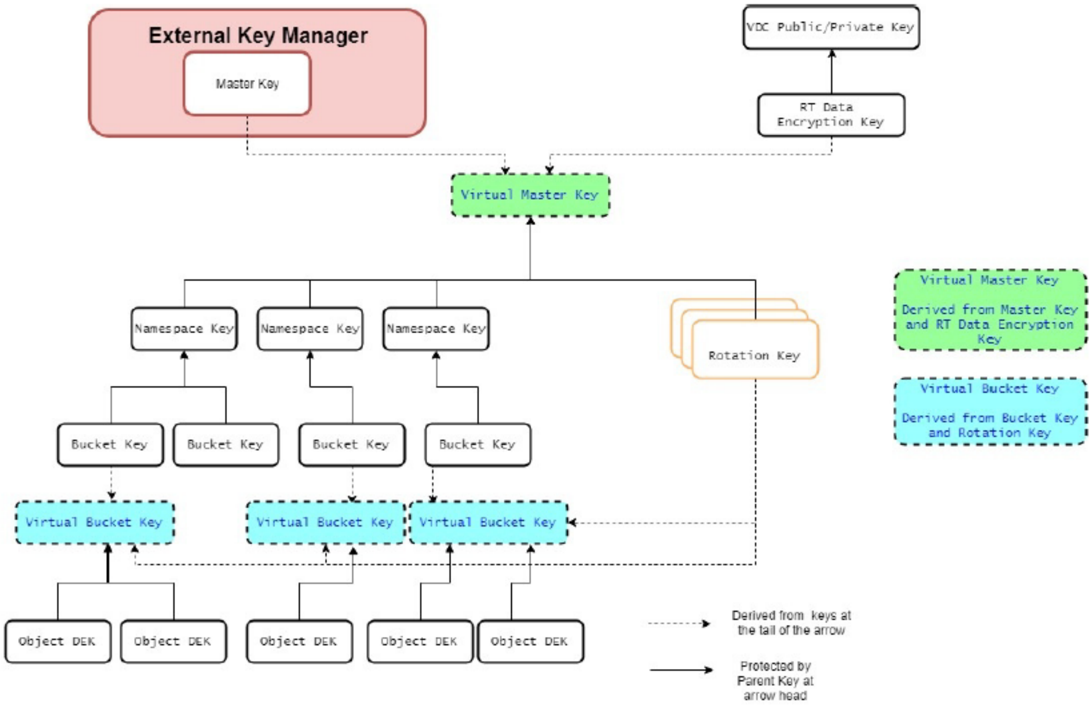


Figure 3 External key management key hierarchy

Figure 4 shows the location of EKM managed keys in ECS systems. The master key is generated and stored securely in the External Key Manager (EKM). The Namespace key and Rotation key are wrapped using AESKeyWrapRFC5649 by virtual master key. Bucket key is wrapped using AESKeyWrapRFC5649 by namespace key. These wrapped keys are stored in the Resource Table (RT) like the natively managed keys. The virtual bucket key wraps the object keys using AESKeyWrapRFC5649 and stores in object table.

The communication with EKMs is protected by SSL using server and client certificates. When ECS persists or retrieves the keys, only the encrypted data is transported across nodes of a VDC or across VDCs. Decryption of key happens locally at each service that requires a specific key.

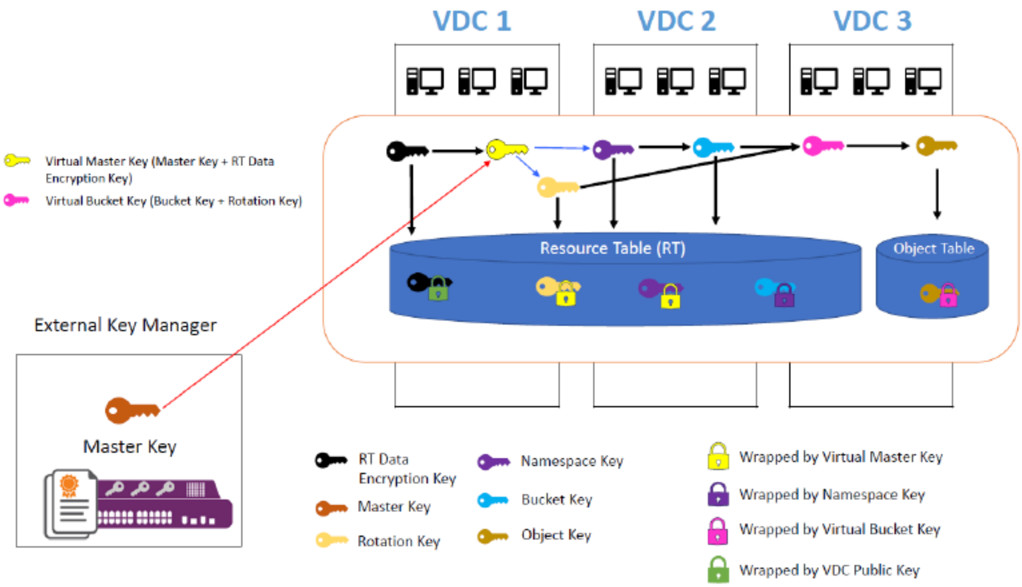

VDC 1
VDC 2
VDC 3
Virtual Master Key (Master Key + RT Data Encryption Key)
Virtual Bucket Key (Bucket Key + Rotation Key)
Resource Table (RT)
Object Table
External Key Manager
Master Key
RT Data Encryption Key
Master Key
Rotation Key
Namespace Key
Bucket Key
Object Key
Wrapped by Virtual Master Key
Wrapped by Namespace Key
Wrapped by Virtual Bucket Key
Wrapped by VDC Public Key

Figure 4 Storage Locations of EKM Managed keys

Table 2 provides a brief explanation of the various keys that are used in key management using EKM.

Table 2 ECS EKM management keys

Key name	Key type	Protected by	Description	Storage
Master Key	KEK	External Key Manager	AES 256-bit key that is generated by EKM and is used with the RT Data Encryption key to create a Virtual Master key that protects rotation and namespace keys.	Unique per ECS Federation and is created and stored in EKM. A new Master key is generated every time a user requests a key rotation.
Rotation Key	KEK	Virtual Master Key using AESKeyWrapRFC5649	AES 256-bit key that is generated by EKM and is used to create a Virtual Bucket key for wrapping object keys.	New rotation key that is generated and stored in ECS every time a user requests key rotation.
RT Data Encryption Key	KEK	VDC Public Key	Randomly generated AES 256-bit key used to create a Virtual Master Key.	Unique per ECS Federation and is stored as a wrapped key using each VDC Public Key in the Resource Table (RT). The RT is a KV-store across all VDCs in the federation.
Virtual Master Key	KEK	Does not require protection, as it is not stored.	Is computed from the Master and RT Data Encryption keys when required.	This key is never persisted to disk.
Namespace Key	KEK	Virtual Master Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key per namespace that is used to wrap all bucket keys belonging to buckets in the namespace.	Unique to each namespace and stored as a wrapped key using Virtual Master Key in the Resource Table.

Key name	Key type	Protected by	Description	Storage
Bucket Key	KEK	Namespace Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key per bucket that is used with the Rotation Key to generate a Virtual Bucket Key that is used to wrap all object keys.	Unique to each bucket and is stored as a wrapped key using the namespace key in the Resource Table.
Virtual Bucket Key	KEK	Does not require protection, as it is not stored.	Is computed from the Bucket and Rotation keys when required.	This key is never persisted to disk.
Object Data Encryption Key	DEK	Virtual Bucket Key using AESKeyWrapRFC5649	Randomly generated AES 256-bit key that is generated for each object and is used to encrypt object data.	Object Key is wrapped using the Virtual Bucket KEK in the Object Table (KV-store) in commodity disks.

2.3 User-supplied keys with the S3 API headers

With the S3 API, encryption keys can be specified in the header to encrypt objects. When an object is encrypted using user-supplied key, the key is never stored, only the hash of the key is stored in object table. The user must supply the encryption key every time an operation is performed on that object. ECS validates that the key provided for update, appends, and reads it as the same used during object creation.

Below is the example with user-supplied key:

```
> PUT /foobucket/fooobject HTTP/1.1
> User-Agent: curl/7.28.1
>Host: somehost.emc.com:9021
> Authorization: AWS user1:rYXxrNSrIW2d+apG3MjU4sAAzVs
> x-amz-server-side-encryption:AES256
>Content-Length: 15536
> PUT /foobucket/fooobject HTTP/1.1
> User-Agent: curl/7.28.1
>Host: somehost.emc.com:9021
>x-amz-server-side-encryption-customer-algorithm:AES256
> x-amz-server-side-encryption-customer-key-MD5:w79dwNhAgGtXei9fHOb+Gw==
> Content-Length: 15536
> Expect: 100-continue
```

2.4 Key rotation

ECS supports rotation of keys, which is a practice of changing keys to limit the amount of data that is protected by any given key to support industry standard practices. It can be performed on demand both through the API and user interface and is designed to minimize the risk from compromised keys.

- When the rotation task is initiated, it will rotate the master key in EKM or in ECS (if native) and the rotation key in ECS.
- When the keys are rotated, RotationKeyReWrapTaskScanner and NamespaceRewrapTaskScanner will also start.
- NamespaceRewrapTask will trigger rewrapping of all the namespaces on the system. Rewrapping the namespace means protecting the namespace key using the virtual master key.
- RotationKeyReWrapTask protects all existing rotation keys with the new virtual master key.
- In case of the EKM, there will be separate EKMBackgroundScanner that will look at old master key that is not in use and deactivate those keys in EKM. Once the keys are deactivated, a customer can delete those keys externally on EKM.

Once the EKM cluster activation or key rotation is initiated through the API or UI, the following events will occur through the EKMClusterActivationScanner, RotationTaskScanner, RotationKeyRewrapScanner, and NamespaceRewrapScanner as shown in Figure 5.

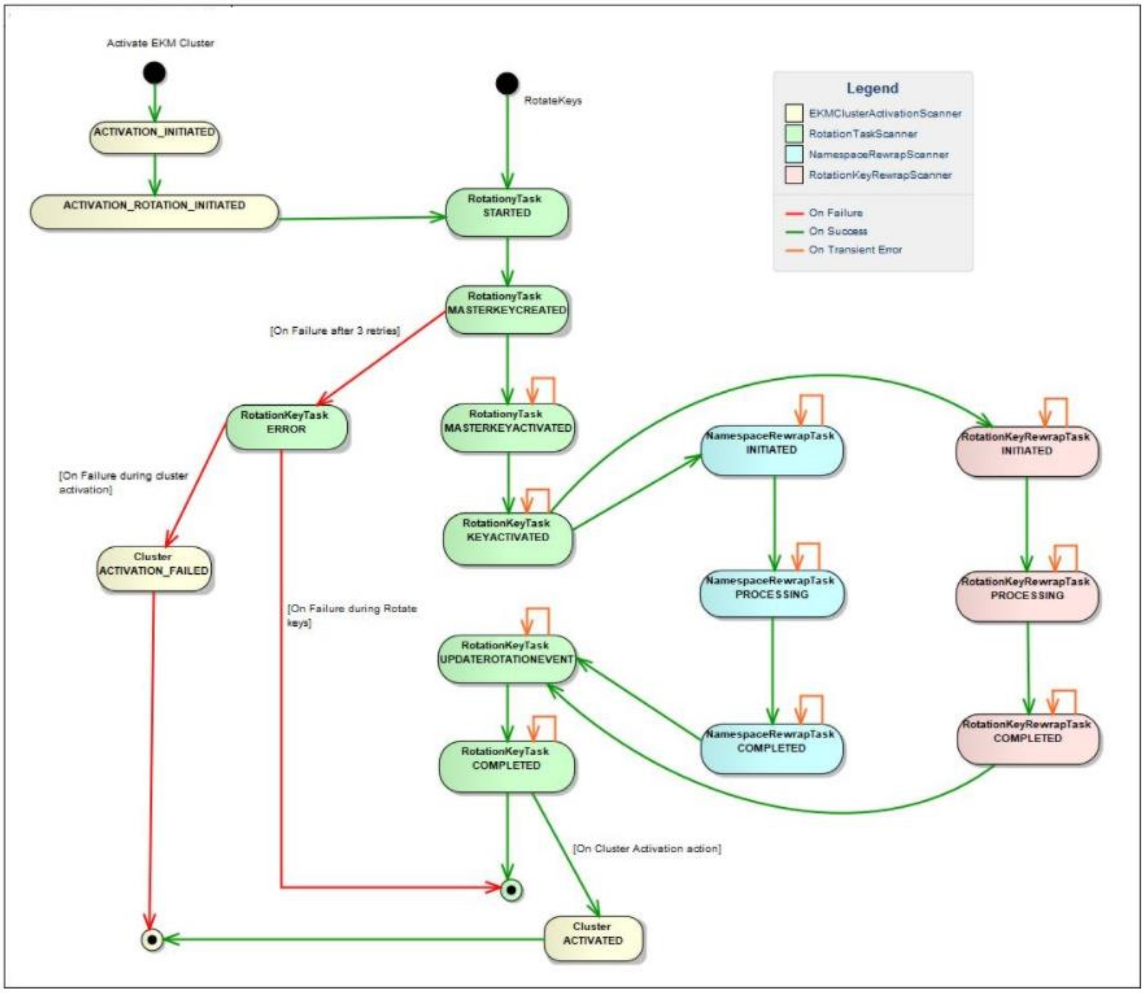

Activate EKM Cluster
ACTIVATION_INITIATED
ACTIVATION_ROTATION_INITIATED
RotateKeys
RotationyTask STARTED
RotationyTask MASTERKEYCREATED
[On Failure after 3 retries]
RotationKeyTask ERROR
RotationyTask MASTERKEYACTIVATED
[On Failure during cluster activation]
Cluster ACTIVATION_FAILED
RotationKeyTask KEYACTIVATED
[On Failure during Rotate keys]
RotationKeyTask UPDATEROTATIONEVENT
RotationKeyTask COMPLETED
[On Cluster Activation action]
Cluster ACTIVATED
NamespaceRewrapTask INITIATED
NamespaceRewrapTask PROCESSING
NamespaceRewrapTask COMPLETED
RotationKeyRewrapTask INITIATED
RotationKeyRewrapTask PROCESSING
RotationKeyRewrapTask COMPLETED
Legend
EKMClusterActivationScanner
RotationTaskScanner
NamespaceRewrapScanner
RotationKeyRewrapScanner
On Failure
On Success
On Transient Error

Figure 5 Cluster activation state

The system does the following process:

- Creates master key in EKM (For native rotate keys, this creates a master key in ECS)
- Creates rotation key in ECS
- Updates ActiveMasterKeyRecord to point to new MasterKeyRecord created
- Updates ActiveRotationKeyRecord to point to new RotationKey Record
- Initiates RotationKeyReWrapTask and NamespaceRewrapTask.
- RotationKeyReWrapTask protects all existing rotation keys with new virtual master key (Derived from Master Key and RT data encryption Key).
- NamespaceRewrapTask will trigger rewrapping of the all the namespaces on the system; rewrapping the namespace means protecting the namespace key using the virtual master key
- In case of master key creation failure, there will be maximum three retries which will pick up the state it failed.
  - Failures in the other state will lead to a TRANSIENT_ERROR state, which is expected to complete in a retry. When in TRANSIENT_ERROR, the state machine will start from the last

saved state. For example, if the last saved state is MASTERKEYACTIVATED and TRANSIENT_ERROR, then the next state performed will be KEYACTIVATED.

Note: Switch from EKM to Internal ECS Key Management, which must be done only with support help through dtquery.

2.5 Encryption of the master key in a geo-replicated environment

This scenario applies when a system is being added to form or extend a federation that generates public or private keys locally. These keys are used for encryption or decryption of the federation's master key. On a federation, the new system that does not know the master key stores the public key in resource table. A VDC that knows the master key uses this public key to encrypt and share the encrypted key with new system. The master key is now global and known to both systems within the federation.

As shown in Figure 6, the master key is global and is known to both systems within the federation. The ECS system that is labeled VDC 2 joins the federation. The master key of VDC 1 (the existing system) is extracted and passed to VDC 2 for encryption with the public-private key randomly generated by VDC 2.

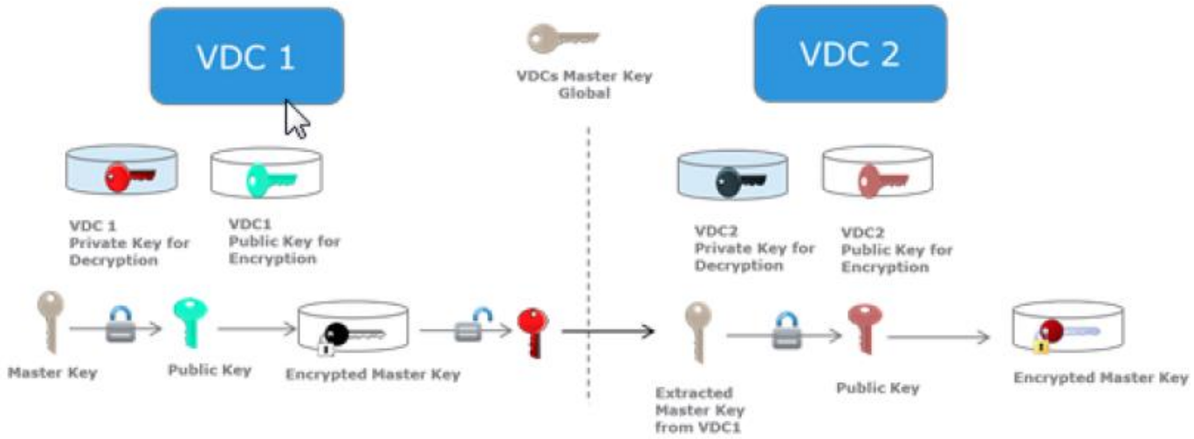

VDC 1
VDCs Master Key Global
VDC 2
VDC 1 Private Key for Decryption
VDC1 Public Key for Encryption
VDC2 Private Key for Decryption
VDC2 Public Key for Encryption
Master Key
Public Key
Encrypted Master Key
Extracted Master Key from VDC1
Public Key
Encrypted Master Key

Figure 6 Encryption of the master key in a geo-replicated environment

2.6 PSO/TSO considerations

All new keys and KEK records are stored in the RT. We will not support rotation of keys during a TSO. If TSO happens during key rotation, then the key rotation task will be suspended until the system comes out of TSO. If the ActiveRotationKey has not been changed, the old rotation key will be used but once it is changed, and the new one will be used irrespective of whether the rotation task history or state has been changed. Like TSO, we will not support key rotation while PSO is in progress. Hence, the system has to come out of PSO before key rotation will be enabled. If a PSO happens during rotation, the task will fail immediately.

3 Performance

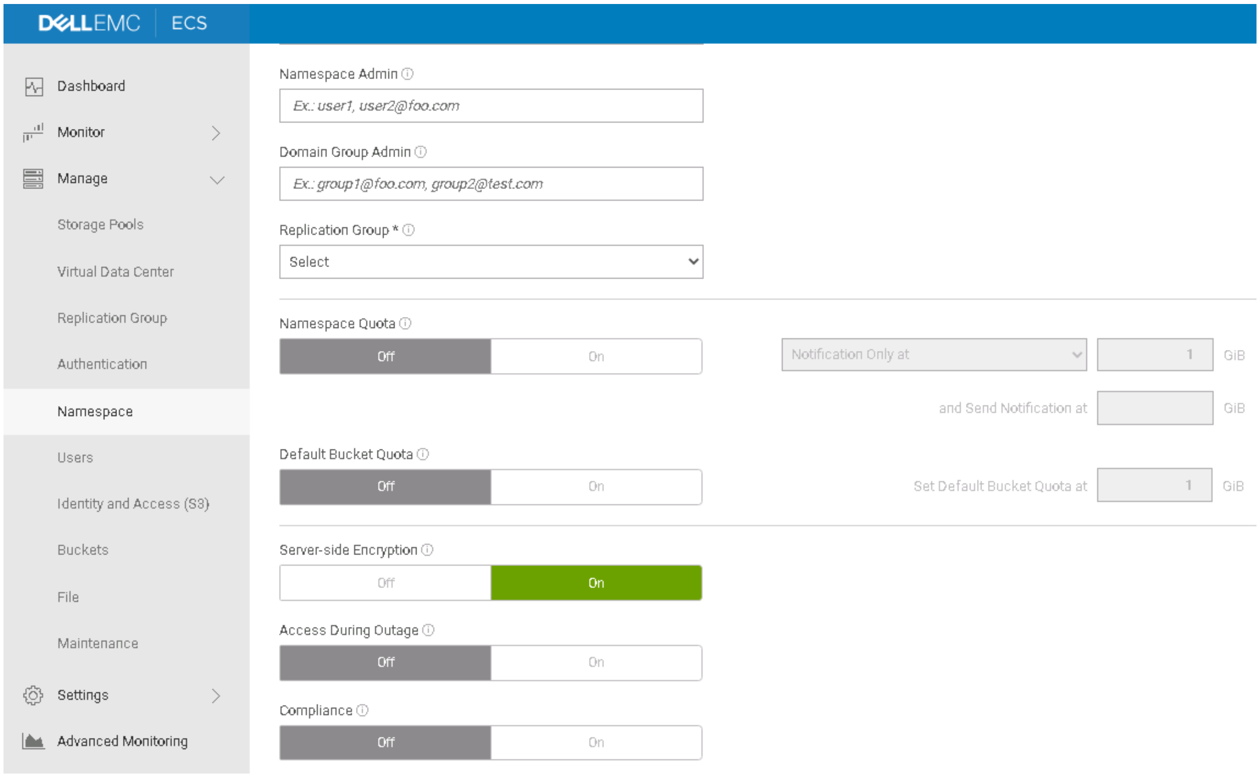
From the data-path perspective, every read/write path for system encrypted objects will incur some extra cycles since we now have to obtain the virtual wrapping key. Currently, we use the bucket key as the wrapping key. With this change, the virtual wrapping key is derived from the bucket key and the rotation key will be used. The rotation key will be cached after the first call to get it, and it will be available to all reads/writes after that. Also, the bucket key will likely be available in the cache (this behavior is the same as before). The step to derive the virtual wrapping key from the bucket and rotation key will be the extra performance penalty incurred during reads/writes, resulting in an approximately 10% performance penalty on S3 TPS when D@RE is enabled.

Note: For more information about the ECS performance, see the ECS performance paper which is internal only and can be shared with partners and customers under NDA.

4 D@RE configuration

4.1 ECS Portal configuration

The Create namespace as shown in Figure 7 and create bucket as shown in Figure 8 include a Server-Side Encryption On/Off control. Server-side encryption can only be enabled during namespace or bucket creation and cannot be enabled or disabled later.



Namespace Admin ⓘ

Ex.: user1, user2@foo.com

Domain Group Admin ⓘ

Ex.: group1@foo.com, group2@test.com

Replication Group * ⓘ

Select

Namespace Quota ⓘ

Off | On

Notification Only at | 1 | GiB

and Send Notification at | | GiB

Default Bucket Quota ⓘ

Off | On

Set Default Bucket Quota at | 1 | GiB

Server-side Encryption ⓘ

Off | On

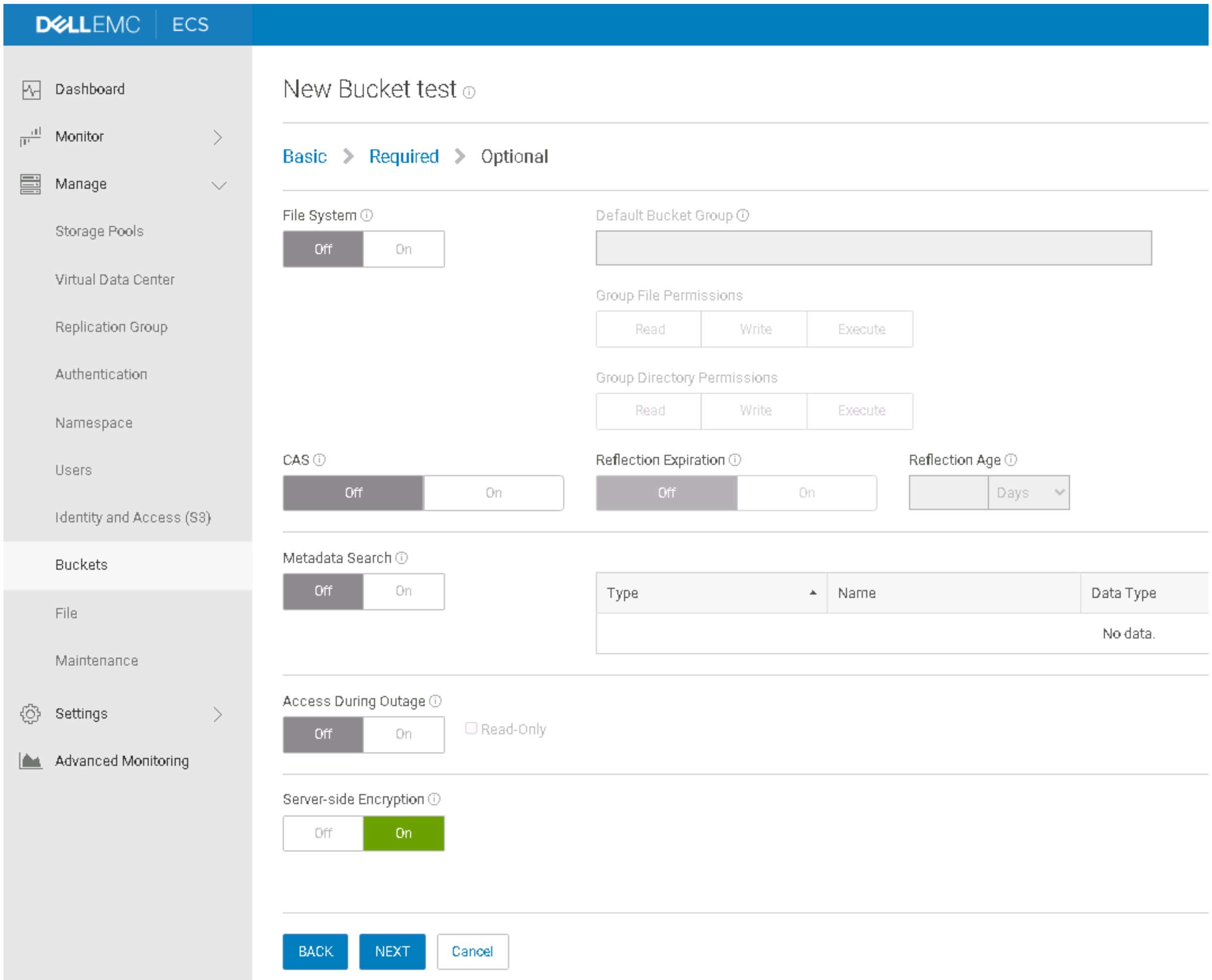
Access During Outage ⓘ

Off | On

Compliance ⓘ

Off | On

Figure 7 Enable D@RE in namespace



New Bucket test

File System: Off | On

Default Bucket Group

Group File Permissions: Read | Write | Execute

Group Directory Permissions: Read | Write | Execute

CAS: Off | On

Reflection Expiration: Off | On

Reflection Age: Days

Metadata Search: Off | On

Type	Name	Data Type
		No data.

Access During Outage: Off | On □ Read-Only

Server-side Encryption: Off | On

Figure 8 Enable D@RE in bucket

Note: When you turn encryption on when the bucket is created, this feature cannot be turned off later.

4.2 ECS management API configuration

Customers can also use the ECS management API to specify the D@RE in the namespace and bucket. To enable the encryption by the following commands:

```
object_namespace_create with the
<is_encryption_enabled>true</is_encryption_enabled> parameter

object_bucket_create with the
<is_encryption_enabled>true</is_encryption_enabled> parameter
```

The following example shows an `object_bucket_create` example:

```
>POST /object/bucket HTTP/1.1
> "<object_bucket_create> <name>bucket_for_test</name>
<vpool>urn:storageos:ReplicationGroupInfo:b3bf2d47-d732-457c-bb9b-
d260eb53a76b:global</
vpool>
```

```
<filesystem_enabled>false</filesystem_enabled> <head_type>s3</head_type>
<namespace>s3</
namespace>
<is_encryption_enabled>true</is_encryption_enabled>
<is_stale_allowed>false</is_stale_allowed> <retention>100</retention> </
object_bucket_create>
```

The S3 API support for D@RE includes these object-level encryption abilities:

- Create an S3 bucket with encryption enabled
- Create, update, or read an encrypted object with a system-generated key
- Create, update, or read an encrypted object with a user-supplied key
- Check the get response to see the encryption status

For more information about the ECS API configuration part, see the ECS API guide.

4.3 External key manager configuration

System administrators can add a cluster, view VDC EKM mapping information, and rotate keys on the **Settings > Key Management** page in the ECS Portal as shown in Figure 9.

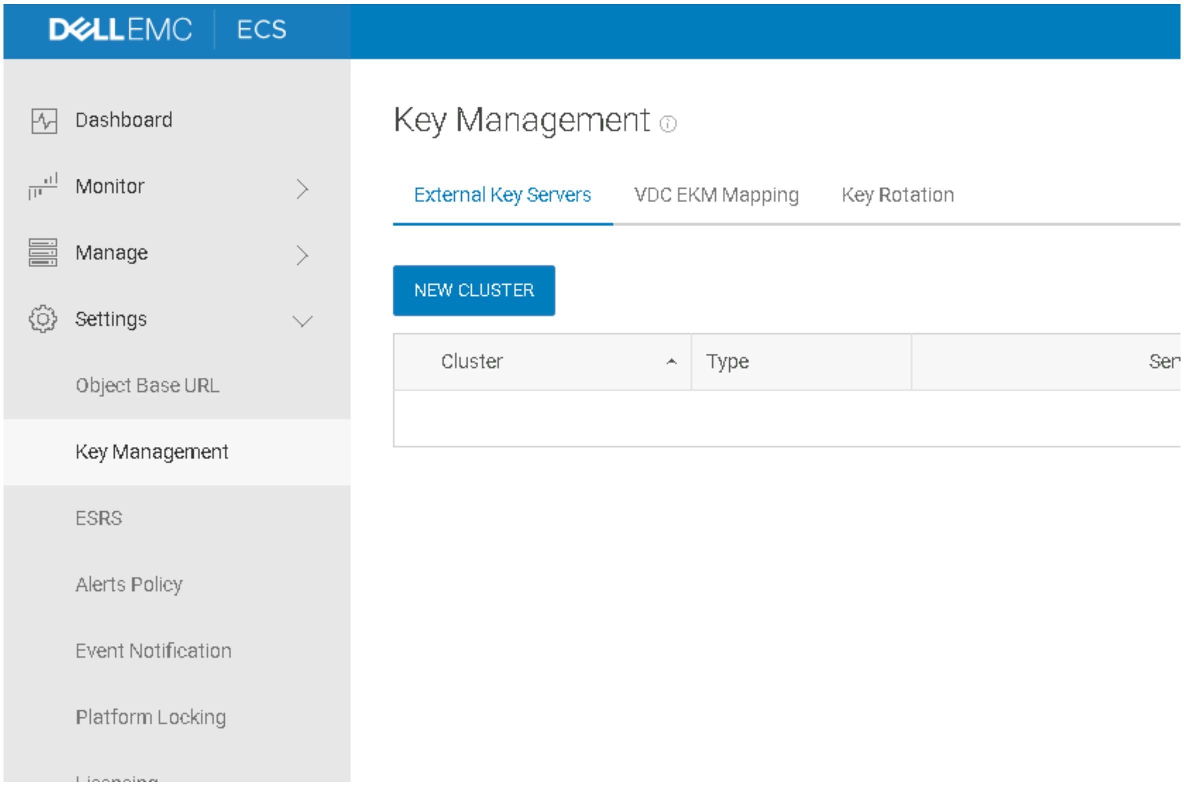
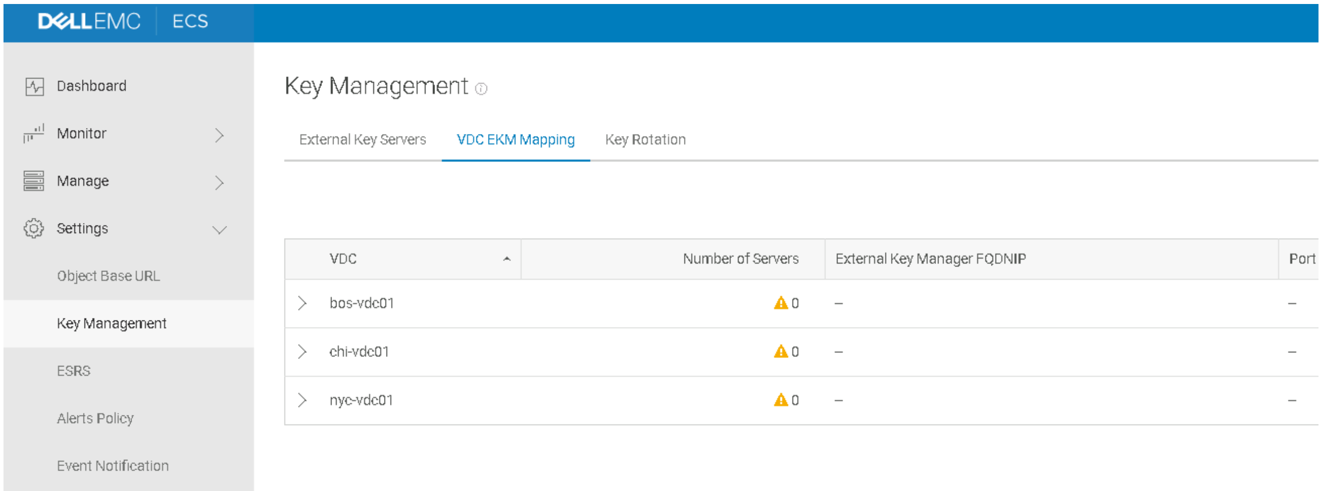


Figure 9 External key manager configuration

For more detailed steps about how to configure the EKM, see the ECS administrator guide.

4.4 VDC EKM mapping

As shown in Figure 10, VDC EKM mapping assigns a subset of a cluster member server to a VDC so that nodes in the VDC can use them to access cryptographic keys.



VDC	Number of Servers	External Key Manager FQDNIP	Port
bos-vdc01	⚠ 0	–	–
chi-vdc01	⚠ 0	–	–
nyc-vdc01	⚠ 0	–	–

Figure 10 VDC EKM mapping

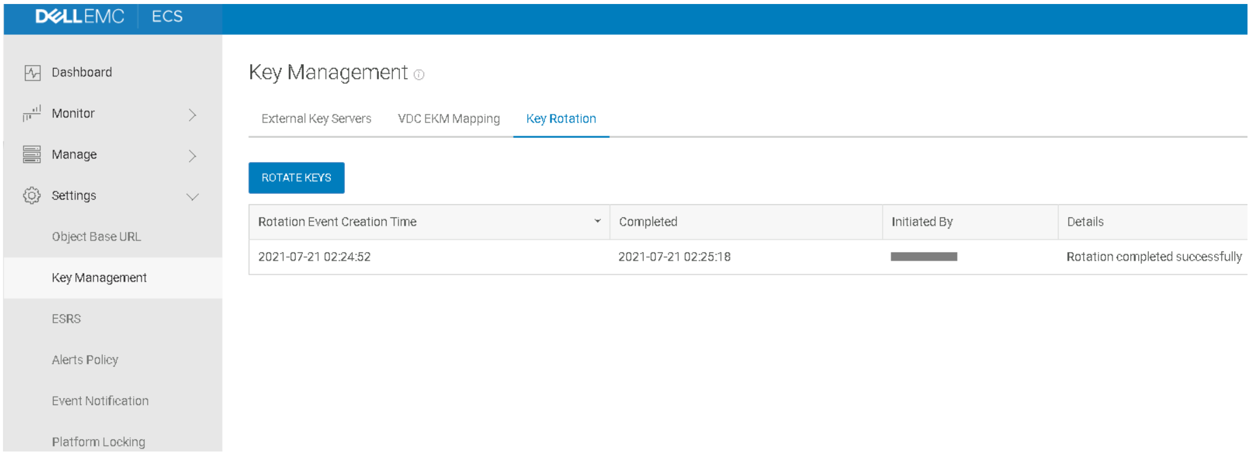
For more details steps about how to configure the EKM mapping, see the ECS administrator guide.

Note: At least two key servers should be added before proceeding to VDC EKM Mapping. Activation is not enabled without a minimum of two key servers. Once a key server is mapped, the delete option is disabled. You must remove the EKM mapping to delete the key server.

4.5 Key rotation

ECS supports rotation of keys, a practice of changing keys to limit the amount of data that is protected by any given key to support industry standard practices. It can be performed on-demand both through the API and user interface and is designed to minimize the risk from compromised keys.

To initiate key rotation, click **Settings > Key Management > Key Rotation > Rotate Keys** (Figure 11).



Rotation Event Creation Time	Completed	Initiated By	Details
2021-07-21 02:24:52	2021-07-21 02:25:18	[redacted]	Rotation completed successfully

Figure 11 Key rotation

Rotation is an asynchronous operation, and the latest status of current operation can be seen in the table. The Rotate Keys table also lists the status of previous rotation operations.

A 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 EMC storage platforms.