GUIDE

STORMSHIELD DATA SECURITY FOR KUBERNETES KMS PLUGIN

ADMINISTRATION GUIDE

Version 2.0

Document last updated: November 7, 2025

Reference: sds-en-sds-kubernetes-kms_plugin_guide-v2

Table of contents

1. Getting started

SDS for Kubernetes KMS Plugin is a solution that enables the implementation of a Kubernetes Key Management Service (KMS) provider, as described in the Kubernetes documentation.

With SDS for Kubernetes KMS Plugin, sensitive data stored in Kubernetes etcd database is protected by robust at-rest encryption. This solution secures your infrastructure against a wide range of threats, including physical attacks, by protecting your most sensitive data.

It is based on three technologies:

- **Kubernetes KMS provider v2** which encrypts data in etcd using Data Encryption Keys (DEKs).
- **SDS Key Management as a Service** (KMaaS) with enabled Key Access Service (KAS) feature: it offers an API for encrypting/decrypting DEKs.
- **Stormshield Software Development Kit** (SDK) that provides higher-level and standardized methods to encrypt/decrypt data using DEKs.

1.1 Kubernetes KMS provider

The Kubernetes KMS provider v2 uses envelope encryption scheme where data stored in etcd is encrypted with Data Encryption Keys (DEKs) generated from a secret seed combined with random data.

It encrypts data at rest in etcd, including Secrets, ConfigMaps, and custom resources defined in CustomResourceDefinitions. Resources are encrypted on write operations. The encrypted data is stored in etcd prefixed with *k8s:enc:kms:v2:* to indicate KMS v2 encryption. During startup, the Kubernetes API server performs *DecryptRequest* procedure calls to fill its watch cache.

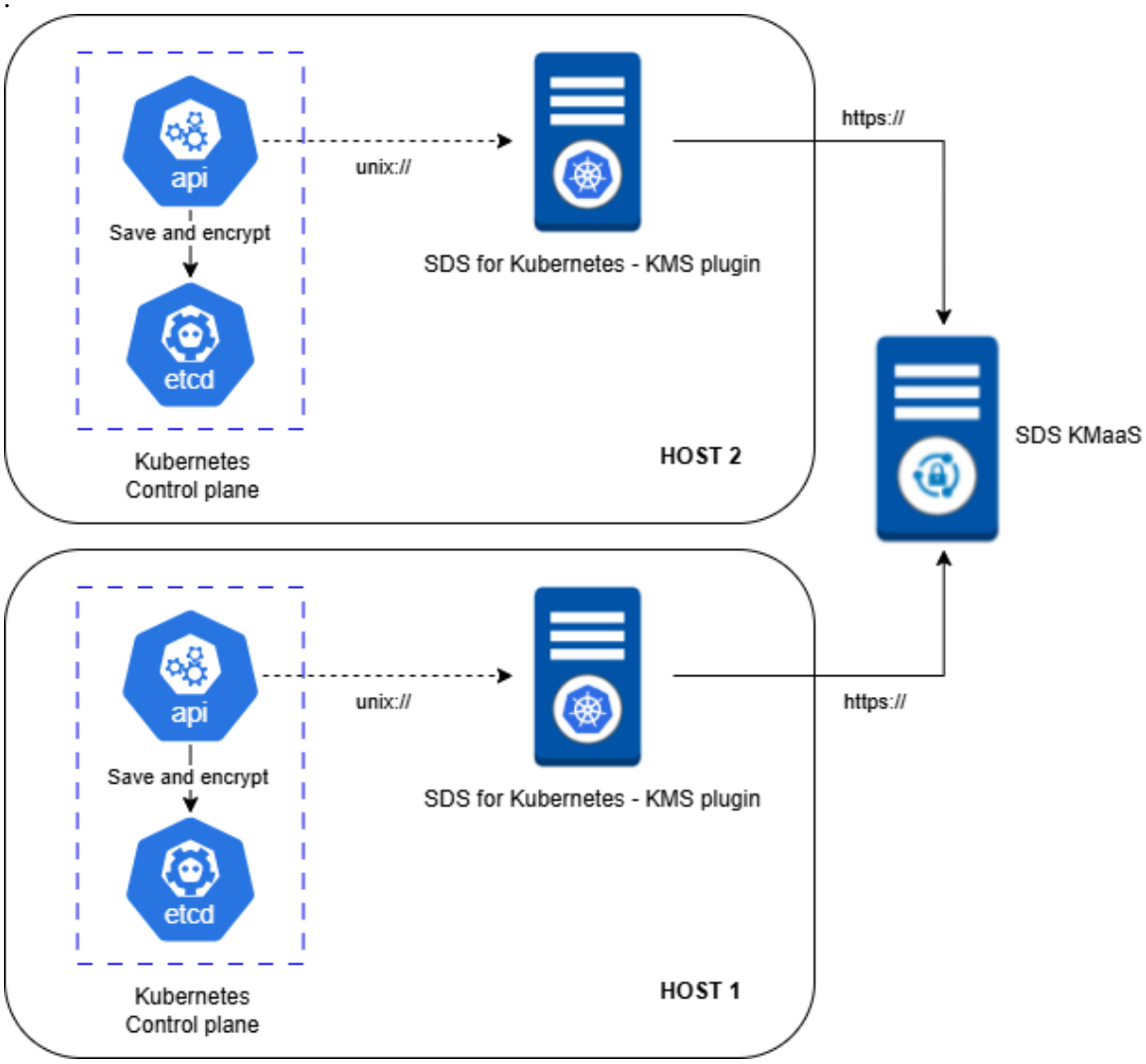
Kubernetes KMS provider requires a KMS Plugin to encrypt the seed used to generate the DEKs.

The workflow within the KMS provider is as follows:

1. When the Kubernetes control plane starts, it generates a seed, then sends it to SDS for Kubernetes KMS Plugin via gRPC over Unix sockets.
2. SDS for Kubernetes KMS Plugin generates a DEK and encrypts the seed.
3. The DEK is sent to the KMaaS via HTTPS to be encrypted by a Key Encryption Key (KEK).
4. The encrypted seed and encrypted DEK are stored in etcd. It will be used to generate a DEK each time the Kubernetes API server needs to encrypt data.

1.2 Architecture diagram

This simplified diagram illustrates a high-availability Kubernetes configuration with two control plane hosts. Each control plane runs a Kubernetes API server that communicates with a local SDS for Kubernetes KMS Plugin via a Unix socket (unix://).



NOTE
The use of the solution in any way other than as described in the documentation is not managed. Alternatively, get in touch with Stormshield Support for clarification.

2. Understanding the requirements

2.1 Security requirements

2.1.1 Environment

- The server on which SDS for Kubernetes KMS Plugin is installed must be healthy. There must be an information system security policy whose requirements are met on the server. This policy shall ensure that the installed software is regularly updated and the system is protected against viruses and spyware or malware (firewall properly configured, antivirus updates, etc.). It is imperative to follow the operating system security recommendations issued by the ANSSI in their document *ANSSI-BP-028-EN*.
- Access to the administrative functions of the workstation system is restricted only to system administrators.
- The operating system must manage the logs generated by the product in accordance with the security policy of the company. It must for example restrict read access to these logs to only those explicitly permitted. For more information, refer to the section To meet the logging requirements for SDS for Kubernetes KMS Plugin, you must follow the security recommendations for logging systems issued by the ANSSI in their document ANSSI-PA-012 (French only)..
- SDS for Kubernetes KMS Plugin must be installed on a server whose system and OpenSource contributions are kept up to date.
- The server hosting the solution must be located in a secure physical environment with access control protocols and must be trusted.

2.1.2 Administration

- SDS for Kubernetes KMS Plugin administrators are considered as trusted. They are responsible for defining SDS for Kubernetes KMS Plugin security policy in compliance with the state-of-the-art standards.
- The system administrator is also considered as trusted. He/She is responsible for the installation and maintenance of the application and server. He/She applies the security policy defined by SDS for Kubernetes KMS Plugin administrators.

2.1.3 KMaaS

Stormshield recommends having a dedicated tenant for SDS for Kubernetes KMS Plugin to allow the API key to be revoked without interfering with other applications. We also recommend using only one API key per instance of SDS for Kubernetes KMS Plugin.

2.2 Infrastructure requirements

The SDS for Kubernetes KMS Plugin requires:

- A Kubernetes cluster version v1.29 or higher with etcd v3 or higher. For more information about Kubernetes requirements for the KMS provider, refer to the Kubernetes documentation,

- An available KMaaS instance version 4.5 or higher. The instance requires:
  - A base KMaaS URL,
  - A tenant identifier with enabled KAS feature,
  - An API key for authentication.

  For more information, refer to the KMaaS Guide.
- Node.js 20 or higher.
- An operating system supporting Unix sockets.
- The following network streams must be open:

Description	Protocol	Source	Base URL	Port	Route
KMaaS KAS decrypt endpoint	HTTP SDS for Kubernetes KMS Plugin must be able to reach the KMaaS kas/decrypt route in order to decrypt the DEK.	SDS for Kubernetes KMS Plugin	<KMAAS_URL>	<KMAAS_PORT>	api/v1/<tenant_id>/kas/decrypt
KMaaS KAS encrypt endpoint	HTTP SDS for Kubernetes KMS Plugin must be able to reach the KMaaS kas/encrypt route in order to encrypt the DEK.	SDS for Kubernetes KMS Plugin	<KMAAS_URL>	<KMAAS_PORT>	api/v1/<tenant_id>/kas/encrypt

Contact your KMaaS administrator to get the KMaaS URL and port.

3. Deploying SDS for Kubernetes KMS Plugin

The SDS for Kubernetes KMS Plugin is provided as a compressed archive containing:

- The Node.js server for SDS for Kubernetes KMS Plugin: *stormshield-kubernetes-kms-plugin.cjs*,
- The *sbom.json* file containing a list of all the dependencies and libraries used by *stormshield-kubernetes-kms-plugin.cjs*,
- The gRPC service definition file: *api.proto*, which contains the API configuration that SDS for Kubernetes KMS Plugin must use to communicate with the Kubernetes control plane,
- The LICENSE file containing the SDS for Kubernetes KMS Plugin terms and conditions.

You must deploy SDS for Kubernetes KMS Plugin on each host where a Kubernetes control plane is installed.

3.1 Creating the SDS for Kubernetes KMS Plugin service

Stormshield recommends running the server SDS for Kubernetes KMS Plugin as a systemd service.

1. Create a new file */etc/systemd/system/sds-kms-plugin.service* from the following sample:

```
[Unit]
Description=Stormshield Data Security for Kubernetes KMS Plugin

[Service]
ExecStart=node <PATH_TO_SCRIPT>/stormshield-kubernetes-kms-
plugin.cjs
Restart=always
StandardOutput=journal
StandardError=journal
SyslogIdentifier=sds-kms-plugin

Environment="KMAAS_URL=<KMAAS_URL>"
Environment="KMAAS_TENANT_ID=<KMAAS_TENANT_ID>"
Environment="KMAAS_API_KEY=<KMAAS_API_KEY>"
Environment="PROTO_FILE_PATH=<PROTO_FILE_PATH>"
# Environment="BIND_STRING=<BIND_STRING>"
# Environment="KEY_ID=<KEY_ID>"
# Environment="GRPC_VERBOSITY=DEBUG"

[Install]
WantedBy=multi-user.target
```

   The file must have permission 600 to ensure that the API key is secure.

2. Indicate the path to the *stormshield-kubernetes-kms-plugin.cjs* script by replacing the <PATH_TO_SCRIPT> string.

3. Set the following environment variables by replacing the strings between tags:

Environment variable	Description	Mandatory/Optional
KMAAS_URL	URL of the KMaaS.	Mandatory
KMAAS_TENANT_ID	ID of the tenant declared in the *tenant_id* parameter of the KMaaS *config.json* file.	Mandatory
KMAAS_API_KEY	Base64 string of the concatenated values `_KAS_AUTHENTICATION_API_KEY_NAME:_KAS_AUTHENTICATION_API_KEY_VALUE` of the KMaaS *config.json* file. On Unix-like systems, generate the KMaaS API key with this command: `$ echo -n "<API_KEY_NAME>:<API_KEY_PASSWORD>" | base64` **NOTE** Stormshield recommends that you use one different API key per control plane.	Mandatory
PROTO_FILE_PATH	Local path to the *api.proto* file provided by SDS for Kubernetes KMS Plugin. Example: "/mnt/shared/api.proto"	Mandatory
BIND_STRING	Path to the Unix socket used to communicate with Kubernetes. If a socket already exists, it will be deleted and then recreated. Default value: "unix:///tmp/sds-kms-plugin.sock"	Optional
KEY_ID	Identifier used for key rotation. The KEY_ID value must be less than 1 KB and must contain only alphanumeric characters. Default value: "stormshield-kms-plugin-key-id" For more information, see Configuring seed rotation.	Optional
GRPC_VERBOSITY	Enables gRPC server debug logs. Default value: "NONE" For more information, see Setting gRPC dependencies logging verbosity.	Optional

3.2 Configuring seed rotation

A new seed is generated by Kubernetes each time the cluster starts. If you want to rotate the seed more frequently, modify the value of the KEY_ID environment variable. The value of this variable is an alphanumeric string. If Kubernetes detects a change in this value, then it generates a new seed. For more information and best practices, see the Kubernetes documentation.

Stormshield recommends modifying the KEY_ID variable value each time the Key Encryption Key (KEK) of your KMaaS is rotated.

3.3 Running the SDS for Kubernetes KMS Plugin service

1. Start the SDS for Kubernetes KMS Plugin service using the following systemd commands:

```
systemctl enable sds-kms-plugin
systemctl start sds-kms-plugin
```

2. Check the server status using the following command:

```
systemctl status sds-kms-plugin
```

3.4 Uninstalling the SDS for Kubernetes KMS Plugin service

1. If you need to uninstall the SDS for Kubernetes KMS Plugin, run the following systemd commands:

```
systemctl stop sds-kms-plugin
systemctl disable sds-kms-plugin
```

2. Delete the service file with the following command:

```
rm /etc/systemd/system/sds-kms-plugin.service
```

4. Configuring the Kubernetes cluster

Once SDS for Kubernetes KMS Plugin has been deployed and started, you must configure each control plane of your Kubernetes cluster to enable the encryption via the KMS provider.

WARNING
The Kubernetes configuration provided in this section is an example for version 1.32, please refer to the Kubernetes documentation to customize your configuration:
https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
https://kubernetes.io/docs/tasks/administer-cluster/kms-provider/

4.1 Configuring encryption

- Create the following */opt/kms/kube-encryption.yaml* file to enable the use of SDS for Kubernetes KMS Plugin.

```
apiVersion: apiserver.config.k8s.io/v1
kind: EncryptionConfiguration
resources:
  - resources:
    - secrets
    - configmaps
       providers:
      - kms:
        apiVersion: v2    # SDS for Kubernetes KMS plugin supports only v2
        name: SDSKMSPlugin
        endpoint: unix:///opt/kms/kube-kms-plugin.sock   # Same path as BIND_STRING
on server
        timeout: 3s (entre KMS plugin et API server)
      - identity: {}
```

4.2 Updating the Kubernetes API server configuration

- Update the */etc/kubernetes/manifests/kube-apiserver.yaml* file to configure the Kubernetes API server to use the KMS provider.

Add the lines commented with "Add this line" in the example below:

```
...
spec:
  containers:
  - command:
    - kube-apiserver
    ...
    - --encryption-provider-config=/opt/kms/kube-encryption.yaml  # Add this line
with path to kube-encryption.yaml file
    - --encryption-provider-config-automatic-reload=true          # Add this line

    ...
    volumeMounts:
    - mountPath: /opt/kms      # Add this line
      name: opt-kms          # Add this line
  ...
  volumes:
  - hostPath:                     # Add this line
      path: /opt/kms           # Add this line
```

```
    type: DirectoryOrCreate  # Add this line
  name: opt-kms              # Add this line
```

5. Managing logs

SDS for Kubernetes KMS Plugin generates logs for every operation, making it possible to trace all operations performed and potential issues. It is a technical activity essential to the security of information systems.

5.1 Logging prerequisites

To meet the logging requirements for SDS for Kubernetes KMS Plugin, you must follow the security recommendations for logging systems issued by the ANSSI in their document *ANSSI-PA-012* (French only).

5.2 Accessing logs

Run the following journalctl command to access SDS for Kubernetes KMS Plugin logs:

```
journalctl -fu sds-kms-plugin
```

5.3 Setting gRPC dependencies logging verbosity

Some underlying dependencies used for gRPC communication may issue error-level logs. For example, in case of permission issues on the socket directory, you may get messages such as:

```
> E No address added out of total 1 resolved
```

To prevent such messages from polluting your log output, the *GRPC_VERBOSITY* environment variable is automatically set to *NONE* by default when launching SDS for Kubernetes KMS Plugin.

If you want to keep these logs (e.g., for debugging purposes), set the *GRPC_VERBOSITY* environment variable to *DEBUG* in the *sds-kms-plugin.service* file. For more information, refer to Creating the SDS for Kubernetes KMS Plugin service.

5.4 Generic log fields

The following fields are displayed for all logs generated by SDS for Kubernetes KMS Plugin in the order shown in the table.

Field	Description	Type
timestamp	Date and time at which the log was created. In UTC format. Example: "2025-05-21T08:54:17.262"	String in ISO 8601 format
severity	Level of severity of the log. Prescribed values: • *info*: Normal operation information message that requires no action, • *err*: Action failed.	String
application_version	Application version. Example: "1.0.0"	

Field	Description	Type
kind	Log family to which the log belongs. Prescribed value: • *domain*: SDS for Kubernetes KMS Plugin business operation logs.	String
category	Log category. Example: • *keys*: Logs related to the the cryptographic operations of SDS for Kubernetes KMS Plugin.	String
action	Event that occurred. Example: • start	String

The fields in the *error* block described below are displayed for all logs generated by the SDS for Kubernetes KMS Plugin in the event of an error when executing the action:

Field	Description	Type
error.code	Error identifier.	String
error.message	Error message.	String

5.5 Domain - Business operation logs

The log fields described below relate to business operations performed by SDS for Kubernetes init container. They belong to the *Domain* log family (Kind:domain).

5.5.1 *application* category

This category of logs relates to the operation of SDS for Kubernetes KMS Plugin.

start action

- The *start* action means that a *start* request has been made. This is the case whenever the SDS for Kubernetes KMS Plugin server is started.

This action generates an "info" severity log in the event of success, or an "err" severity log in the event of an error.

The log fields for this action are as follows:

Field	Description	Type
socket	Path of the Unix socket. Example: "unix:///tmp/sds-kms-plugin.sock"	String
protofile_path	Path to the *api.proto* file. Example: "/mnt/shared/api.proto"	String

Field	Description	Type
kmaas_url	URL of the KMaaS. Example: "https://host-kmaas:443"	String
tenant_id	ID of the KMaaS tenant. Example: "1961bc95-8ffb-48b2-92c7-6d93ee1cdae9"	String
kub_key_id	Value of the key id for seed rotation. Example: "stormshield-kms-plugin-key-id"	String

status action

- The *status* action means that a *status* request has been made. This is the case whenever Kubernetes checks if the SDS for Kubernetes KMS Plugin server is up and ready.

This action generates an "info" severity log in the event of success, or an "err" severity log in the event of an error.

The log field for this action is as follows:

Field	Description	Type
kub_key_id	Value of the key id for seed rotation. Example: "https://kubernetes.io/docs/tasks/administer-cluster/kms-provider/#developing-a-kms-plugin-gRPC-server-notes-kms-v2"	String

stop action

- The *stop* action means that a *stop* request has been made. This is the case whenever the SDS for Kubernetes KMS Plugin server stops.

This action generates an "info" severity log in the event of success, or an "err" severity log in the event of an error.

This action generates only generic logs. For more information, refer to Generic log fields.

5.5.2 *keys* category

This category of logs contains all the cryptographic operations.

encrypt action

- The *encrypt* action means that a *encrypt* request has been made. This is the case whenever the SDS for Kubernetes KMS Plugin encrypts data.

This action generates an "info" severity log in the event of success, or an "err" severity log in the event of an error.

The log field for this action is as follows:

Field	Description	Type
url	KMaaS URL used to encrypt the DEK.	String

decrypt action

- The *decrypt* action means that a *decrypt* request has been made. This is the case whenever the SDS for Kubernetes KMS Plugin decrypts data.

This action generates an "info" severity log in the event of success, or an "err" severity log in the event of an error.

The log field for this action is as follows:

Field	Description	Type
url	KMaaS URL used to decrypt the DEK.	String

6. Further reading

Additional information and responses to questions you may have about high availability are available in the Stormshield knowledge base (authentication required).

documentation@stormshield.eu