KeySafe 5

Keysafe 5 v1.4 Quick Start Guide

16 October 2024

Table of Contents

1. Quick Start Guide

1.1. Overview and prerequisites

The following steps provide a quick-start guide to installing KeySafe 5 and its dependencies using the provided deploy script.

The script is designed to be run on UNIX/Linux based systems by a non-root user. The script may call `sudo` as required.

These steps install KeySafe 5 and its dependencies. The included deploy script (`deploy.sh`) will provide a substitute installation for the various dependencies but they are only suitable for evaluation purposes, and should *not* be used for production environments.

Please see Hardening The Deployment for steps to harden the deployment. Entrust recommends these steps as a minimum and that additional hardening may be required dependent on your own requirements.

A production deployment will have as a minimum the following:

- Maintained and patched versions of all the dependencies
- A secure CA with TLS v1.3 support for certificates. The deploy script can provide a local insecure CA.
- A secure Kubernetes installation. The deploy script can install K3s locally.
- A secure MongoDB database. The deploy script can provide a replicated MongoDB with X.509 authentication running in Kubernetes.
- A secure RabbitMQ message broker. The deploy script can provide a RabbitMQ with X.509 authentication running in Kubernetes.
- A secure means of large object storage. The deploy script can provide local object storage within the Kubernetes cluster or be configured for using an NFS for object storage.
- HTTPS secured by a trusted certificate for the KeySafe 5 endpoints. The deploy script will enable HTTPS connections with a self-signed insecure certificate.
- Require authentication to access KeySafe 5. OIDC & OAUTH2 are currently supported in KeySafe 5. The deploy script will not set up

> authenticated access.
> - Time synchronization between the central platform and all agents.

The script requires a local installation of Docker or Podman. When using `podman` on Red Hat Enterprise Linux, you should install the `podman-docker` package to provide the Docker alias.

The user executing the deploy script must be able to successfully execute `docker info`. If this is not the case, please consult the appropriate documentation for your platform, Docker Documentation or Podman Documentation.

This release includes Docker images that need to be pushed to a Docker registry. If you have a private registry you may push the images from a different machine.

1.2. Hardware Requirements

See Hardware Requirements.

1.3. Unpack the release

```
mkdir keysafe5-install
tar -xf nshield-keysafe5-1.4.0.tar.gz -C keysafe5-install
cd keysafe5-install
```

The user executing the `deploy.sh` script must have permission to read and write files within the `keysafe5-install` directory. This will automatically be the case if the user extracting the release package is the same user that executes the deploy script.

1.4. Existing infrastructure

This section describes the interaction with infrastructure that you may like to use when installing KeySafe 5 via the deployment script. Otherwise skip to Authentication.

1.4.1. Docker images

If you have a private registry you may push the images to it like so:

```
# Load the Docker images to your local Docker
docker load < docker-images/codesafe-mgmt.tar
docker load < docker-images/hsm-mgmt.tar
docker load < docker-images/sw-mgmt.tar
docker load < docker-images/ui.tar
```

```
# We need to use a private registry in many places
export DOCKER_REGISTRY=private.registry.local/my_space/keysafe5

# Tag the Docker images for a private registry
docker tag codesafe-mgmt:1.4.0 $DOCKER_REGISTRY/codesafe-mgmt:1.4.0
docker tag hsm-mgmt:1.4.0 $DOCKER_REGISTRY/hsm-mgmt:1.4.0
docker tag sw-mgmt:1.4.0 $DOCKER_REGISTRY/sw-mgmt:1.4.0
docker tag mgmt-ui:1.4.0 $DOCKER_REGISTRY/mgmt-ui:1.4.0

# Log in to ensure pushes succeed
docker login private.registry.local

# And push
docker push $DOCKER_REGISTRY/codesafe-mgmt:1.4.0
docker push $DOCKER_REGISTRY/hsm-mgmt:1.4.0
docker push $DOCKER_REGISTRY/sw-mgmt:1.4.0
docker push $DOCKER_REGISTRY/mgmt-ui:1.4.0
```

By setting the `DOCKER_REGISTRY` environment variable the deploy script will pull images from that registry. Otherwise the deploy script will set up a local insecure Docker registry. In this case, ensure that Docker is installed.

1.4.2. Kubernetes

If you have a Kubernetes cluster available, ensure that `kubectl` points to it, and that `kubectl get pods -A` returns a list of pods. Otherwise the deploy script will install K3s locally to `/usr/local/bin` and create a `${HOME}/.kube/config` to point to it. Kubernetes tools use the `KUBECONFIG` environment variable for the location of its configuration file, but when unset this defaults to `${HOME}/.kube/config`. For this setting to persist it needs to be added to your shell's configuration file.

1.4.3. Istio

If you have Istio installed, ensure that `istioctl` is on your path. Otherwise the deploy script will download a local copy of `istioctl`, and install Istio as required.

1.5. RabbitMQ

Entrust recommends that you use your standard secure RabbitMQ installation, along with your policies for authentication and virtual hosts on your production system; this is only a demo system.

If you have an existing RabbitMQ environment:

- Set the environment variable `RABBIT_URL` to point to the RabbitMQ server, port, and vhost like: `rabbitmq.example.com:5671/keysafe5`.

- You will also need to create a Kubernetes generic secret in the `nshieldkeysafe5` namespace with `ca.crt`, `tls.crt`, and `tls.key` for a user that has full access to the vhost. Set `RABBIT_SECRETS` to the name of this generic secret.

If you do not have an existing RabbitMQ server, the deploy script will set one up on the Kubernetes cluster along with the secrets for both the server and agents. By default, neither RabbitMQ's management interface nor peer discovery interface for Kubernetes will be enabled. Should either of these be required, set the environment variable `RABBITMQ_PLUGINS`. To enable both, set it to `"rabbitmq_management rabbitmq_peer_discovery_k8s"`. To enable only one, set it to the appropriate value. For example, to enable just the management interface, set it to `"rabbitmq_management"`.

1.5.1. MongoDB

Entrust recommends that you use your standard secure MongoDB Replica Set installation.

If you have an existing MongoDB deployment:

- Set the environment variable `MONGODB` to a backslash-comma separated list of servers, along with their port numbers in the form: `mongo-1.example.com:27017\,mongo-2.example.com:27017` The backslash should be visible when running `echo $MONGODB`. A quick tip: using single-quotes `'` will prevent the bash command line acting on the backslash you have typed.
- You will also need to create a Kubernetes generic secret in the `nshieldkeysafe5` namespace with `ca.crt`, `tls.crt`, and `tls.key` for a user that has readWrite roles on the databases: `codesafe-mgmt-db`, `hsm-mgmt-db` and `sw-mgmt-db`. Set `MONGO_SECRETS` to the name of this generic secret.

If you do not have an existing MongoDB deployment, the deploy script will set one up on the Kubernetes cluster along with the secrets for both the server and backend services.

MongoDB 5.0 and newer requires use of the AVX instruction set for processors. For more information, see MongoDB Production Notes

1.5.2. Object Storage

If you do not have an existing Kubernetes cluster, or if your cluster contains only 1 worker node, the deploy script will use local storage on the single worker node.

If you would like to use an NFS for large object storage, set the environment variable `NFS_IP` to the NFS server address, and `NFS_PATH` to the path of the directory being exported from the NFS server.

To set the user and group IDs used by the KeySafe 5 application when accessing the object storage, configure the `podSecurityContext.runAsUser`, `podSecurityContext.runAsGroup` and `podSecurityContext.fsGroup` Chart parameters. To do this, specify the environment variable `KEYSAFE_BACKEND_CHART_EXTRA_ARGS`. For example, `KEYSAFE_BACKEND_CHART_EXTRA_ARGS="--set podSecurityContext.runAsUser=2000 --set podSecurityContext.runAsGroup=3000"`.

1.6. Authentication

To disable OIDC authentication, set the environment variable `DISABLE_AUTHENTICATION` to `yes`, and you may move on to Install KeySafe 5.

To configure authentication for Istio, the environment variable `AUTH_ISSUER_URL` needs to point at the issuer URL. Additionally, either `AUTH_JWKS` (for the payload) or `AUTH_JWKS_URL` (for the URL) also needs to be set. `AUTH_AUDIENCES` should be a comma-delimited list. The deploy script will automatically add the fully qualified domain name for the host to this list if not already present.

For UI authentication the deploy script requires an `OIDCProviders.json` file. Its location should be set in the environment variable `OIDC_PROVIDERS_FILE_LOCATION`.

Further details on configuring authentication for KeySafe 5 can be found in the Helm Chart Installation section of the *KeySafe 5 Installation Guide*.

1.7. Install KeySafe 5

It is now possible to run the deploy script.

Do not run the deploy script under `sudo`. If `sudo` permissions are required, `sudo` will be called by the script and you will be prompted for your credentials.

The deploy script must be run from inside the directory to which it is extracted. Running with the `-n` flag will perform a set of pre-flight checks and show what will happen then exit without taking any action.

```
./deploy.sh -n
```

To disable authentication set the environment variable `DISABLE_AUTHENTICATION` to `yes`. Otherwise you may follow the instructions in the Authentication section.

You may now perform the deployment with the `-y` flag.

```
./deploy.sh -y
```

The script will take a few minutes to run, showing what actions are taking place. You may be prompted for your password by `sudo`, for example when installing K3s.

The script will create a local insecure Certificate Authority to be used by the `agentcert.sh` and `updateinternalcerts.sh` scripts. This directory should be preserved to allow this.

The script will also produce two archives, `agent-config.tar.gz` (for Unix) and `agent-config.zip` (for Windows), that contains the agent configuration file and CA certificates for TLS authentication to RabbitMQ. The contents are used for configuring nShield client machines below.

1.7.1. K3s

If the `deploy.sh` script has installed K3s, you should configure `kubectl` access for the current user account by running:

```
mkdir -p $\{HOME}/.kube
sudo /usr/local/bin/k3s kubectl config view --raw > $\{HOME}/.kube/config
chmod 600 $\{HOME}/.kube/config
export KUBECONFIG=$\{HOME}/.kube/config
```

This step is necessary for any user account on this machine that will be required to administrate the KeySafe 5 central platform.

You may append the `export KUBECONFIG=${HOME}/.kube/config` to your shell's configuration file.

1.8. Configure nShield client machines

Ensure no firewall rules are blocking the AMQP port communication between the machine exposing the AMQP port from Kubernetes and the machine running the agent.

In summary, to configure your nShield client machine to be managed and monitored by this deployment:

1. Install the KeySafe 5 agent on the nShield client machine containing the relevant Security World or HSMs.

2. Extract the `agent-config.tar.gz` or `agent-config.zip` archive on the nShield client machine alongside the KeySafe 5 agent.
3. Generate a unique private key and client CSR using the provided `ampqtls` for each individual KeySafe 5 agent.
4. Copy the client CSR to the central platform and sign it with the CA.
5. Copy the certificate to the nShield client machine, and place it alongside the KeySafe 5 agent configuration.

The steps to install and configure vary depending on the client.

The deploy script will output a tar archive called `agent-config.tar.gz`, and a zip archive called `agent-config.zip` that contains the agent configuration file and CA certificate for TLS authentication to RabbitMQ. A tool is provided for generating a unique private key and client CSR for each individual KeySafe 5 agent. Please see below.

1.8.1. Install on Linux

1. On a machine with a supported Security World installation installed, run:

```
sudo tar -xf keysafe5-install/keysafe5-agent/keysafe5-1.4.0-Linux-keysafe5-agent.tar.gz -C
/opt/nfast/keysafe5/
sudo tar -xf agent-config.tar.gz -C /opt/nfast/keysafe5/
```

2. Generate a TLS private key and CSR for this KeySafe 5 agent.

```
/opt/nfast/keysafe5/bin/ks5agenttls -keypath=/opt/nfast/keysafe5/conf/messagebus/tls/tls.key -keygen
/opt/nfast/keysafe5/bin/ks5agenttls -keypath=/opt/nfast/keysafe5/conf/messagebus/tls/tls.key -csrgen
```

3. Transfer the resulting CSR to the central platform.
4. Follow the instructions in Signing the Agent CSR to sign the CSR, then transfer the resulting certificate and the `cacert.pem` to this computer.
5. Place the files (renaming the certificate to `tls.crt` and `cacert.pem` to `ca.crt`) into the tls directory alongside the original key.

```
sudo cp ks5agent_demohost.crt /opt/nfast/keysafe5/conf/messagebus/tls/tls.crt
sudo cp cacert.pem /opt/nfast/keysafe5/conf/messagebus/tls/ca.crt
```

6. If the hardserver is already running, use the KeySafe 5 install script to start the KeySafe 5 Agent without restarting the hardserver:

```
sudo /opt/nfast/keysafe5/sbin/install
```

Otherwise use the nShield install script which will start all the services:

```
sudo /opt/nfast/sbin/install
```

1.8.2. Install on Windows

1. Run the `Keysafe5-agent.msi` installer if the KeySafe 5 agent is not already installed.

   This creates the nShield KeySafe 5 agent service but does not start it.

2. Populate the KeySafe 5 agent configuration using the files from the generated `agent-config.zip`
3. Generate a TLS private key and CSR for this KeySafe 5 agent.

```
%NFAST_HOME%\bin\amqptls.exe -keypath=%NFAST_DATA_HOME%\keysafe5\conf\amqp\tls\tls.key -keygen
%NFAST_HOME%\bin\amqptls.exe -keypath=%NFAST_DATA_HOME%\keysafe5\conf\amqp\tls\tls.key -csrgen
```

4. Transfer the resulting CSR to the central platform.
5. Follow the instructions in Signing the Agent CSR to sign the CSR, then transfer the resulting certificate and the `cacert.pem` to this computer.
6. Place the files (renaming the certificate to `tls.crt` and `cacert.pem` to `ca.crt`) into the tls directory alongside the original key.

```
%NFAST_DATA_HOME%\keysafe5\conf\amqp\tls\tls.crt
%NFAST_DATA_HOME%\keysafe5\conf\amqp\tls\ca.crt
```

7. Start the nShield KeySafe 5 agent service using the Windows Service Manager.

1.8.3. Signing the Agent CSR

1. Use the `agentcert.sh` script to create a client TLS certificate for this KeySafe 5 agent using the CSR from the Agent, and the CA created by the deploy script.

2. Using the username printed in the output of the previous command, configure the RabbitMQ server to allow access for this X.509 user in the appropriate virtual host.

```
export x509user=ks5agent_demohost
export RUN_RABBIT="kubectl -n rabbitns exec rabbit-chart-rabbitmq-0 -c rabbitmq -- "
${RUN_RABBIT} rabbitmqctl add_user $x509user "ephemeralpw"
${RUN_RABBIT} rabbitmqctl set_permissions -p $RABBIT_VHOST $x509user ".*" ".*" ".*"
${RUN_RABBIT} rabbitmqctl clear_password $x509user
```

3. Copy the client TLS certificate output from the `agentcert.sh` command to the nShield host machine with the KeySafe 5 agent.
4. Also copy over the CA certificate `internalCA/cacert.pem`

1.9. Configure an nShield Connect

A KeySafe 5 agent is installed on the nShield Connect for nShield Connect images released with Security World v13.4 and later software. This agent allows an nShield Connect to be monitored and managed without, or in addition to, the Connect being enrolled to a nShield host machine (a machine with nShield Security World software installed) which also has a KeySafe 5 agent installed.

Output of certificate/CSR data is truncated in these examples.

To configure an nShield Connect to be managed and monitored by your KeySafe 5 central platform:

1. On the nShield Connect Serial Console, generate a certificate signing request (CSR) for the KeySafe 5 agent using the `ks5agent amqpcsr` command.

```
(cli)ks5agent amqpcsr
-----BEGIN CERTIFICATE REQUEST-----
MII...
-----END CERTIFICATE REQUEST-----
```

2. On the machine where the KeySafe 5 central platform was installed, copy and paste the generated CSR to a new file in the directory where the `deploy.sh` script was executed.
3. Generate a TLS certificate for this agent using `agentcert.sh` script, specifying how many days the generated certificate should be valid for.

```
$ ./agentcert.sh connect_AAAA_AAAA_AAAA.csr 365
Certificate generated to connect_AAAA_AAAA_AAAA.crt. Valid for 365 days
CA Certificate available at /path/to/keysafe5-install/internalCA/cacert.pem

RabbitMQ will need to be configured to allow access for the user 'ks5agent_nshield_module_AAAA_AAAA_AAAA':
   export RUN_RABBIT="kubectl -n rabbitns exec rabbit-chart-rabbitmq-0 -c rabbitmq -- "
   ${RUN_RABBIT} rabbitmqctl add_user ks5agent_nshield_module_AAAA_AAAA_AAAA ephemeralpw
   ${RUN_RABBIT} rabbitmqctl set_permissions -p nshieldvhost ks5agent_nshield_module_AAAA_AAAA_AAAA '.*'
'.*' '.*'
   ${RUN_RABBIT} rabbitmqctl clear_password ks5agent_nshield_module_AAAA_AAAA_AAAA
```

4. On the machine where the KeySafe 5 central platform was installed, configure the deployed RabbitMQ server to allow access for this agent.

The output of the `agentcert.sh` script contains the exact

commands to run for the specific agent username. The username has the format `ks5agent_nshield_module_<connect_esn>`. In the provided examples, the ESN of the Connect is AAAA-AAAA-AAAA.

```
export RUN_RABBIT="kubectl -n rabbitns exec rabbit-chart-rabbitmq-0 -c rabbitmq -- "
${RUN_RABBIT} rabbitmqctl add_user ks5agent_nshield_module_AAAA_AAAA_AAAA ephemeralpw
${RUN_RABBIT} rabbitmqctl set_permissions -p nshieldvhost ks5agent_nshield_module_AAAA_AAAA_AAAA '.*' '.*'
'.*'
${RUN_RABBIT} rabbitmqctl clear_password ks5agent_nshield_module_AAAA_AAAA_AAAA
```

5. Install the agent TLS certificate on the Connect.
   a. On the machine where the TLS certificate was generated, `base64` encode the certificate.

   ```
   $ base64 --wrap=0 connect_AAAA_AAAA_AAAA.crt
   Q2VydGlmaW...
   ```

   b. On the nShield Connect Serial Console, load the TLS certificate for the agent using the `ks5agent amqptls` command.

   ```
   (cli) ks5agent amqptls tls.crt <base64_encoded_crt>
   Saved tls.crt
   ```

6. Install the CA certificate on the Connect.
   a. On the machine where the TLS certificate was generated, `base64` encode the certificate of the Certificate Authority. After running the `deploy.sh` script this certificate can be found at `internalCA/cacert.pem`.

   ```
   $ base64 --wrap=0 internalCA/cacert.pem
   LS0t...
   ```

   b. On the nShield Connect Serial Console, load the CA certificate using the `ks5agent amqptls` command.

   ```
   (cli) ks5agent amqptls ca.crt <base64_encoded_crt>
   Saved ca.crt
   ```

7. On the nShield Connect Serial Console, configure the agent AMQP URL to point to the deployed RabbitMQ server. The correct URL can be found in the generated `agent-config.tar.gz` on the machine where the KeySafe 5 central platform was installed.

The `deploy.sh` script creates a dedicated virtual host called `nshieldvhost` in the deployed RabbitMQ server, for KeySafe 5 to

> use.

```
(cli) ks5agent cfg amqp.url=<central_platform_ip>:5671/nshieldvhost
```

8. On the nShield Connect Serial Console, enable the agent using the `ks5agent enable` command.

```
(cli) ks5agent enable
```

1.10. Access KeySafe 5

You can now access KeySafe 5 through the URL provided by the deploy script. For example, you could send curl requests:

```
curl -s --cacert internalCA/cacert.pem https://$(hostname -f)/mgmt/v1/hsms | jq
curl -s --cacert internalCA/cacert.pem https://$(hostname -f)/mgmt/v1/pools | jq
```

You can access the Management UI in a web browser at `https://$(hostname -f)`.

1.11. Uninstall

If Kubernetes was not provided and K3s was installed by the deploy script, you may simply uninstall K3s which will clear up all the installed helm charts.

```
/usr/local/bin/k3s-uninstall.sh
```

This will request `sudo` permissions.

If a private Docker Registry was not provided, the deploy script will have created a local one and it will be removed when the script finishes. Should this fail, you may uninstall it manually by running:

```
docker stop registry
docker rm registry
```

If a Kubernetes installation was provided then the helm charts will need to be uninstalled individually.

```
helm --namespace nshieldkeysafe5 uninstall keysafe5-istio
helm --namespace nshieldkeysafe5 uninstall keysafe5-backend
helm --namespace nshieldkeysafe5 uninstall keysafe5-ui
```

If an existing RabbitMQ installation was not provided, then the deploy script will have installed a RabbitMQ helm chart that should be uninstalled. The same is also true for MongoDB.

```
helm --namespace rabbitns uninstall rabbit-chart
helm --namespace mongons uninstall mongo-chart
```

If Istio was installed by the deploy script, it may be uninstalled by running:

```
keysafe5-install/istioctl uninstall --purge
```

To uninstall the KeySafe 5 agent, run the KeySafe 5 uninstaller:

```
sudo /opt/nfast/keysafe5/sbin/install -u
```

2. Security Guidance

Your nShield HSM protects the confidentiality and integrity of your Security World keys. KeySafe 5 allows an authorized client to remotely configure and manage an estate of nShield HSMs. All network traffic between KeySafe 5 and clients using the UI, the REST API, or both, passes through a secure channel. This TLS based secure channel is set up using token-based client authentication. The administrator of the KeySafe 5 system must remain diligent concerning the entities who are given access to the system and the secure configuration of the system.

Entrust recommends the following security-related actions for KeySafe 5 deployments:

- Ensure that log levels are set appropriately for your environment.

  More verbose log levels might expose information that is useful for auditing users of KeySafe 5, but the log information also reveals which REST API operations were performed. While this log information might be useful for diagnostics, it could also be considered sensitive and should be suitably protected when stored.

- Rotate the logs regularly. The log files could grow quickly if left unattended for a long time. The system administrator is responsible for log rotation.
- Verify the integrity of the KeySafe 5 tar file before executing it. You can verify the integrity of this file with the hash provided with the software download.
- Suitably protect the network environment of KeySafe 5 to maintain its availability, for example using firewalls and intrusion detection and prevention systems.
- Ensure that the KeySafe 5 platform's system clock is set accurately and only authorized system administrators can modify it so that the platform correctly interprets certificate and token lifetimes.
- Ensure that only authorized system administrators have access to the KeySafe 5 system, and only trusted software is run on the platform hosting KeySafe 5.
- Take standard virus prevention and detection measures on the platform hosting KeySafe 5.
- The system administrator should consider whether threats in the KeySafe 5 deployment environment would justify the encryption of the sensitive configuration data held in Kubernetes secrets, see Kubernetes documentation.

3. Hardening The Deployment

To harden the demo deployment there are a number of steps to follow. The documentation below requires modifying the configuration of the Helm charts installed by following the Manual Install steps or running the `deploy.sh` script. To obtain the installed configuration for each installed Helm chart, run the following commands:

```
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-backend > keysafe5-backend-values.yaml
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-ui > keysafe5-ui-values.yaml
helm -n nshieldkeysafe5 get values --all --output yaml keysafe5-istio > keysafe5-istio-values.yaml
```

Documentation for each configurable value in the KeySafe 5 Helm charts can be found by untarring the chart.tgz and viewing the contents of either README.md or the default values.yaml file.

3.1. Certificates

The Manual Install steps and the `deploy.sh` script will generate a local Certificate Authority, and install a number of short-lived demo certificates. These certificates must be replaced to continue using the system after their expiry.

Your new certificates will need to adhere to X.509 v3, sometimes known as a multiple-domain certificates, or SAN certificates. The X.509 extension Subject Alternative Name (SAN) allows specifying multiple hostnames, and has replaced Common Name as the source of the hostname.

3.1.1. External KeySafe 5 Server TLS Certificate

To update the TLS certificate used for the KeySafe 5 Server (for HTTPS connections to the REST API or User Interface) you must create a Kubernetes Secret containing the new certificate/private key and redeploy the `keysafe5-istio` Helm chart.

For more information on enabling HTTPS for helm-keysafe5-istio, see the Helm Chart Installation section of the Installation Guide.

The Manual Install steps and `deploy.sh` script create a Kubernetes Secret called `keysafe5-server-credential`. You can either delete the existing Secret as shown below, or use a different name for the Secret containing your new TLS certificate.

```
kubectl --namespace istio-system delete secret keysafe5-server-credential

kubectl --namespace istio-system create secret tls keysafe5-server-credential \
  --cert=path/to/cert/file \
```

```
  --key=path/to/key/file
```

Before running `helm upgrade`, set the following values in your `keysafe5-istio-values.yaml`:

- httpsEnabled=true
- tls.existingSecret=keysafe5-server-credential (or the name you used when creating the Kubernetes Secret containing your certificate/private key)

```
helm upgrade --install keysafe5-istio \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-istio-values.yaml \
  --wait --timeout 1m \
helm-charts/nshield-keysafe5-istio-1.4.0.tgz
```

3.1.2. Internal Certificates

The Manual Install steps and `deploy.sh` script create a Certificate Authority, and install KeySafe 5 using TLS authorised by the CA for the communications between the central platform and MonogDB/RabbitMQ.

You can refresh these internal certificates by running the `updateinternalcerts.sh` script, specifying the number of days for which the new certificates will be valid.

Alternatively if you have an external CA, you may generate keys and certificates and pass the directory containing them to `updateinternalcerts.sh` script.

The help for `updateinternalcerts.sh` is shown with the `-h` option:

```
  Usage: ./updateinternalcerts.sh [OPTION]... [SERVER] DAYS
         ./updateinternalcerts.sh [OPTION]... [SERVER] DIRECTORY

  Update the internal server and client TLS credentials for a deployment created
  by deploy.sh.

  This can update certificates and keys for both RabbitMQ and MongoDB as set up
  by deploy.sh, along with the certificates and keys for the KeySafe 5 Backend
  services.  RabbitMQ has a single key and certificate.  MongoDB has three keys
  and certificates, two for the replica-set, and one for the arbiter.  The
  KeySafe 5 backend services also have two keys and client certificates for
  connecting to both RabbitMQ and MongoDB.

  When using the insecure internal CA, specify when the new keys and certs will
  expire and this script will do the rest.

  When using a secure external CA, all the keys and certificates will have to
  be provided and they will be packaged up for the deployment.

  When updating the certificates for RabbitMQ the script will generate an
  updated agent-config.tar.gz and agent-config.zip files containing a config
  file and ca.crt.

  Certificate Addresses
```

```
The certificates contain names and IP addresses that verify the connection,
signed by a mutally trusted authority.

The backend servers only use Kubernetes internal addressing to access MongoDB
and RabbitMQ.

MongoDB is only used by the backend services so does not require an external
address.

RabbitMQ is also accessed by the KeySafe 5 Agents through an external address.

Shared Optional parameters

-n, --namespace=NAMESPACE   The deploy script normally uses nshieldkeysafe5
                            as the namespace for the KeySafe 5 servers,
                            mongons for MongoDB, and rabbitns for RabbitMQ.
                            However you can override these to use a single
                            namespace for all the servers.  If that is the
                            case, then use this option.
SERVER                      One of "mongodb" or "rabbitmq".
                            If the server is not specified then both will
                            be updated.
Internal CA
-----------

DAYS                        Number of days before the generated certificates
                            expire

This script will use the IP address provided by the loadbalancer, but you may
override this with a different IP address or DNS name.

-r, --rabbitmq=ADDRESS      The external address for RabbitMQ
-m, --mongodb=ADDRESS       The external address for MongoDB

External CA
-----------

DIRECTORY                   The directory containing the server and client
                            keys and certificates in PEM format.
Files in the directory
----------------------
ca.crt                      The certificate of the CA that is to be trusted by
                            the system.
rabbitmq.key                The key to be used by rabbit-chart-rabbitmq
rabbitmq.crt                And its certificate
ks5rabbitmq.key             The key to be used by ks5
ks5rabbitmq.crt             And its certificate
mongodb-0.key               The key to be used by mongo-chart-mongodb-0
mongodb-0.crt               And its certificate
mongodb-1.key               The key to be used by mongo-chart-mongodb-1
mongodb-1.crt               And its certificate
mongodb-a.key               The key to be used by mongo-chart-mongodb-arbiter
mongodb-a.crt               And its certificate
ks5mongodb.key              The key to be used by ks5-mongo-user
ks5mongodb.crt              And its certificate

If rabbitmq is specified, the mongodb files may be skipped, and vice versa.

Internal CA examples

Refresh certificates for both servers for the next 30 days, and set a DNS
entry for RabbitMQ
  ./updateinternalcerts.sh -r rabbitmq.example.com 30

Refresh just MongoDB for the next 4 weeks
  ./updateinternalcerts.sh mongodb 28
```

```
External CA examples

Refresh certificates for both servers (and KeySafe 5) in a specific namespace
  ./updateinternalcerts.sh -n kansas all-certs-dir

  Refresh certificates for just MongoDB (and KeySafe 5) in standard namespaces
  ./updateinternalcerts.sh mongodb mongo-certs-dir
```

This script will update both MongoDB and RabbitMQ TLS certificates, though you may choose to update only one set of TLS certificates by specifying only that server.

The `updateinternalcerts.sh` script must be run from a directory containing the KeySafe 5 Helm charts (for example, from the root directory of the untarred KeySafe 5 package). If the CA, created when the original deploy script was run, is not available then a new one will be created and used.

If both certificates have expired when running `updateinternalcerts.sh`, updating the certificates of only one service may return an error. In this case re-running `updateinternalcerts.sh` without specifying any server will update both server certificates at once, and will usually solve the problem.

If an error occurs during certificate update you can restore the previous setup by rolling back Helm chart installations to a previous release, see Helm Chart Upgrade in the Upgrade section of the Installation Guide.

3.1.2.1. MongoDB TLS Certificates

The Manual Install steps and `deploy.sh` script installs the Bitnami MongoDB Helm chart with TLS enabled for the connections between KeySafe 5 and the MongoDB server and also with TLS used for authentication. These certificates will initially be valid for 30 days from the time the process was run.

You can refresh the MongoDB certificates by running the `updateinternalcerts.sh` script, specifying the number of days the new certificates will be valid for.

```
updateinternalcerts.sh mongodb 365
```

This script will:

- Generate the new TLS certificates
- Update the MongoDB helm chart to use the new certificates

- Update the keysafe5-backend helm chart to use the new certificates

You may specify an external address with the -m parameter but this is usually not required.

For an external CA, the following command will apply the keys and certificates from the directory mycerts

```
updateinternalcerts.sh mongodb mycerts
```

If using an external CA, the following suggestions should improve the chances of success: * Create keys for all the MongoDB servers (mongo-chart-mongodb-arbiter-0, mongo-chart-mongodb-0, and mongo-chart-mongodb-1) * Use 4k RSA keys * Set the Key Usage to Digital Signature, Non Repudiation, Key Encipherment * For the server certificates, do not set the Extended Key Usage (the keys are used as both server and client keys) * For the server certificates' Subject Alternative Name, add DNS records for mongodb, mongo-chart-mongodb.mongons.svc.cluster.local, <server>.mongo-chart-mongodb-headless.mongons.svc.cluster.local, and mongo-chart.mongons.svc.cluster.local * For the client certificate, set the Extended Key Usage as "TLS Web Client Authentication" * For the client certificate's Subject Alternative Name, set the first entry to the DNS value of ks5-mongo-user, and add DNS entries for keysafe5-backend-ks5-mongo-user-headless, keysafe5-backend-ks5-mongo-user.nshieldkeysafe5.svc.cluster.local, and keysafe5-backend.nshieldkeysafe5.svc.cluster.local.

3.1.2.2. RabbitMQ

The Manual Install steps and deploy.sh script installs the Bitnami RabbitMQ Helm chart with TLS enabled.

You can refresh the RabbitMQ certificates by running the updateinternalcerts.sh script, specifying the number of days the new certificates will be valid for.

```
updateinternalcerts.sh rabbitmq 365
```

This script will:

- Generate the new TLS certificates
- Update the RabbitMQ helm chart to use the new certificates
- Update the keysafe5-backend helm chart to use the new certificates
- Output a new agent-config.tar.gz that contains the agent configuration file and TLS certificates for authentication to RabbitMQ

The external address that will be put into the certificate, and the configuration file, will be

the IP address of the Ingress service provided by the Kubernetes server. If you wish to use a different IP address or DNS name, this can be passed in with the `-r` parameter.

For an external CA, the following command will apply the keys and certificates from the directory `mycerts`

```
updateinternalcerts rabbitmq mycerts
```

If using an external CA, the following suggestions should improve the chances of success: * Use 4k RSA keys * Set the Key Usage to `Digital Signature, Non Repudiation, Key Encipherment` * For the server certificate, set the Extended Key Usage to `TLS Web Server Authentication` * Ensure the server certificate's Subject Alternative Name includes the IP address and DNS names that will be used to connect to RabbitMQ from external clients, along with the DNS entries `rabbitmq`, `rabbit-chart-rabbitmq-headless`, `rabbit-chart-rabbitmq.rabbitns.svc.cluster.local`, **`.rabbit-chart-rabbitmq-headless.rabbitns.svc.cluster.local`**, and `rabbit-chart.rabbitns.svc.cluster.local.` * For the client certificate, set the Extended Key Usage to `TLS Web Client Authentication.` * Ensure the client certificate's Subject Alternative Name starts with the DNS value of `ks5` (the KeySafe 5 Backend's RabbitMQ username), followed by `keysafe5-backend-ks5-headless`, `keysafe5-backend-ks5.nshieldkeysafe5.svc.cluster.local`,`.keysafe5-backend-ks5-headless.nshieldkeysafe5.svc.cluster.local`, and `keysafe5-backend.nshieldkeysafe5.svc.cluster.local`.

You will need to update your client machines with a KeySafe 5 agent installed to use the updated config and restart the agent so that the new configuration is applied.

```
sudo tar xf agent-config.tar.gz -C /opt/nfast/keysafe5/
/opt/nfast/scripts/init.d/keysafe5-agent restart
```

3.2. Authentication

If you chose to install the demo deployment without authentication you should enable authentication for accessing the KeySafe 5 REST API and User Interface.

For how to configure authentication for the KeySafe 5 REST APIs see `Helm Chart Installation: helm-keysafe5-istio authentication` in the Installation Guide.

To update the `keysafe5-istio` Helm chart installed by the demo deployment, set the following values in `keysafe5-istio-values.yaml`.

- requireAuthn=true

- issuer[0].authIssuer="https://foobar.auth0.com"
- issuer[0].authJWKsURI="https://www.googleapis.com/oauth2/v1/certs"
- issuer[0].authAudiences[0]="https://keysafe5.location"

Then run `helm upgrade`.

```
helm upgrade --install keysafe5-istio \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-istio-values.yaml \
  --wait --timeout 1m \
helm-charts/nshield-keysafe5-istio-1.4.0.tgz
```

To update the `keysafe5-ui` Helm chart installed by the demo deployment, set the following values in `keysafe5-ui-values.yaml`:

- authMethod=oidc

Untar the chart and copy your OIDC provider config file (`OIDCProviders.json`) into the config directory:

For more details on how to populate `OIDCProviders.json` and how to configure authentication for the KeySafe 5 User Interface see `Helm Chart Installation: Configure UI authentication` in the Installation Guide.

```
tar -xf helm-charts/nshield-keysafe5-ui-1.4.0.tgz -C helm-charts
cp my-oidc-provider-config.json helm-charts/nshield-keysafe5-ui/config/OIDCProviders.json
```

Then run `helm upgrade`:

```
helm upgrade --install keysafe5-ui \
  --namespace=nshieldkeysafe5 \
  --values keysafe5-ui-values.yaml \
  --wait --timeout 3m \
helm-charts/nshield-keysafe5-ui
```

3.3. K3s

If not using your own Kubernetes cluster, the `deploy.sh` script will create one using K3s. To harden this K3s install follow the official documentation at K3s Hardening Guide.

The `deploy.sh` script installs K3s with `traefik` and `metrics-server` explicitly disabled.