



EverTrust Stream documentation v2.0

Installation Guide

EVERTRUST

Table of Contents

1. Introduction	1
1.1. Description	1
1.2. Prerequisites	1
2. Installing on CentOS/RHEL	2
2.1. Pre-requisites	2
2.2. Installation	2
2.3. Configuration	6
2.4. Security Guidelines	45
2.5. Upgrade	50
2.6. Uninstallation	51
3. Installing on Kubernetes	54
3.1. Installation	54
3.2. First login	56
3.3. Production checklist	58
3.4. Upgrade	60
3.5. Uninstallation	61
4. Running with Docker/Compose	62
4.1. Docker Compose example	62
4.2. Vanilla Docker example	63
4.3. Injecting extra configuration	64
4.4. Custom startup scripts	64
5. Troubleshooting	65
5.1. Stream Doctor	65

1. Introduction

1.1. Description

Stream version 2.0 is EverTrust Certification Authority. This document is an installation procedure detailing how to install and bootstrap a Stream instance on your infrastructure for the version 2.0. It does not describe how to configure and operate the instance. Please refer to the administration guide for administration related tasks.

1.2. Prerequisites

1.2.1. Choose an installation method

We offer two installation modes:

- Enterprise Linux 7.x/8.x/9.x x64
- A cloud-native installation using Kubernetes

Depending on your needs, you'll have to choose the solution that fits your use cases the best. Reach out to our support team to get suggestions on how to deploy on your infrastructure.

1.2.2. Gathering your credentials

Both methods require that you download the binaries of the Stream software from our [software repository](#). The access to this repository is protected by username and password, which you should have got from our tech team. If you don't, you won't be able to continue with the installation. Email us to get your credentials, and come back to this step.

2. Installing on CentOS/RHEL

2.1. Pre-requisites

This section describes the system and software pre-requisites to install Stream.

2.1.1. System pre-requisites

The following elements are considered as system pre-requisites:

- A server running a **en-US** minimal install EL [7.x-8.x-9.x] x64 (RHEL / AlmaLinux / RockyLinux / Oracle Linux) with the network configured.
- Base EL [7.x-8.x-9.x] x64 repositories activated;
- An access with administrative privileges (root) to the server mentioned above as most commands are system-related and require super user privilege;

2.1.2. Software pre-requisites

All the following packages can be necessary to deploy Stream. Most are available on public repositories but some require specific configurations.

Package name	Mandatory	Online Instructions	Offline instructions	Additional information
stream-2.0.x-1.x86_64.rpm	☑	Online steps	Offline steps	
mongodb-mongosh, mongod-org-server-mongodb-org-tools, mongodb-database-tools`, mongodb-org-database-tools-extra	☑	Online steps	Offline steps	
nginx		Online steps	N/A	Recommended reverse proxy
stream-hardening-1.x86_64.rpm		Online steps	Offline steps	Configuration hardening rpm

2.2. Installation

2.2.1. Installing MongoDB

NOTE

Stream requires at least MongoDB version 4.4.2. For support reasons, EVERTRUST recommends to use the latest available version, which is MongoDB 6 at the time of writing.

Stream relies on MongoDB to store its data, whether it be configuration elements or certificate data. The necessary packages are **mongodb-org-server**, **mongodb-mongosh**, **mongodb-org-tools**, **mongodb-**

`database-tools` and `mongodb-org-database-tools-extra`. To install and configure MongoDB on a Redhat-based OS, follow these steps using an account with administrative privileges:

Installation with Internet Access

These steps are for when the server has internet access

1. Follow step 1 of the official MongoDB installation tutorial.
2. Run the following command to install the RPMs:

```
# yum install -y mongodb-org-server mongodb-mongosh mongodb-org-tools mongodb-org-database-tools-extra mongodb-database-tools
```

Installation without Internet Access

1. Download the .rpm files directly from the [MongoDB repository](#). Downloads are organized by Red Hat / CentOS version (e.g. 7 - do not select the Server folders), then MongoDB release version (e.g. 6.0), then architecture (e.g. x86_64). Upload the files to the server.
2. Run the following command to install the RPMs:

```
# yum localinstall mongodb-org-server-x.y.z.arch.rpm mongodb-mongosh-x.y.z.arch.rpm mongodb-org-tools-x.y.z.arch.rpm mongodb-org-database-tools-extra-x.y.z.arch.rpm mongodb-database-tools-x.y.z.arch.rpm
```

Common installation steps

3. Enable the service at startup with the following command:

```
# systemctl enable mongod
```

4. Start the `mongod` service with the following command:

```
# systemctl start mongod
```

5. Start the `mongosh` executable using the following command to check that the database is up and running:

```
# mongosh
```

For now, since we did not set up access control, everyone using `localhost` as DB URI can connect as administrator, which is something that needs to be prevented before setting-up Stream.

NOTE

The following section is not mandatory to get Stream up and running, but is highly recommended for security purposes.

6. In the mongo shell that was just opened, run the following commands:

```
> use admin;
> db.createUser(
  {
    user: "stream_db_admin",
    pwd: "AComplexPassword",
    roles: [ { role: "dbOwner", db: "stream" } ]
  }
)
```

This way, the created *stream_db_admin* user has owner permissions on the database named *stream*. You can change the *stream_db_admin* value to what you want to use as database username, the password to be what you want to use as a database password to match your password policies and the database name (the value to the *db* key) to what you want to use as the stream database. For the password, you can also `passwordPrompt()` (without quotes) as the password value, which will prompt you for a password upon pressing Enter. Be careful though as this is a password prompt without confirmation.

CAUTION

If you plan on using special characters in the password, be careful as the MongoDB engine has trouble with some of them. For more information on this topic, please refer to the [MongoDB documentation](#).

7. Edit the `/etc/mongod.conf` file and add the following section at the end:

```
security:
  authorization: enabled

setParameter:
  enableLocalhostAuthBypass: false
```

These options will prevent anonymous login to the MongoDB instance and will disable the localhost bypass.

8. Restart the MongoDB daemon to make the changes effective:

```
# systemctl restart mongod
```

9. When setting up Stream, use this connection string as the MongoDB URI :

```
mongodb://stream_db_admin:AComplexPassword@127.0.0.1:27017/stream?authSource=admin
```

If you used another username for the MongoDB user, replace the *stream_db_admin* part with the username that you used. Replace the *AComplexPassword* in the URI by the password that you chose when creating the account.

Replace */stream* in the URI by */databaseName* if you chose to use another name for your Stream database when creating the user.

2.2.2. Installing NGINX

CAUTION

In order to install Stream, the server must have access to a repository (mirror, iso file, ...) of the linux distribution you are using in order to be able to install the dependencies of the software.

1. Connect to the server with an account with administrative privileges;
2. Install the NGINX web server using the following command:

```
# yum install nginx
```

3. Enable NGINX to start at boot using the following command:

```
# systemctl enable nginx
```

4. Stop the NGINX service with the following command:

```
# systemctl stop nginx
```

2.2.3. Installing Stream

CAUTION

In order to install Stream, the server must have access to a repository (mirror, iso file, ...) of the linux distribution you are using in order to be able to install the dependencies of the software. Stream package has the following dependencies:

- `dialog`
- `java-17-openjdk-headless`
- `tzdata-java`

Please note that these packages may have their own dependencies.

Installation from the EverTrust repository

Create a `/etc/yum.repos.d/stream.repo` file containing the EverTrust repository info:

```
[stream]
```

```
enabled=1
name=Stream Repository
baseurl=https://repo.evertrust.io/repository/stream-rpm/
gpgcheck=0
username=<username>
password=<password>
```

Replace `<username>` and `<password>` with the credentials you were provided.

You can then run the following to install the latest Stream version:

```
# yum install stream
```

To prevent unattended upgrades when running yum update, you should pin the Stream version by adding

```
exclude=stream
```

at the end of the `/etc/yum.repos.d/stream.repo` file after installing Stream.

Installing from RPM

Download the latest RPM for version 2.0 on the [Official EVERTRUST repository](#).

Upload the file '***stream-2.0.x-1.x86_64.rpm***' to the server;

Access the server with an account with administrative privileges;

Install the Stream package with the following command:

```
# yum localinstall /root/stream-2.0.x-1.x86_64.rpm
```

2.3. Configuration

2.3.1. Initial Configuration

Introduction

This section assumes that Stream is running in a confined environment: nobody but the person performing the configuration operation and the key ceremony stakeholders should have access to Stream yet, and they should do so under the supervision of a security officer.

Selinux should be disabled during the initial configuration and bootstrapping operations. It will be re-enabled following the [security guidelines](#).


```
# setenforce Permissive
```

To ensure that it is permissive, run the following command

```
# getenforce
```

This should return **Permissive**

Configuring the firewall

NOTE

In order for Stream to work properly, the following ports are used:

- Exposed: 443 for HTTPS access to the product (through the web interface or through the API);
- Exposed: 80 for HTTP access to the product only to retrieve CRLs (the only allowed endpoint must be `/crls/*`, this is the case for the **default NGINX** configuration);
- Internal: 25520 and 8558 for high-availability configurations through the AKKA framework.
- Internal: 9000 for the Stream API.

Connect to the server with an account with administrative privileges;

Open port TCP/443 on the local firewall with the following command:

```
# firewall-cmd --permanent --add-service=https
```

Stream also needs HTTP traffic allowed since it is required to set up the CRLDPs :

```
# firewall-cmd --permanent --add-service=http
```

To make the change effective, you need to restart the firewall service:

```
# systemctl restart firewalld
```

Enable the service at startup with the following command:

```
# systemctl enable firewalld
```

Generating a Tink keyset

To protect its secrets, Stream relies on Tink. A Tink keyset can be issued as:

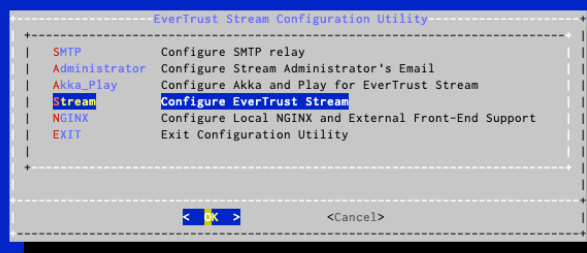
- A plaintext keyset (stored as a file, protected by the filesystem rights and SELinux);
- A GCP keyset (protected by a master key in a GCP KMS);
- An AWS keyset (protected by a master key in an AWS KMS).

Connect to the server with an account with administrative privileges;

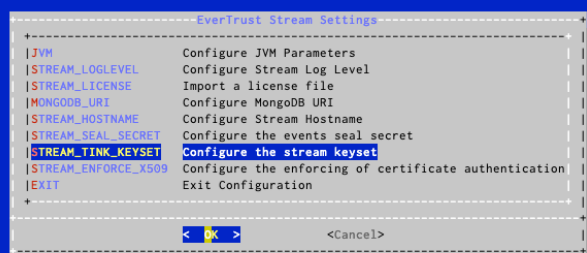
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

In the main menu, select '**Stream**':

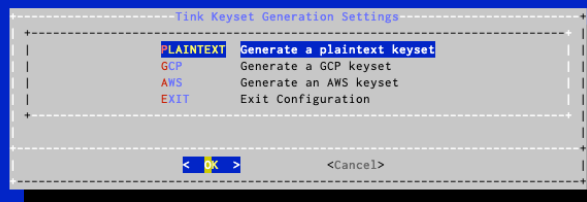


In the Stream menu, select '**STREAM_TINK_KEYSET**':



Generating a plaintext keyset

In the Tink Keyset Generation menu, select '**PLAINTEXT**':

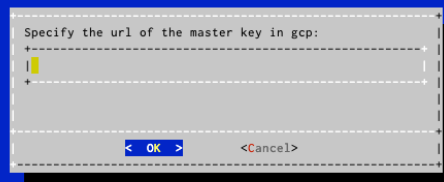


The keyset will be generated automatically. For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

Generating a GCP protected keyset

In the Tink Keyset Generation menu, select '**GCP**':



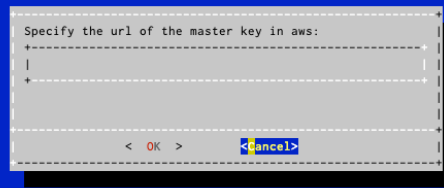
The URL of the GCP master key must be typed in the menu.

After pressing **OK**, the keyset will be generated automatically. For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

Generating an AWS protected keyset

In the Tink Keyset Generation menu, select '**GCP**':



The URL of the AWS master key must be typed in the menu.

After pressing **OK**, the keyset will be generated automatically. For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

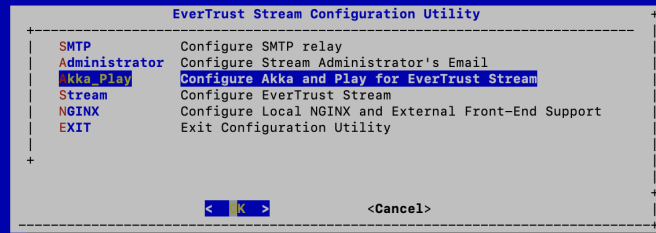
Generating a Play secret

Connect to the server with an account with administrative privileges;

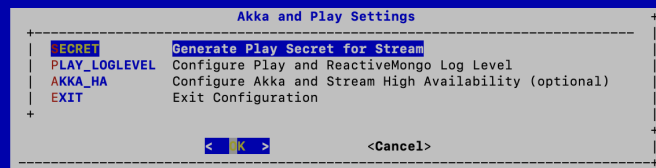
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

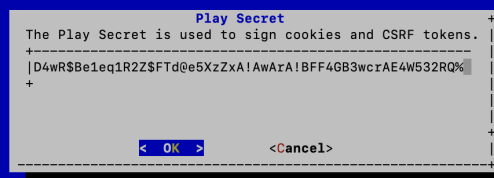
In the main menu, select '**Akka_Play**':



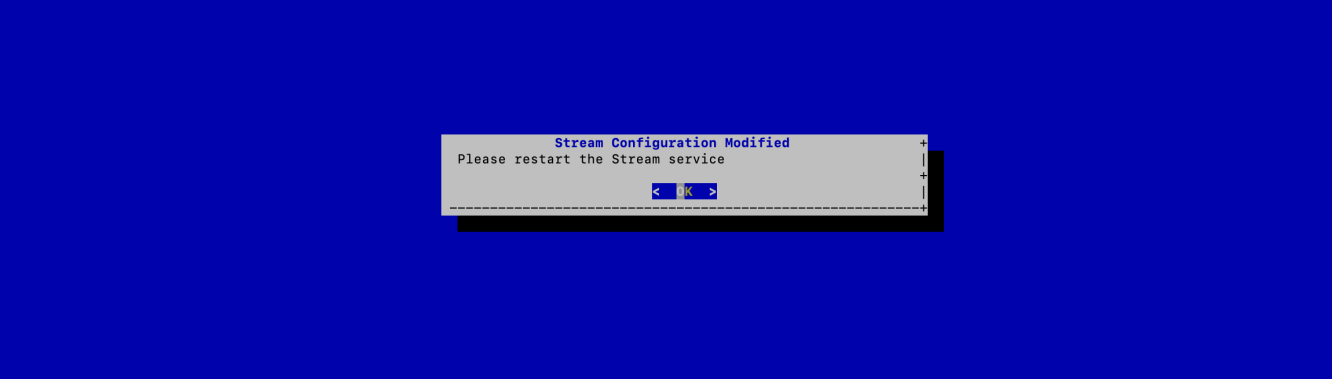
In the Akka_Play menu, select '**SECRET**':



Validate the new Stream Application Secret:



The Stream configuration is updated:



Stream Configuration Modified
Please restart the Stream service

< OK >

For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

JVM Configuration

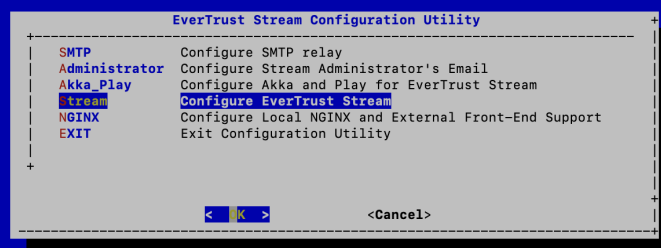
Stream allows you to configure the **Xms** (minimum memory allocation pool) and **Xmx** (maximum memory allocation pool) parameters of the JVM running Stream using the configuration tool.

Connect to the server with an account with administrative privileges;

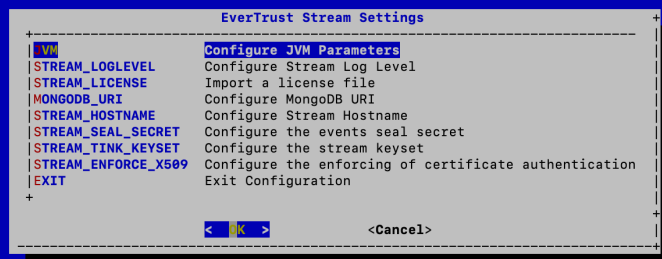
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

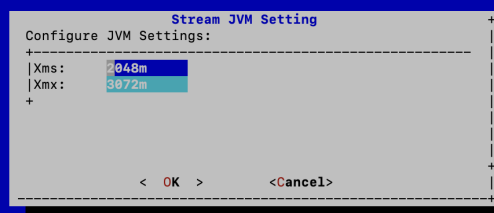
In the configuration menu, select **Stream**:



In the Stream configuration menu, Select **JVM**:



Specify the 2048 for **xms** and 3072 for **xmx** parameters and select 'OK':



The new JVM parameters are configured.

For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

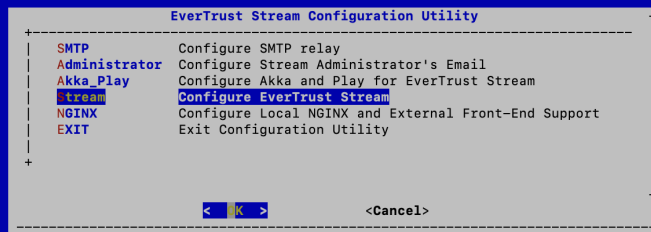

MongoDB URI Configuration

Connect to the server with an account with administrative privileges;

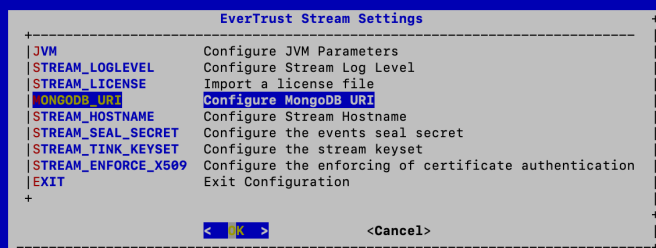
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

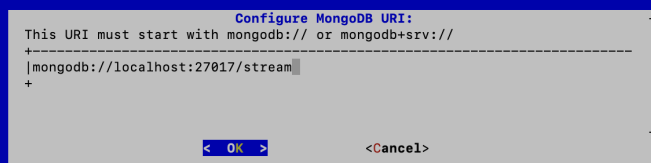
In the main menu, select **Stream**:



In the Stream configuration menu, Select **MONGODB_URI**:



Specify the MongoDB URI to target your MongoDB instance:



NOTE

Stream is installed to target a local MongoDB instance by default.

If you use an external MongoDB (such as MongoDB Atlas Database or dedicated On-premises database) instance:

- Create a user with "read/write" permissions on your MongoDB instance;
- Create a replicaSet if using a MongoDB cluster;
- Specify a MongoDB URI that does match your context.

External MongoDB database URI syntax

```
mongodb+srv://<user>:<password>@<hostname>:<port>/stream
```

External MongoDB cluster of databases URI syntax

```
mongodb+srv://<user>:<password>@<hostname1>:<port1>,<hostname-2>:<port2>/stream?replicaSet=<replicaset>&authSource=admin
```

The MongoURI is configured.

For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

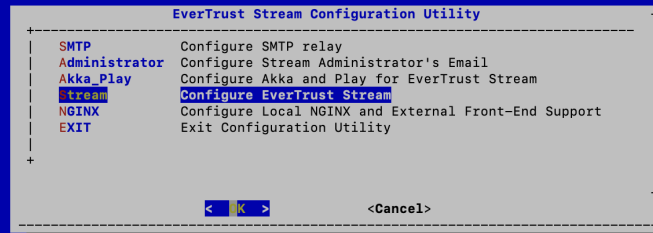
Stream Hostname Configuration

Connect to the server with an account with administrative privileges;

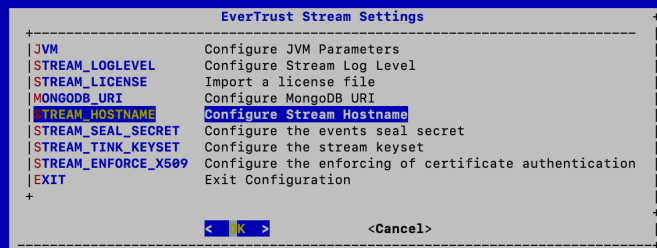
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

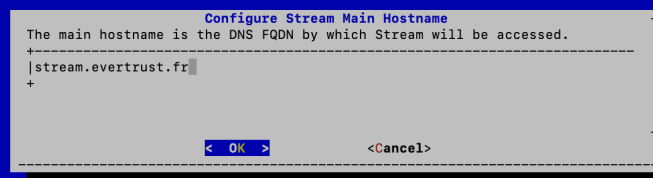
In the main menu, select **Stream**:



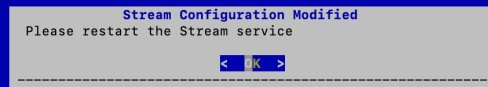
In the Stream configuration menu, Select **STREAM_HOSTNAME**:



Specify the DNS FQDN by which Stream will be accessed:



The Stream Hostname is configured:

A terminal window with a blue background. In the center, there is a white rectangular box with a thin black border. Inside the box, the text "Stream Configuration Modified" is displayed in blue, and "Please restart the Stream service" is displayed in black. At the bottom of the box, there are three small icons: a left arrow, a key, and a right arrow.

```
Stream Configuration Modified
Please restart the Stream service
< [key] >
```

For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

Generating an event seal secret

Stream will generate functional events when using the software.

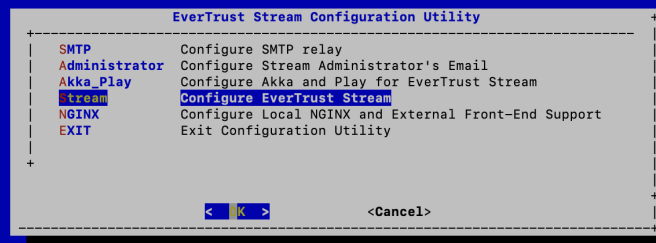
These events are typically signed and chained to ensure their integrity. Therefore, you must specify a sealing secret for this feature to work properly.

Connect to the server with an account with administrative privileges;

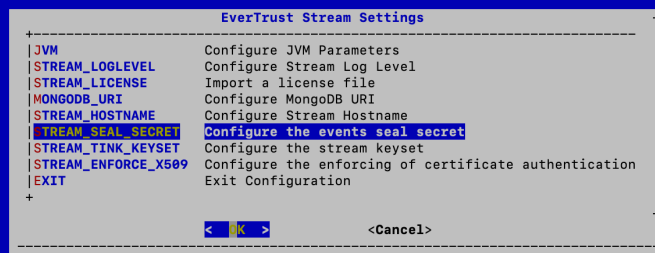
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

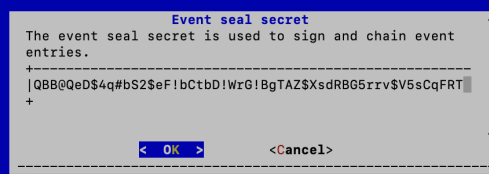
In the main menu, select '**Stream**':



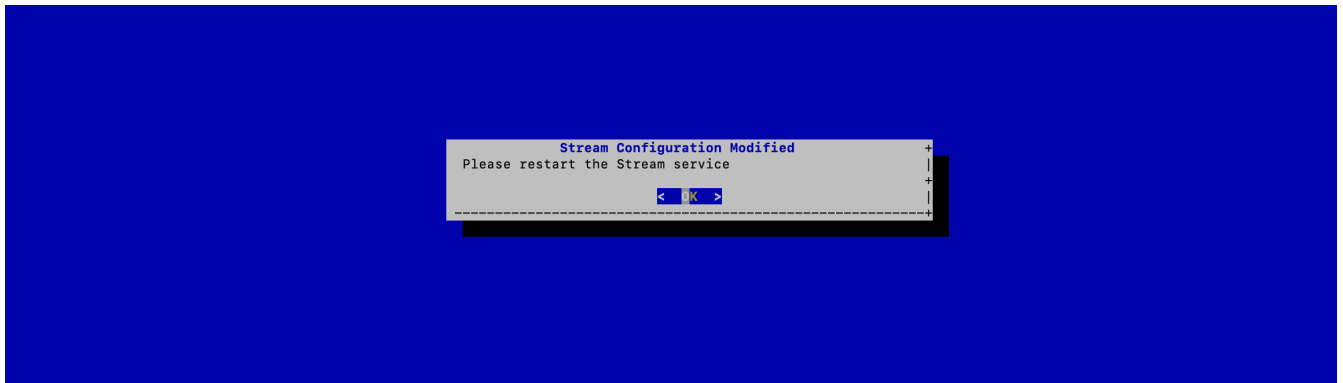
In the Stream menu, select '**STREAM_SEAL_SECRET**':



Validate the new event seal secret:



The even seal secret is now configured:



For the changes to take effect, you must restart the Stream service by running:

```
# systemctl restart stream
```

Installing the Stream license

NOTE

You should have been provided with a `stream.lic` file. This file is a license file and indicates an end of support date.

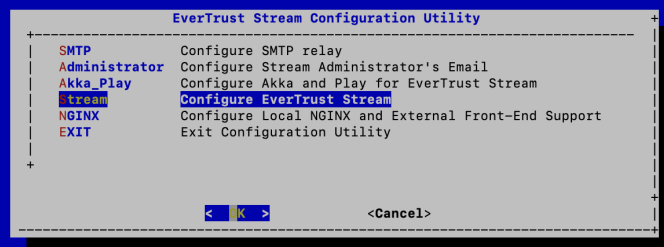
Upload the `stream.lic` file (using SCP or other means) under `/tmp/stream.lic`;

Connect to the server with an account with administrative privileges;

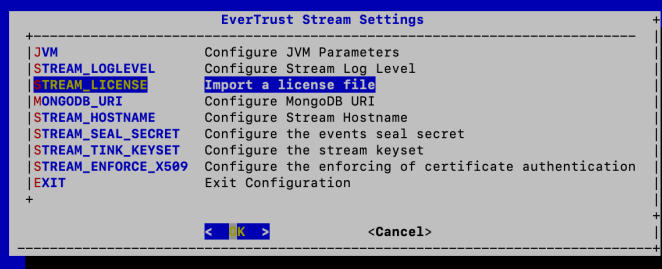
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

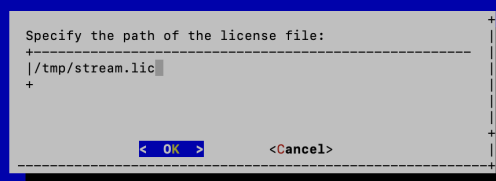
In the main menu, select **Stream**:



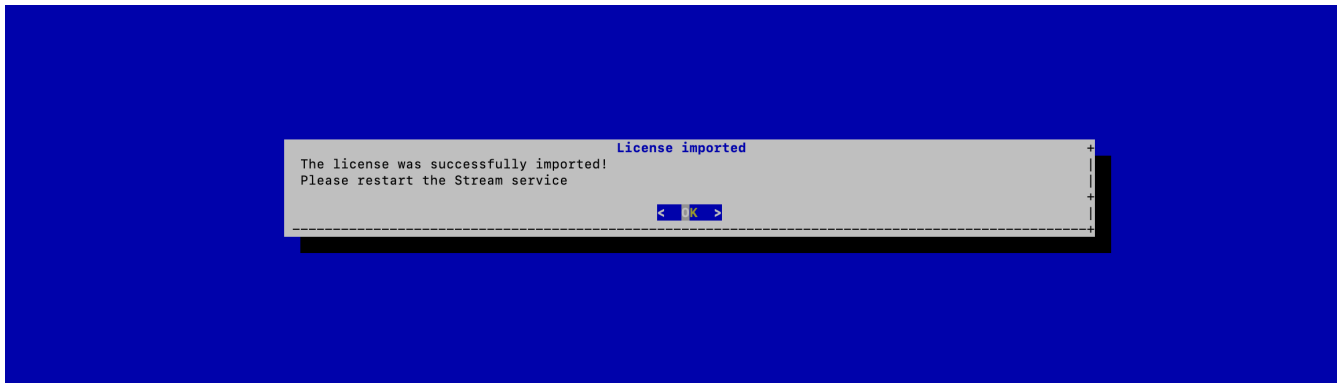
In the Stream configuration menu, Select **STREAM_LICENSE**:



Specify the path `/tmp/stream.lic` and validate:



The Stream License is configured:



For the changes to take effect, you must restart the Stream service by running:

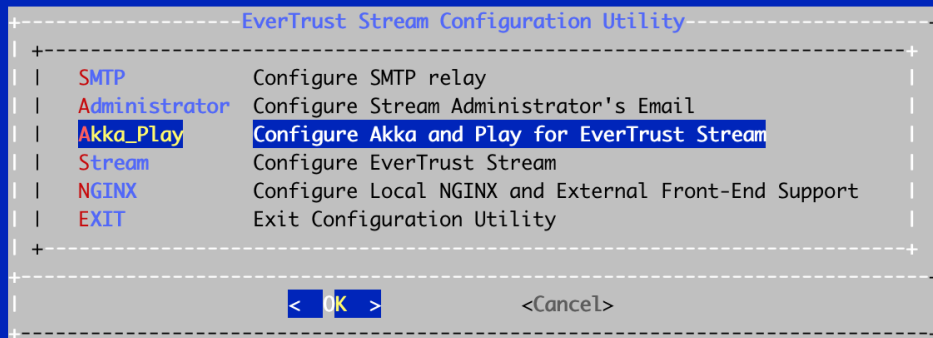
```
# systemctl restart stream
```


Installing Stream on a cluster of servers

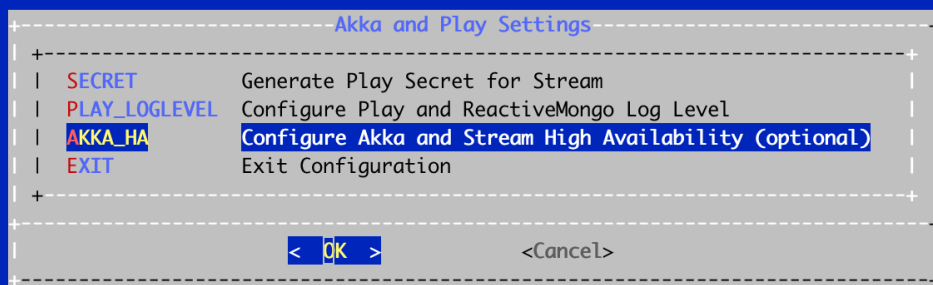
WARNING

This section must not be followed if you plan on deploying Stream in standalone mode (vs cluster mode). WARNING: This section does not explain how to install Stream on a Kubernetes cluster. Please refer to the dedicated section.

In the main menu, select '**Akka_Play**':



In the Akka_Play menu, select '**AKKA_HA**':



In this menu, specify either the IP address or the DNS name for each server that will be running Stream on this cluster with akka management port, as well as the local node index (the number of the node that you are configuring at that moment).

NOTE

Note that the local node index must match the current node hostname or ip parameter:

```
-----HA: Akka Nodes Configuration for Stream-----
| Enter 2 or 3 Akka Nodes information if you want to setup High Availability. |
| Leave empty otherwise. |
| +-----+ |
| |HA nodes in hostname:port format, comma separated: |
| |node1.stream.fr:8558,node2.stream.fr:8558,node3.stream.fr:8558 |
| |Local Node Index (starts at 0):0 |
| |Artery of the local node in hostname:port format: |
| |node1.stream.fr:25520 |
| +-----+ |
| |< OK > |<Cancel> |
| +-----+ |
```

Save your changes from the menu.

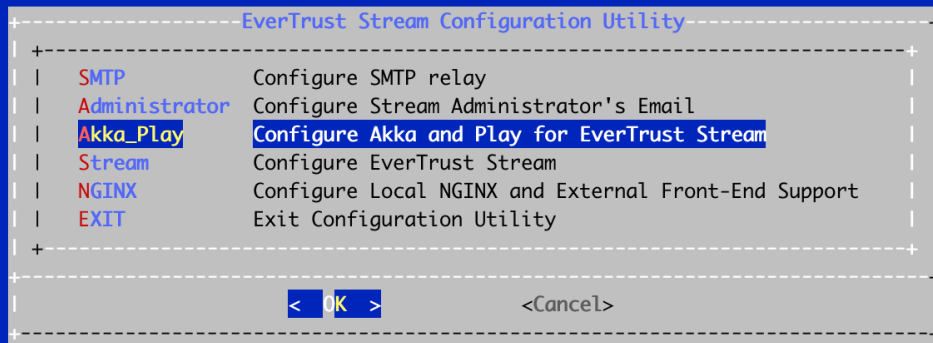
The High Availability mode is now configured on the current node:

```
-----Stream Configuration Modified-----
| WARNING: you modified the /etc/default/stream file. |
| That file MUST be exactly the same on all the nodes, except for the |
| AKKA_MANAGEMENT_LOCAL (Local Node Index) parameter. |
| Once configuration has been adjusted on all nodes, please restart Stream. |
| +-----+ |
| |< OK > |
| +-----+ |
```

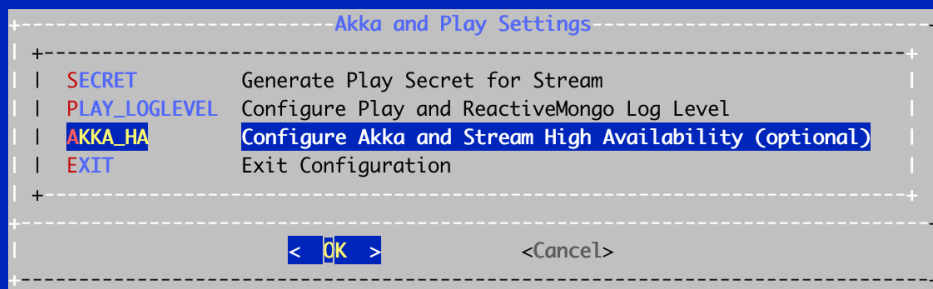
You must now configure your other nodes, but because they belong to the same cluster they need to share the **same akka play secret**, the **same stream licence**, the **same stream seal secret**, the **same stream hostname**, the **same mongo database**, the **same x509 enforcing** and the **same stream tink keyset**. In order to be able to do that, you need to copy the configuration file that was generated by the stream-config app, named `/etc/default/stream` and paste it on each one of your nodes;

Then on each other node, run the Stream Configuration utility with the following command:

```
$ /opt/stream/sbin/stream-config
```



In the Akka_Play menu, select 'AKKA_HA':



Here, you need to change the local node index to match the hostname of the node that you are configuring:

```
-----HA: Akka Nodes Configuration for Stream-----
| Enter 2 or 3 Akka Nodes information if you want to setup High Availability. |
| Leave empty otherwise. |
|-----+-----|
| HA nodes in hostname:port format, comma separated: |
| node1.stream.fr:8558,node2.stream.fr:8558,node3.stream.fr:8558 |
| Local Node Index (starts at 0):1 |
| Artery of the local node in hostname:port format: |
| node2.stream.fr:25520 |
|-----+-----|
| < OK > <Cancel> |
```

WARNING

You will need to import the Stream licence file on each node manually, following the guidelines of section [Installing the Stream license](#).

Additionally, on each node, you will need to open the ports used for Akka_HA and Akka_MGMT, which are by default 25520 and 8558:

```
$ firewall-cmd --permanent --add-port=25520/tcp
$ firewall-cmd --permanent --add-port=8558/tcp
```

Reload the firewall configuration with:

```
$ systemctl restart firewalld
```

Restart the Stream service on each one of the nodes:

```
$ systemctl restart stream
```

2.3.2. Bootstrapping EverTrust Stream

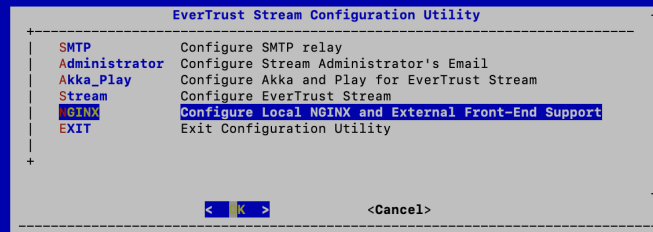
Installing a bootstrap certificate

Connect to the server with an account with administrative privileges;

Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

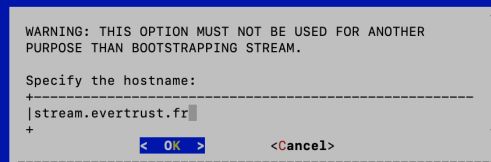
In the main menu, select '**NGINX**':



In the NGINX menu, select '**BOOTSTRAP**':



Specify the DNS Name of the Stream server (the same that you used as Stream hostname previously):



The self-signed certificate is going to be generated and automatically installed for Nginx to use directly:



CAUTION

This certificate is meant to be used to bootstrap Stream and should be replaced

as quickly as possible as it is highly unsecured.

Starting the services

Assuming that all prior configuration operations have been performed as documented in the installation guide and that a bootstrap certificate has been installed as explained in the previous section, the services must be started:

1. Connect to the server with an account with administrative privileges;
2. Start the stream service with the following command:

```
# systemctl start stream
```

3. Verify the NGINX configuration with the following command:

```
# nginx -t
```

4. Restart the NGINX service with the following command:

```
# systemctl restart nginx
```

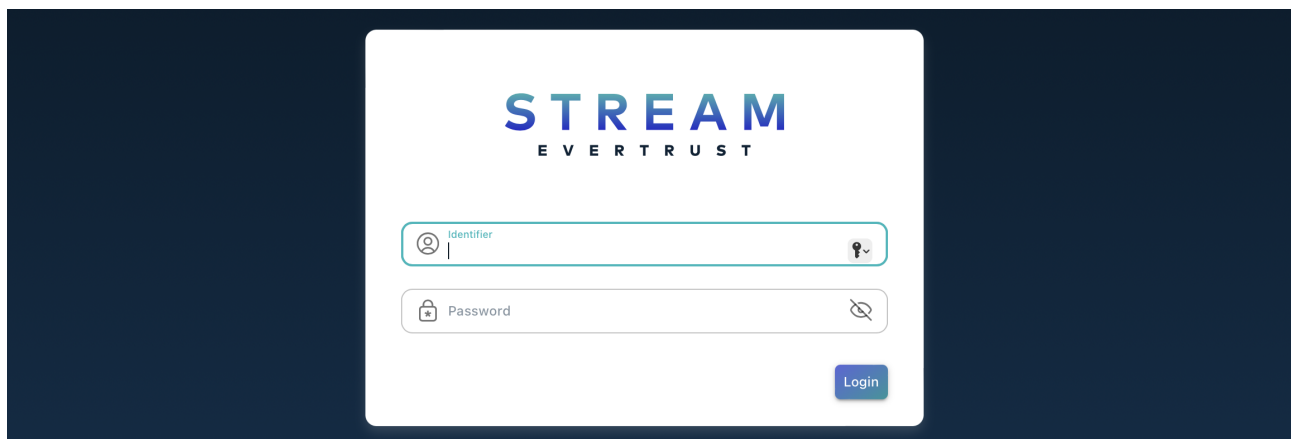
Retrieving initial password

For the first log-in, you must find the administrator password in the `/opt/stream/var/run/adminPassword` file.

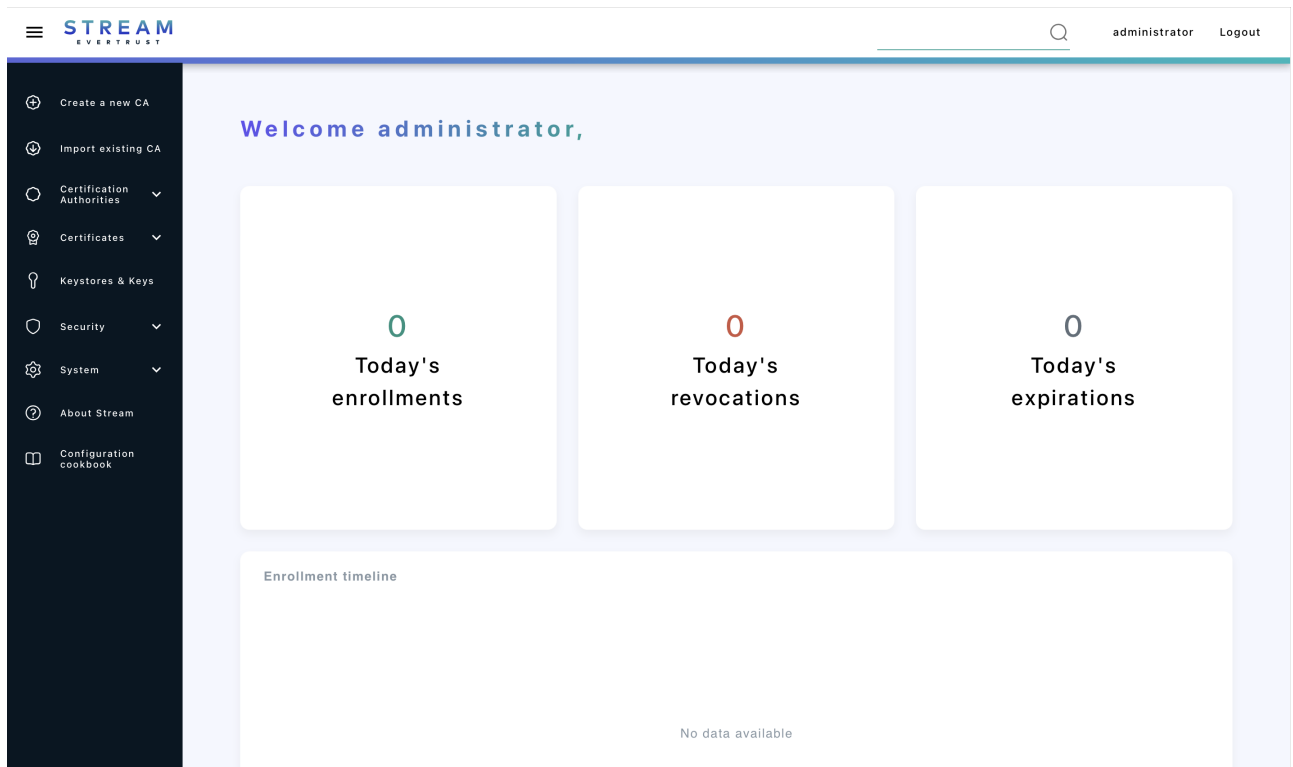
The default administration login is `administrator`.

Accessing the Stream Web Interface

1. Launch a web browser;
2. Browse to `https://[IP or DNS Name of the Stream component]/ui#`;
3. When prompted with a security issue, click on the button to accept the risks and proceed anyway. This alert is raised by the use of the self-signed certificate.



4. Specify the default administration credentials and hit the '**Login**' button:



CAUTION

It is **highly recommended** to delete the adminPassword file from your machine once you saved it somewhere safe.

2.3.3. Initial Key Ceremony

Before deploying to production, the initial key ceremony should take place

Configure a keystore

To protect the keys, keystores (cloud or physical) should be configured. Follow the Administration Guide steps in **Managing Keystores & Keys › Keystores in Stream** to configure your Keystore.

Create keys

A key should be created for each Certification you wish to add. The keys can be generated externally, or using Stream.

Key creation steps depend on the type of keystore:

- **KMS:**

KMS keys can be created using Stream following the Administration Guide steps in **Managing Keystores & Keys › Managing keys in Stream › Cloud KMS** or directly in the KMS following your KMS documentation.

- **Software Keystore:**

Software keys can be created using Stream following the Administration Guide steps in **Managing Keystores & Keys › Managing keys in Stream › Software keystore**.

- **Hardware Security Module:**

HSM keys can be created using Stream following the Administration Guide steps in **Managing Keystores & Keys › Managing keys in Stream › PKCS#11 HSM**. Please note that extra steps may be required at HSM level depending on the model of HSM used.

Once the keys have been created, they should appear in the keystore on Stream after a refresh.

Create your Certification Authorities

Once keys have been created the Certification Authorities can be created following the Administration Guide steps in **Managing Certification Authorities**.

2.3.4. Finalizing Stream Configuration

Configuring Stream default templates

If you intend to use Stream as your certification authority, default templates should be created.

As Stream is template-oriented, default templates will be used when enrolling certificates.

Two templates are considered as defaults:

- A **tlsClient** template for TLS client certificates
- A **tlsServer** template for TLS server certificates

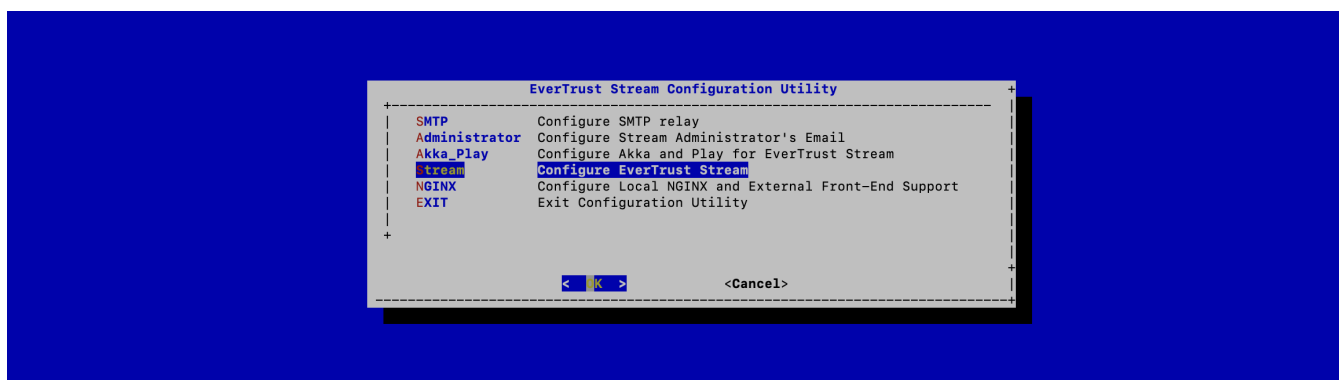
To create these templates:

Connect to the server with an account with administrative privileges;

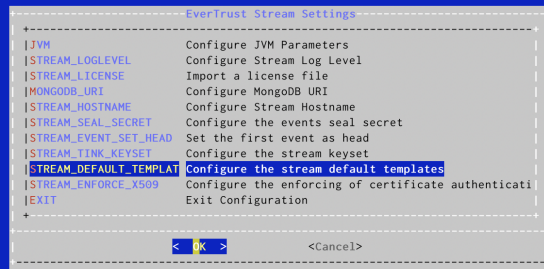
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

In the main menu, select '**Stream**':



In the Stream menu, select '**STREAM_DEFAULT_TEMPLATE**':



The templates are automatically created and available in **Certificates › Templates**.

NOTE `mongosh` must be installed and the mongo URI configured for this configuration to work

Installing a Server Authentication Certificate

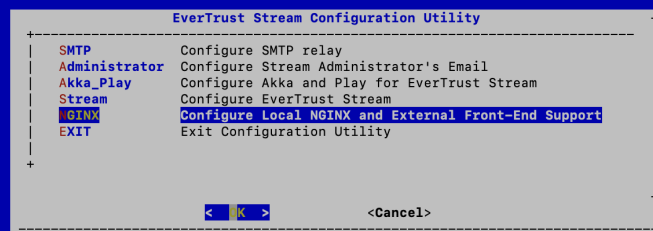
Issuing a Certificate Request (PKCS#10)

Connect to the server with an account with administrative privileges;

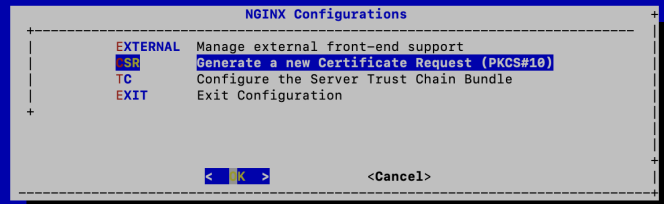
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

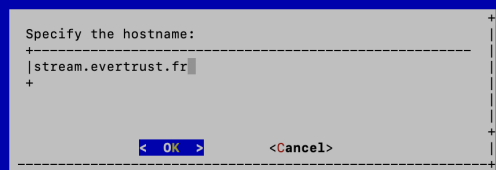
In the main menu, select '**NGINX**':



In the NGINX menu, select '**CSR**':



Specify the DNS Name of the Stream server (the same that you used as Stream hostname previously):



The certificate request is generated and available under '*etc/nginx/ssl/stream.csr.new*':

A terminal window with a blue background. A grey box contains the text: "New NGINX Server Certificate Request" followed by "The new request is available under "/etc/nginx/ssl/stream.csr.new". At the bottom of the box are navigation arrows and the text "9K".

```
New NGINX Server Certificate Request
The new request is available under "/etc/nginx/ssl/stream.csr.new"
< 9K >
```

Signing the certificate

The CSR generated at the previous steps must then be signed using an existing PKI. If you do not have an existing PKI, please refer to the [Key Ceremony Documentation](#) to create one.

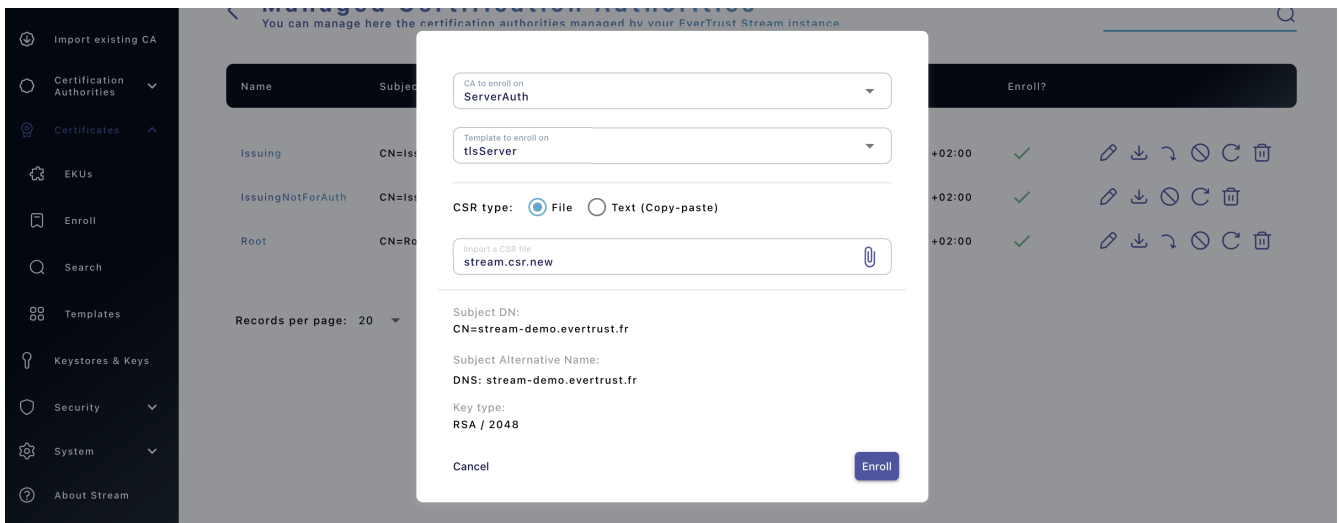
Sign using Stream

NOTE

This step assumes that the Key Ceremony already happened, i.e the operational CAs are imported into Stream as managed CAs.

1. Access Stream's web administration console and log in using the default administration credentials that were obtained at [this section](#);
2. Browse to '**Certificates > Enroll**';
3. Fill in the following information:
 - CA to enroll on: Select the CA that is issuing the server certificates for your organization;
 - Template to enroll on: Select the certificate template that was created at [this step](#) that should be named 'tlsServer';
 - CSR type: Select 'File', then click the paper clip icon and import the CSR that was generated at [this step](#);

Then click 'Enroll':



You can retrieve the enrolled certificate in PEM format from the '**Certificates > Search**' : download this certificate in PEM

Sign using an external Certification Authority

NOTE

The certificate must have the **serverAuthentication** Extended Key Usage

You will need to provide your certificate authority with the `/etc/nginx/ssl/stream.csr.new` file that was generated at the previous step.

Installing the Server Certificate

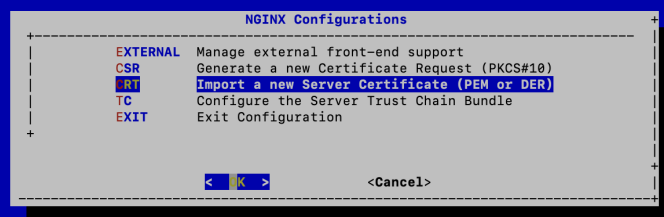
Upload the signed server certificate (in PEM format) on the Stream server under `/tmp/stream.crt` (using SCP or other means);

Connect to the server with an account with administrative privileges;

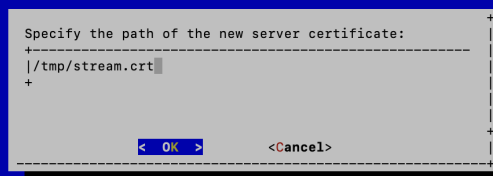
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

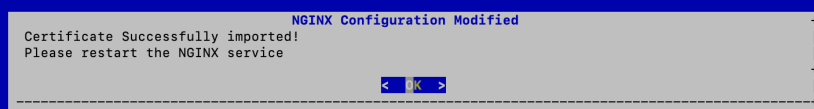
In the NGINX configuration menu, select '**CRT**':



Specify the path `/tmp/stream.crt` and validate:



The server certificate is successfully installed:



Installing the Server Certificate Trust Chain

NOTE

You must follow this section only if you signed the server certificate with an existing PKI. If you self-signed the server certificate, you do not need to follow this step.

Upload the server certificate trust chain (the concatenation of the Certificate Authority certificates in PEM format) on the Stream server under `/tmp/stream.bundle` (using SCP or other means);

WARNING

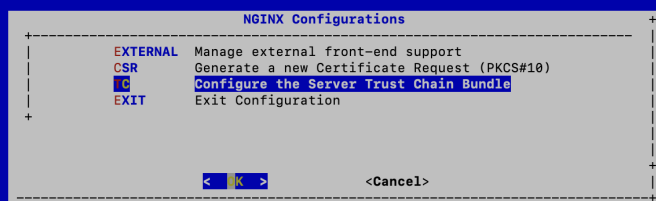
The bundle should contain only the Certificate Authority certificates in PEM format and **NOT** the server certificate

Connect to the server with an account with administrative privileges;

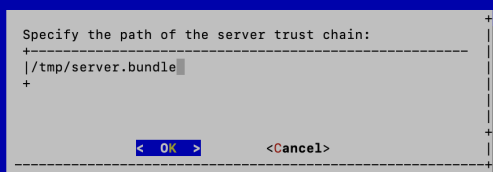
Start the Stream configuration utility by running:

```
# /opt/stream/sbin/stream-config
```

In the NGINX configuration menu, select **'TC'**:



Specify the path `/tmp/stream.bundle` and validate:



The server bundle is successfully installed:

NGINX Configuration Modified
Server Trust Chain successfully imported!
Please restart the NGINX service

< [K] >

Verify the NGINX configuration with the following command:

```
# nginx -t
```

Restart the NGINX service with the following command:

```
# systemctl restart nginx
```

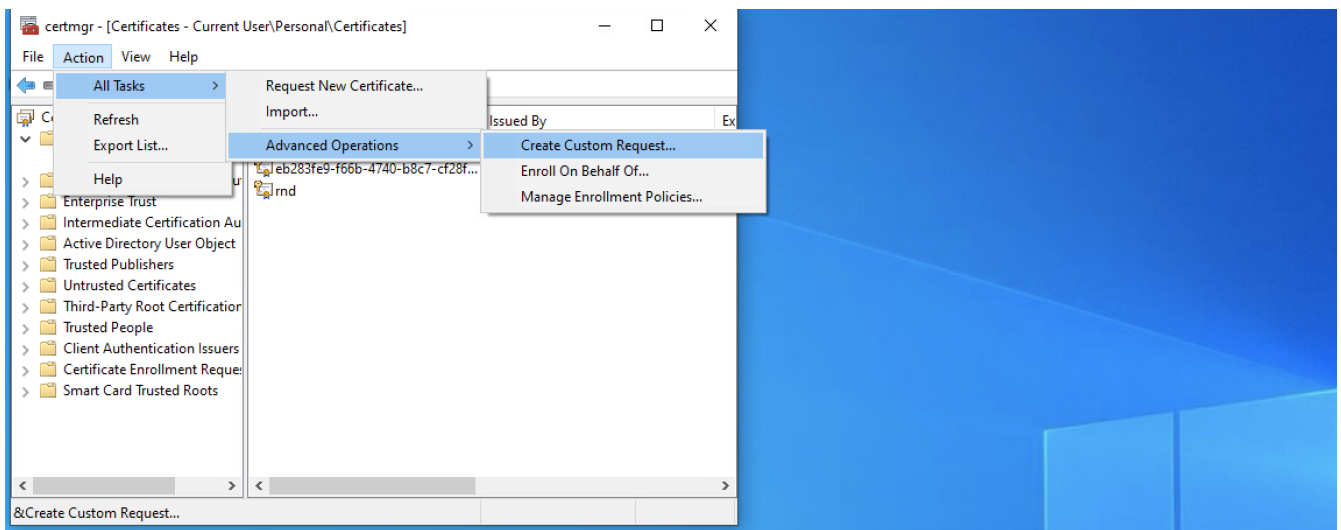
Configuring administrator certificate authentication

To authenticate as administrator on stream using a certificate, it is possible to either use a certificate generated on Stream, or to use an external certificate.

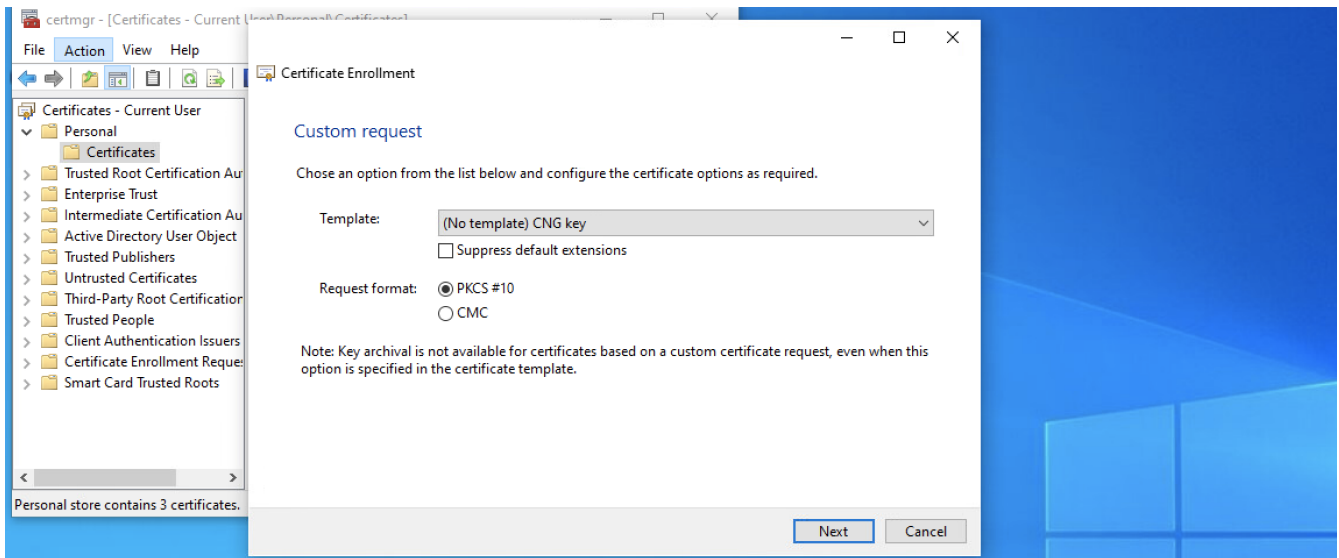
Issuing a Certificate Request (PKCS#10)

Generating the CSR on Windows

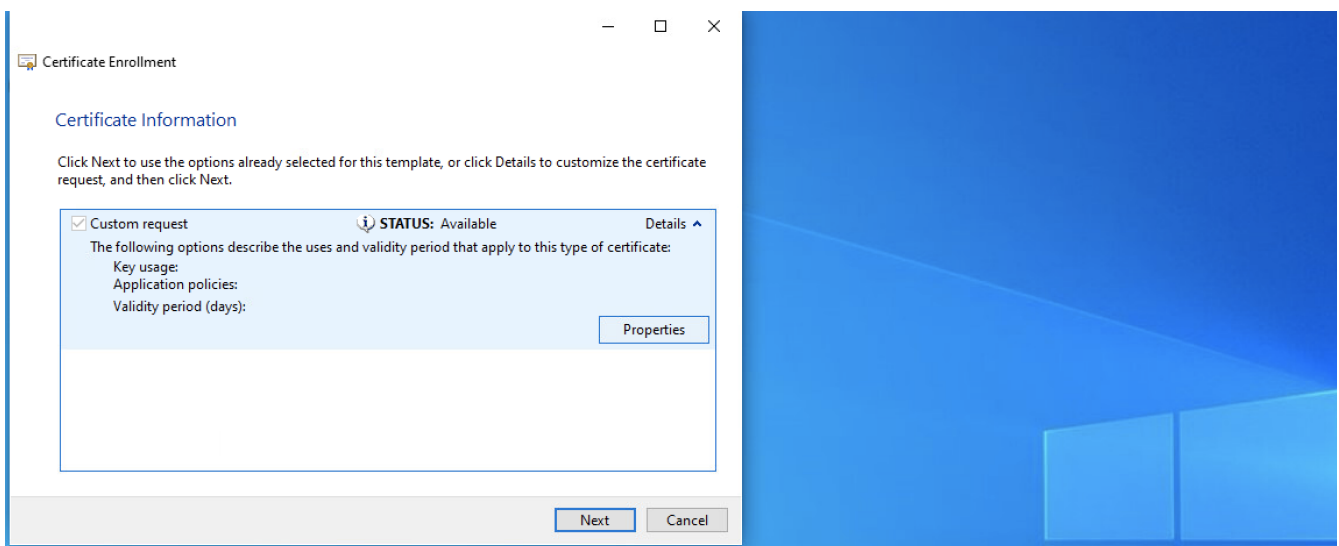
1. Press 'Windows Key + R' then type 'certmgr.msc' and press enter;
2. Expand 'Personal' then double click 'Certificates';
3. In the top bar, click 'Action > All Tasks > Advanced Operations > Create custom requests':



4. On the 'Before You Begin' and the 'Select Certificate Enrollment Policy' screens, click 'Next';
5. In the 'Custom request' window, select 'CNG Key' in the template drop-down menu and 'PKCS#10' as the request format, then click 'Next':



6. In the 'Certificate Information' window, click 'Details' then 'Properties':

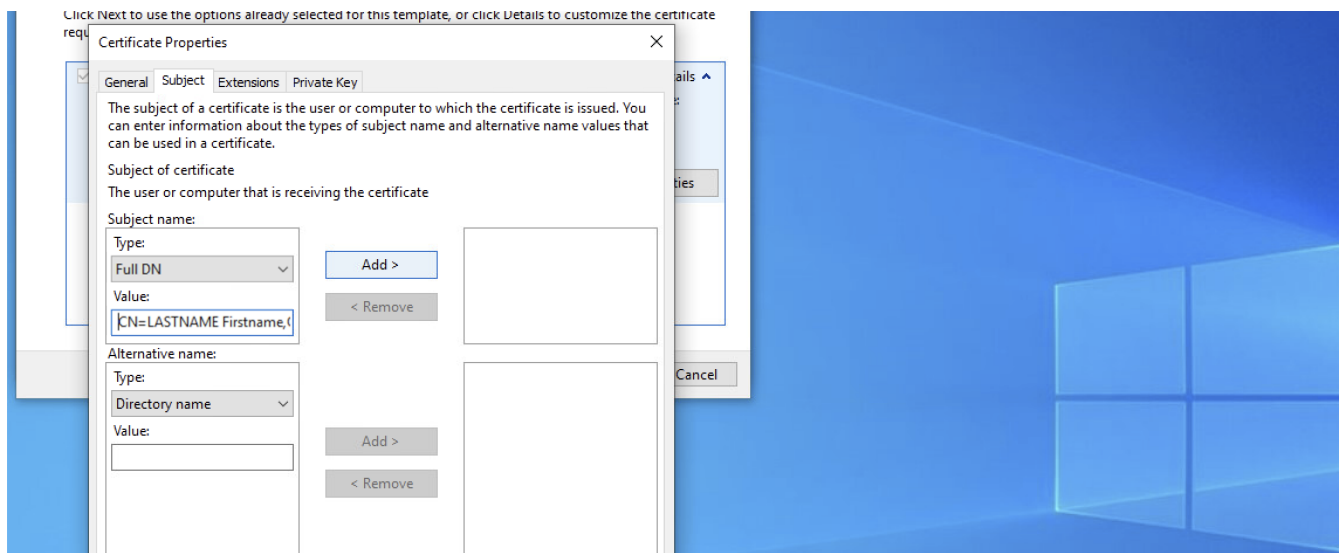


7. In the 'Certificate Properties' pop-up that opened:

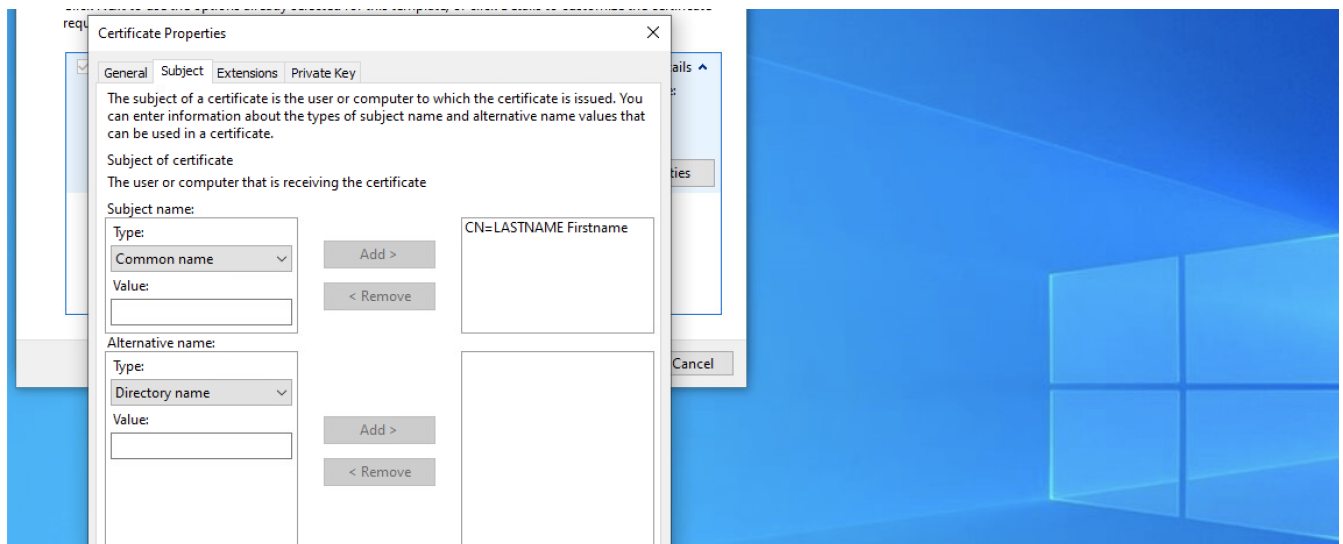
- General tab: Leave everything as default
- Subject tab: In the subject name category enter the DN using the standard DN format then click Add:

NOTE

It is recommended for a certificate DN to contain at least a CN, OU, O and C element

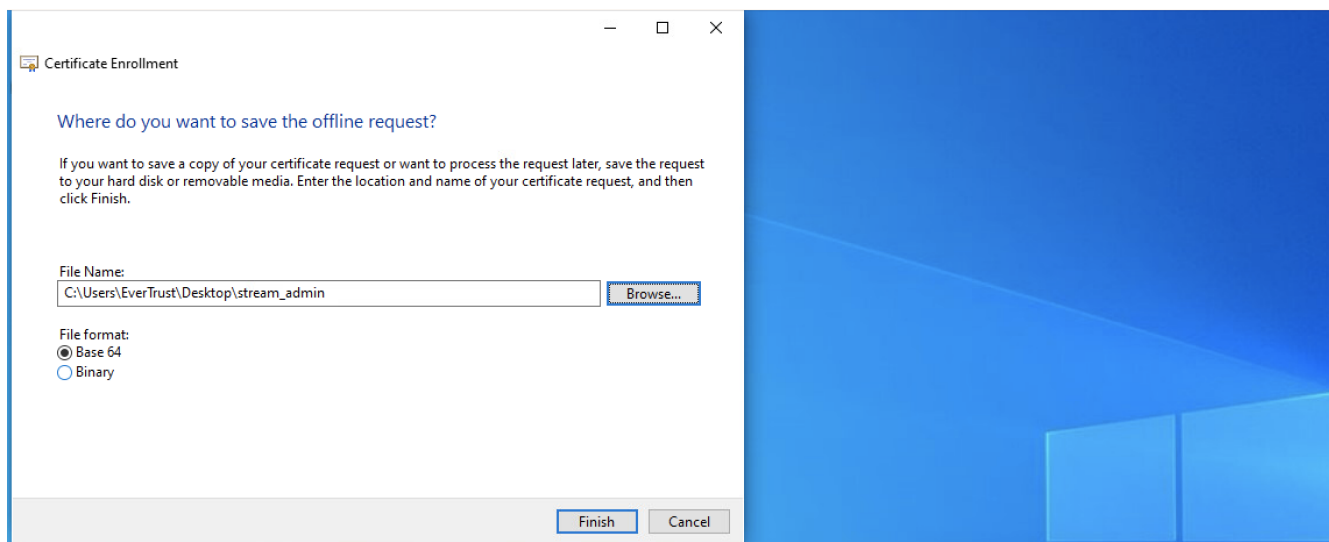


- Extensions tab: Leave everything as default
- Private Key tab: Expand 'Key options' and select '2048' for the key size, then expand 'Select Hash Algorithm' and select 'sha256' for the hash algorithm:



Click apply, then OK, then Next;

8. In the next window, select a place to save the CSR and select 'Base64' as File format, then click 'Finish':



Generating the CSR on Linux/MacOS

1. Start a shell instance;
2. Run the following command, replacing the DN with your information:

```
$ openssl req -new -newkey rsa:2048 -subj "/CN=LASTNAME Firstname/C=FR/OU=.../"  
-keyout ./stream_admin.key -out ./stream_admin.csr
```

This command will prompt you for a password to encrypt the CSR private Key

NOTE It is recommended for a certificate DN to contain at least a CN, OU, O and C element

Signing the certificate

Using a Stream-signed certificate

NOTE This step assumes that the Key Ceremony already happened, i.e the operational CAs are imported into Stream as managed CAs.

1. Access Stream's web administration console and log in using the default administration credentials that were obtained at [this section](#);
2. Browse to '**Certificates > Enroll**';
3. Fill in the following information:
 - CA to enroll on: Select the CA that is issuing the user certificates for your organization. Note that it must have the 'Is trusted for client authentication' switch turned on in the interface;
 - Template to enroll on: Select the certificate template that was created at [this step](#) that should be named 'tlsClient';
 - CSR type: Select 'File', then click the paper clip icon and import the CSR that was generated at [this step](#);

Then click 'Enroll':

CA to enroll on
UserAuthCA

Template to enroll on
tlsClient

CSR type: ☒ File ☐ Text (Copy-paste)

Import a CSR file
stream_admin.csr

Subject DN:
CN=LASTNAME Firstname

No SAN and no extension

Key type:
RSA / 2048

Cancel Enroll

You can retrieve the enrolled certificate in PEM format from the '**Certificates > Search**' : download this certificate in PEM

Using an external-signed certificate

NOTE

The certificate must contain the clientAuthentication Extended Key Usage in order to be used as an authentication certificate on Stream

Trusting the issuing CA for client authentication

In order for the client authentication certificate to work as intended, the issuing CA must be trusted for client authentication on Stream.

- If your certificate was signed by a Stream managed CA:
Browse to **Certification Authorities > Managed CAs**, select your Issuing CA and toggle on '**Is trusted for client authentication**'
- If your certificate was signed by an external CA:
If your CA is not yet imported, import it using the administration guide steps in **Managing Certification Authorities > Importing an External Certification Authority**.
Browse to **Certification Authorities > External CAs**, select your Issuing CA and toggle on '**Is trusted for client authentication**'

Creating the SuperAdmin role in Stream

At this point, you should be logged into the Stream web administration console with the default administrator account.

1. Go to **Security > Roles** and click  ;

2. Fill-in the following information:



- **Name:** SuperAdmin;

- **Description** : Super administrator role that has all rights on Stream. Use with caution;
- **Configuration permissions** : Click the '+' button then click on 'Section' and select 'All permissions of all type' and click the 'Add' button;
- **Lifecycle permissions** : Click the '+' button, then :
 - Click on 'CA' and select 'All';
 - Click on 'Template' and select 'All';
 - Click on 'Rights' and select 'All';
 - Click the 'Add' button.

3. Click the 'Save' button at the top of the page.

The SuperAdmin role is now created.

Giving the SuperAdministrator role to the certificate

1. Access Stream's web administration console and log in using the default administration credentials that were obtained at [this section](#);
2. Browse to '**Security > Authorizations**' then click  and select 'Add authorization manually';
3. Import a certificate by clicking on certificate button  ;
4. Once the identifier has been filled with the certificate DN click the 'Add' button;
4. Click the 'Add role' button and select the 'SuperAdmin' role previously created from the drop down menu;
5. Click the 'Save' button at the top of the page;

Now the PEM certificate has the SuperAdmin permission in Stream.

Creating a PKCS#12 file from the private key and the PEM

This step can only be done using OpenSSL.

1. Connect to the server with an account with administrative privileges;
2. Upload the PEM of the certificate you used to create the SuperAdmin user at the previous step in the same folder where the private key that was used to generate the CSR was created in;
3. Run the following command:

```
$ openssl pkcs12 -export -inkey ./stream_admin.key -in ./stream_admin.pem -name
StreamAdministrator -out ./stream_admin.p12
```

This will create a PKCS#12 file containing the PEM of the certificate and its private key in PKCS#8 format (encrypted). The previous utility will first ask the private key file pass phrase and then ask

for a passphrase to protect the PKCS#12.

NOTE

If using an openssl version greater than 3, you will need to add **-legacy** at the end of the command as the new version of the PKCS#12 is not widely supported.

4. Import this .p12 file into your certificate store or the certificate store of your browser if it has one (ex: Firefox). At the import, you should be prompted to enter the passphrase that you used to encrypt the PKCS#12.

Now you should have the certificate imported and ready to be used.

When going to the Stream web interface while not being logged in, your web browser should prompt you for a certificate to use and you should be able to select the one that you imported just now.

Removing the administrator local account

CAUTION

Before deleting the account, ensure that your administrator certificate is working as intended.

1. Access Stream's web administration console and log in using the client certificate that was set-up at the previous step;
2. Browse to **'Security > Local accounts'**;
3. Click the bin icon next to the *'administrator'* account and confirm deletion.

NOTE

After reviewing the **Security Guidelines**, Stream is ready to leave its confined environment.

2.4. Security Guidelines

The following content are guidelines to have a secure Stream installation.

Stream should run on a dedicated machine. From this fact, all unused packages should be removed from the machine. The system should have been installed following the security guidelines recommended by the operating system vendor.

The following requirements should be met:

- SELinux should be enabled
- The **stream-hardening** rpm should be installed
- Privileged Access Management or SSH/Sudoers should be set up
- The firewall should be enabled and ports 80 and 443 allowed
- Only certificate authentication should be enabled once the product has been initialized and the initial Key Ceremony phase performed
- The NGINX configuration should be modified following the below procedure

- Stacktrace logging in events should be disabled

On top of that, though it is not mandatory, it is recommended to set up other security-related solutions, such as a Web Application Firewall, an Intrusion Detection System and a Security Information and Event Management.

All the following steps should be followed to ensure compliance if they are not already implemented with the above requirements, and should be done with an account with administrative privileges.

2.4.1. SELinux

To enhance security, SELinux should be enabled.

```
# setenforce Enforcing
```

To ensure that it is enabled, run the following command

```
# getenforce
```

This should return **Enforcing**

2.4.2. Install the stream-hardening rpm

Follow the same steps as in Installing Stream but for the **stream-hardening** rpm.

Once the rpm is installed, a system reboot is necessary. The following command can be used:

```
# reboot now
```

CAUTION

In order to install the stream hardening policies, the server must have access to a repository (mirror, iso file, ...) of the linux distribution you are using in order to be able to install the dependencies of the software. The **stream-hardening** package has the following dependencies:

- **stream**
- **policycoreutils-python-utils**

Please note that these packages may have their own dependencies.

2.4.3. Sudoers

To administrate Stream without using the root user, **stream-hardening** rpm creates a **stream-administrator** group with sudoers permissions.

Create a new user with `stream-administrator` and `stream` groups, for instance, user-admin.

```
# useradd -G stream-administrator,stream user-admin
# passwd user-admin
```

Link user-admin to the selinux `sysadm_u` user

```
# semanage login -a -s sysadm_u -rs0:c0.c1023 user-admin
```

NOTE

The `semanage` command is available with the `polycoreutils-python-utils` package.

Relabel the user-admin user home folder with the following command

```
# restorecon -FR -v /home/user-admin
```

In case you need to access the user-admin user account via ssh you will need to set the selinux `ssh_sysadm_login` boolean

```
# setsebool -P ssh_sysadm_login on
```

NOTE

The `setsebool` command is available with the `polycoreutils` package.

Now the user-admin can:

- Manage mongodb server with `systemctl`
- Manage nginx server with `systemctl`
- Manage stream server with `systemctl`
- Execute every script under the folder `/opt/stream/sbin/` as a root user

2.4.4. Configuring the Firewall

The firewall should have been configured at the `setup` step.

In addition to this configuration, the https (443) access should be restricted to :

- The Stream administrators
- External components using Stream certificate lifecycle capabilities (Stream for example)

WARNING

Firewalld sometimes has default ports allowed. No other ports than those referenced in the `setup` step should be allowed.

2.4.5. X509 Enforcing

In order to improve security once an administration certificate has been emitted, all authentication modes should be disabled apart from certificate authentication.

To do that, please follow the dedicated steps in the security section of the administration guide : **Managing Security › Enforce Certificate Authentication**

2.4.6. NGINX Configuration

In order to improve security, the default NGINX configuration should be altered. In the configuration file in `/etc/nginx/nginx.conf`, the `server` instruction block containing the `listen 80` instruction should be deleted or commented.

Once an administration certificate has been emitted, the `/opt/stream/etc/stream-httpd.conf` should be updated. The following line

```
ssl_verify_client          optional_no_ca;
```

should be replaced by the lines

```
ssl_verify_client          on;
ssl_client_certificate      ssl/client-trusted-cas.pem;
ssl_trusted_certificate     ssl/trusted-cas.pem;
ssl_crl                    ssl/crl-bundle.pem;
```

This ensures only valid and trusted certificates can be used to authenticate on the Stream server.

For the following example chain :

Root CA → Client Issuing CA 1

Root CA → Client Issuing CA 2

The `/etc/nginx/ssl/client-trusted-cas.pem` file should contain the PEM certificates of the CAs trusted for client authentication, concatenated one after the other. It should look like:

```
-----BEGIN CERTIFICATE-----
<Client Issuing CA 1 PEM>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Client Issuing CA 2 PEM>
-----END CERTIFICATE-----
```

The `/etc/nginx/ssl/trusted-cas.pem` file should contain the PEM certificates of the **chain** to the CAs trusted for client authentication, concatenated one after the other. It should look like:

```
-----BEGIN CERTIFICATE-----
```

```
<Root CA PEM>
-----END CERTIFICATE-----
```

Both these files should have the following permissions:

```
# chmod 640 /etc/nginx/ssl/trusted-cas.pem
# chmod 640 /etc/nginx/ssl/client-trusted-cas.pem
# chown root:nginx /etc/nginx/ssl/trusted-cas.pem
# chown root:nginx /etc/nginx/ssl/client-trusted-cas.pem
```

To update the CRL, the `/opt/stream/etc/crl-fetching.conf` configuration file (installed with the `stream-hardening` rpm) should be customized

```
# Uncomment and edit the following line
# CRL_URLS=("<CRL URL 1>" "<CRL URL 2>")
CRL_DOWNLOAD_PATH="/etc/nginx/ssl/tmp.crl"
CRL_PEM_PATH="/etc/nginx/ssl/tmp.crl.pem"
TMP_CRL_BUNDLE_PATH="/etc/nginx/ssl/tmp.crl.bundle"
NGINX_CRL_BUNDLE_PATH="/etc/nginx/ssl/crl-bundle.pem"
```

To customize this file, after the `# Uncomment and edit the following line` comment, the `CRL_URLS` line should be uncommented and edited to have each of your CRLs URLs.

NOTE ALL of your CAs present in the `/etc/nginx/ssl/trusted-cas.pem` and `/etc/nginx/ssl/client-trusted-cas.pem` files must have their CRL downloaded.

CAUTION CRL are expected in DER format.

NOTE To fetch the CRL on stream first follow **Managing Certificate Revocation Lists › Configuring Certificate Revocation Lists for a Managed CA** in the administration guide, the CRL path is `http://localhost:9000/crls/<your CA Technical Name>`

The following file should then be put in `/etc/cron.d/nginx-crl`

```
# This cron runs every 5 minutes and execute a script replacing the CRL file for NGINX
*/5 * * * * root /opt/stream/sbin/crl-fetching
```

2.4.7. Stacktraces management

Stacktraces in the functional logs can give a lot of information about the technical architecture of the application. To disable their logging, set the parameter `stream.event.disable-stacktrace` to `true` following the steps in the **Overridable configuration parameters** section of the administration guide.

NOTE

Stacktraces are still available in the technical logs.

2.5. Upgrade

2.5.1. Upgrade the Stream installation

The first step in the upgrade procedure is to upgrade the Stream component itself.

If Stream was installed using a repository

If you installed Stream using our repository (as described in the [installation section](#)), you should:

- Unpin the Stream version by commenting out any line excluding the `stream` package in the `/etc/yum.repos.d/stream.repo` repository file :

```
[stream]
enabled=1
name=Stream Repository
# exclude=stream
```

- Run `yum update stream`

Don't forget to pin the version again by uncommenting the line that was previously commented.

If Stream was installed manually

You must retrieve the latest Stream RPM from the [EverTrust repository](#) manually using the user credentials you were provided.

Connect to the server with an account with administrative privileges;

Install the Stream package with the following command:

```
# yum localinstall stream-2.0.x-1.x86_64.rpm
```

2.5.2. Upgrade the database schema

Some Stream versions require that you run migration scripts against your database. Stream comes bundled with an `stream-upgrade` script that handles this migration logic.

Therefore, after each upgrade, you should run `stream-upgrade` to check whether new migrations should be run.

Connect to the server with an account with administrative privileges;

Run the following command:

```
# /opt/stream/sbin/stream-upgrade -t <target version>
```

In most cases, **stream-upgrade** can detect the version you're upgrading from by checking the database. If the source version is not automatically detected, you will encounter the following error:

```
*** Unable to infer the source version from your database. Specify it explicitly with  
the -s flag. ***
```

You'll have to explicitly tell **stream-upgrade** which version you are upgrading from. To do that, simply set the source version explicitly with the **-s** flag :

```
# /opt/stream/sbin/stream-upgrade -t <target version> -s <source version>
```

Similarly, **stream-upgrade** will try to use the MongoDB URI that was configured by the Stream configuration utility. If it fails to auto-detect your database URI or you wish to migrate another database, specify the URI explicitly using the **-m** flag:

```
# /opt/stream/sbin/stream-upgrade -t <target version> -m "<MongoDB connection string>"
```

NOTE

The upgrade script requires the mongo shell **MongoSH** to connect to your database (**mongosh**). If this client is not installed on the host where Stream is running, consider installing the standalone **mongosh** client or running the upgrade script from another host that has access to the database.

2.6. Uninstallation

WARNING

Before uninstalling, please ensure that you have a **proper backup of the Stream component**. Once uninstalled, all Stream data will be **irremediably lost**!

NOTE

Uninstalling Stream consists in uninstalling:

- The Stream service;
- The MongoDB service;
- The NGINX service.

2.6.1. Uninstalling Stream

Connect to the server with an account with administrative privileges;

Uninstall Stream with the following commands:

```
# systemctl stop stream
# yum remove stream
# rm -rf /opt/stream
# rm -rf /var/log/stream
# rm -f /etc/default/stream
```

2.6.2. Uninstalling NGINX

Connect to the server with an account with administrative privileges;

Uninstall NGINX with the following commands:

```
# systemctl stop nginx
# yum remove nginx
# rm -rf /etc/nginx
# rm -rf /var/log/nginx
```

2.6.3. Uninstalling MongoDB

Connect to the server with an account with administrative privileges;

Uninstall MongoDB with the following commands:

```
# systemctl stop mongod
# rpm -qa | grep -i mongo | xargs rpm -e
# rm -rf /var/log/mongodb
# rm -rf /var/lib/mongodb
```

3. Installing on Kubernetes

3.1. Installation

3.1.1. Concepts overview

In Kubernetes, applications are deployed onto **Pods**, which represents a running version of a containerized application. Pods are grouped by **Deployments**, which represent a set of Pods running the same application. For instance, should you need to run Stream in high availability mode, your deployment will contain 3 pods or more. Applications running in Pods are made accessible by a **Service**, which grants a set of Pods an IP address (which can either be internal to the cluster or accessible on the public Internet through a Load Balancer).

The recommended way of installing on Stream is through the Stream's Helm Chart. Helm is a package manager for Kubernetes that will generate Kubernetes resources necessary to deploy Stream onto your cluster. The official Helm Chart will generate a deployment of one or more Pods running Stream on your cluster.

3.1.2. Prerequisites

Before you start, make sure you have the following prerequisites:

- The `kubectl` command line tool installed and configured to access the destination cluster : installation.
- The `helm` command line tool installed and configured to access the destination cluster: installation.
- A working knowledge of Kubernetes and Helm. If you are new to Kubernetes, we recommend you read the [Kubernetes Basics](#) tutorial. If you are new to Helm, we recommend you read the [Helm Quickstart](#) tutorial.
- A cluster that can pull images from the EVERTRUST container registry. If this is not possible through Internet, see [Running behind a container registry proxy](#) for more information on how to set up a private registry mirror.
- A license file for your Stream installation. This file is usually named `stream.lic` and should have been provided to you by EVERTRUST.
- A set of credentials to access the EVERTRUST container repository. You should have received them from EVERTRUST.

3.1.3. Configuring the namespace

For isolation purposes, we strongly recommend that you create a dedicated namespace for **Stream**:

```
$ kubectl create namespace stream
```

The namespace should be empty. In order to run Stream, you'll need to create two secrets in that

namespace:

- A data secret containing your Stream license file and keyset.
- An image pull secret, allowing Kubernetes to authenticate to the EverTrust's container repository

Creating the application secrets

You should have both a license file (most probably named `stream.lic`) and a keyset for your Stream installation.

To generate a keyset, download our [keyset utility](#) onto a secure environment that has access to your cluster. Extract the archive and run the binary that matches your architecture. For instance :

```
$ ./tinkey-darwin-arm64 generate-keyset --out=keyset.json
```

Then, create a Kubernetes secret containing both files into the Stream namespace :

```
$ kubectl create secret generic stream-data \  
  --from-file=license="<path to your license file>" \  
  --from-file=keyset="<path to your keyset file>" \  
  --namespace stream
```

Creating the image pull secret

Next, you should configure Kubernetes to authenticate to the EverTrust repository using your credentials. They are necessary to pull the Stream docker image, you should have received them upon purchase. Get your username and password and create the secret:

```
$ kubectl create secret docker-registry evertrust-registry \  
  --docker-server=registry.evertrust.io \  
  --docker-username="<your username>" \  
  --docker-password="<your password>" \  
  --namespace stream
```

3.1.4. Setting up Helm repository

Now that the application secrets are configured, add the **EverTrust Helm repository** to your machine:

```
$ helm repo add evertrust https://repo.evertrust.io/repository/charts
```

Verify that you have access to the Chart :

```
$ helm search repo evertrust/stream
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
evertrust/stream	0.2.0	2.0.0	EverTrust Stream Helm chart

3.1.5. Configuring the chart

You'll next need to override the defaults `values.yaml` file of the Helm Chart to reference the secrets that we've created. We'll provide a minimal configuration for demonstration purposes, but please do follow our production setup guide before deploying for production.

Create a `override-values.yaml` file somewhere and paste this into the file:

```
image:
  pullSecrets:
    - evertrust-registry

license:
  secretName: stream-data
  secretKey: license

keyset:
  secretName: stream-data
  secretKey: keyset
```

To finish Stream's installation, simply run the following command:

```
$ helm install stream evertrust/stream -f override-values.yaml -n stream
```

Please allow a few minutes for the Stream instance to boot up. You are now ready to go on with the **first login**. This instance will allow you to test out if Stream is working correctly on your cluster. However, this installation is not production-ready. Follow our **production checklist** to make sure your instance is fit to run in your production environment.

3.2. First login

3.2.1. Fetching the default administrator password

Allow a few seconds for your Stream instance to boot up. You can then fetch the administrator password that has been generated for your instance using the command :

```
$ kubectl exec $(kubectl get pods -n stream -l "app.kubernetes.io/name=stream" --sort -by={.status.podIP} -o jsonpath="{.items[0].metadata.name}") -n <namespace> -- /bin/sh -c "cat /stream/adminPassword"
```

The default administrator credentials are:

NOTE

- Login: **administrator**
- Password: the one you got from the command above

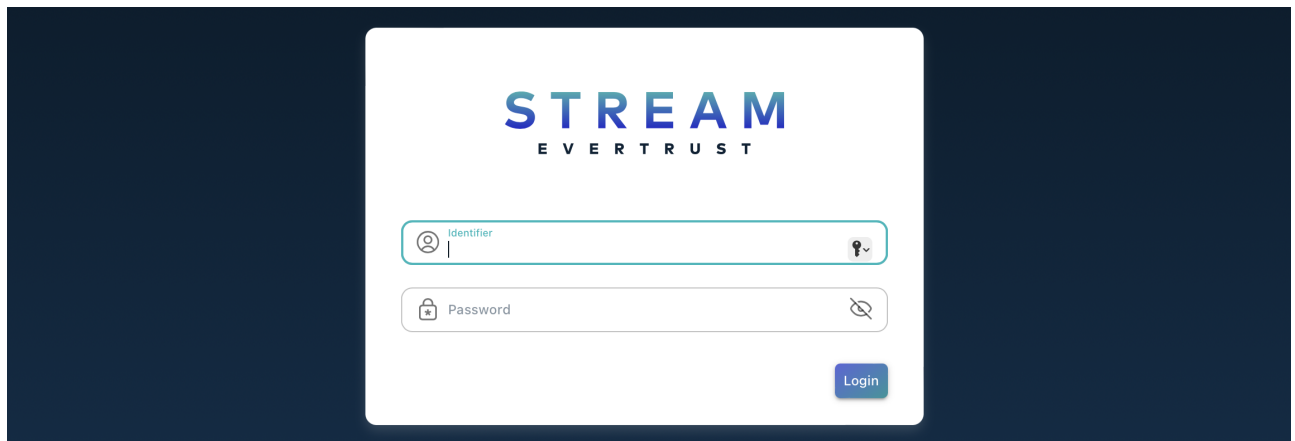
3.2.2. Manually creating the initial user

In case the automatic bootstrap process was disabled, you may need to manually create an administrator user. Launch a MongoDB shell to access your database and run the following command to create the initial administrator:

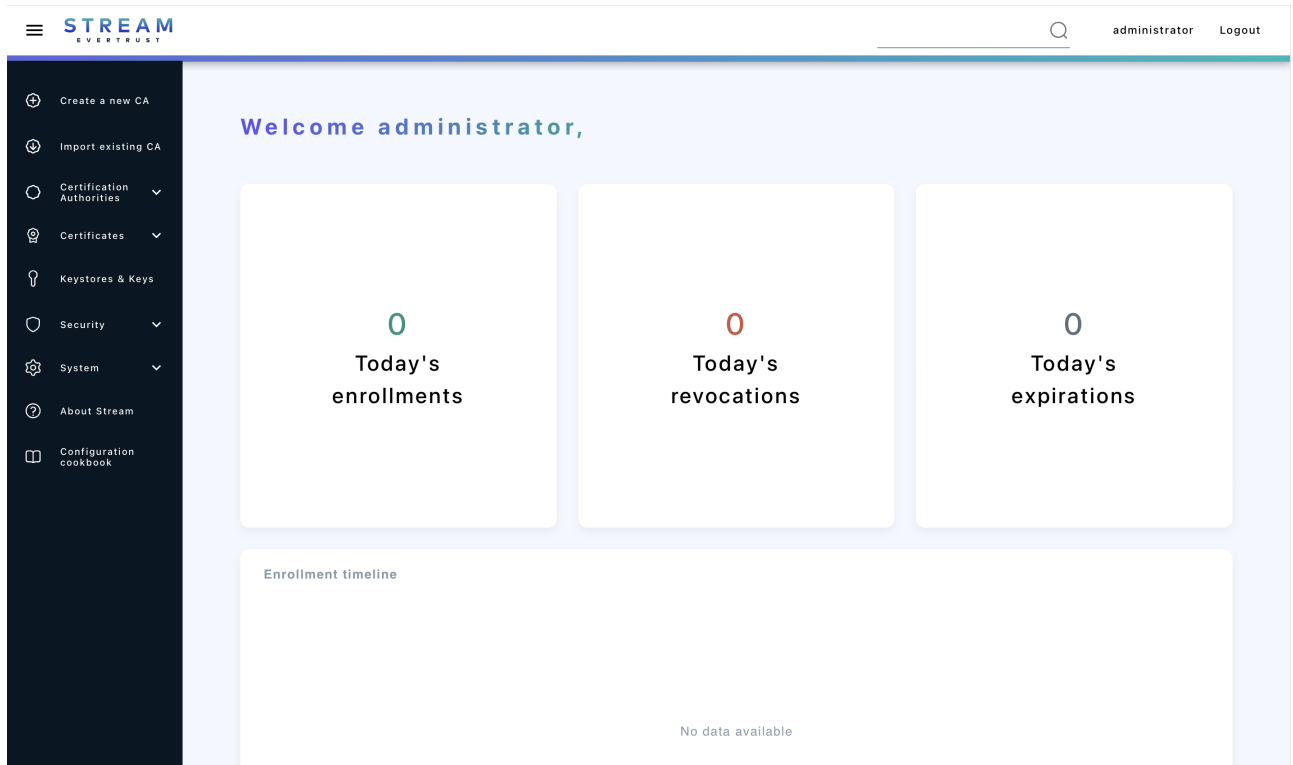
```
db.security_accounts.insertOne({"identifier":"administrator","secret":"$6$96ZV/UmX1oMP  
UVA3$U5MejjbJ9S3jhqq1TDqhZMwV0cDX5BAWY3DL2nsxUHLpHj0LOfPuswy4nWjkMLify4FvKGKhEfADzljy7  
FGc8.", "permissions":[{"value":"configuration:*"}, {"value":"lifecycle:*"}], "roles":[],  
"type":"local"})
```

3.2.3. Accessing the Stream Web Interface

1. Launch a web browser;
2. Browse to **https://[IP or DNS Name of the Stream component]/ui#**:



3. Specify the default administration credentials and hit the '**Login**' button:



CAUTION

It is **highly recommended** to delete the adminPassword file from your machine once you saved it somewhere safe.

3.3. Production checklist

Even though the Helm Chart makes installing Stream a breeze, you'll still have to set up a few things to make Stream resilient enough to operate in a production environment.

3.3.1. Operating the database

All persistent data used by Stream is stored in the underlying MongoDB database. Therefore, the database should be operated securely and backed up regularly.

When installing the chart, you face multiple options regarding your database:

- By default, a local MongoDB standalone instance will be spawned in your cluster, using the `bitnami/mongodb` chart. No additional configuration is required but it is not production ready out of the box. You can configure the chart as you would normally below the `mongodb` key :

```
mongodb:
  architecture: replicaset
  # Any other YAML value from the chart docs
```

- If you want to use an existing MongoDB instance, provide the `externalDatabase.uri` value. The URI should be treated as a secret as it must include credentials:

```
externalDatabase:
```

```
secretName: <secret name>
secretKey: <secret key>
```

The chart doesn't manage the database. You are still in charge of making sure that the database is correctly backed up. You could either back up manually using `mongodump` or use a managed service such as MongoDB Atlas, which will take care of the backups for you.

3.3.2. Managing secrets

Storing secrets is a crucial part of your Stream installation. The keyset is the most important of them, being a master key used to encrypt and decrypt data before they enter the database. Alongside with other application secrets like your MongoDB URI (containing your credentials or certificate). We recommend that you create Kubernetes secrets beforehand or inject them directly into the pod.

Values that should be treated as secrets in this chart are:

Name	Description	Impact on loss
<code>keyset</code>	Master key used to encrypt sensitive data in database.	Highest impact: database would be unusable
<code>events.secret</code>	Secret used to sign and chain events.	Moderate impact: events integrity would be unverifiable
<code>externalDatabase.uri</code>	External database URI, containing a username and password.	Low impact: reset the MongoDB password
<code>appSecret</code>	Application secret use to encrypt session data.	Low impact: sessions would be reset
<code>mailer.password</code>	SMTP server password	Low impact: reset the SMTP password

For each of these values, either :

- leave the field empty, so that a secret will be automatically generated.
- derive the secret value from an existing Kubernetes secret:

```
appSecret:
  secretName: <secret name>
  secretKey: <secret key>
```

WARNING

Always store secrets in a safe place after they're generated. If you ever uninstall your Helm chart, the loss of the keyset will lead to the impossibility of recovering most of your data.

3.3.3. High availability

By default, the chart will configure a single-pod deployment. This deployment method is fine for testing but not ready for production as a single failure could take down the entire application. Instead, we recommend that you set up a Stream cluster using at least 3 pods.

In order to do that, configure an `horizontalAutoscaler` in your `override-values.yaml` file:

```
horizontalAutoscaler:
  enabled: true
  minReplicas: 3
  maxReplicas: 3
```

NOTE

Use `nodeAffinity` to spread your Stream cluster Pods among multiple nodes in different availability zones to reduce the risk of Single Point of Failure.

3.3.4. Configuring ingresses

To create an ingress upon installation, simply set the following keys in your `override-values.yaml` file:

```
ingress:
  enabled: true
  hostname: stream.lab
  tls: true
```

3.4. Upgrade

We recommended that you only change values you need to customize in your `values.yaml` file to ensure smooth upgrading. Always check the upgrading instructions between chart versions.

3.4.1. Upgrading the chart

When upgrading Stream, you'll need to pull the latest version of the chart :

```
$ helm repo update evertrust
```

Verify that you now have the latest version of Stream (through the App version column) :

```
$ helm search repo evertrust/stream
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
evertrust/stream	0.2.0	2.0.0	EverTrust Stream Helm chart

Launch an upgrade by specifying the new version of the chart through the `--version` flag in your

command :

```
$ helm upgrade stream evertrust/stream \
  --values override-values.yaml \
  --version 0.2.0
```

The chart will automatically create a **Job** that runs an upgrade script when it detects that the Stream version has changed between two releases. If the upgrade job fails to run, check the job's pod logs. When upgrading from an old version of Stream, you may need to explicitly specify the version you're upgrading from using the **upgrade.from** key.

WARNING

Before upgrading to specific chart version, thoroughly read any **Specific chart upgrade instructions** for your version.

3.4.2. Specific chart upgrade instructions

Empty section.

3.5. Uninstallation

To uninstall Stream from your cluster, simply run :

```
$ helm uninstall stream -n stream
```

This will uninstall Stream. If you installed a local MongoDB instance through the Stream's chart, it will also be uninstalled, meaning you'll lose all data from the instance.

WARNING

Before uninstalling Stream, if you wish to keep your database, please back up your application secrets (in particular the keyset). Without it, you won't be able to decrypt your database and it will become useless.

4. Running with Docker/Compose

If you just want to try out Stream, one way of doing so could be to directly run Stream from Docker. For resiliency reasons, this is obviously not recommended for production usage.

We provide a Docker image that's entirely configurable through environment variables. All Docker examples require that you login to our Docker repository beforehand :

```
$ docker login registry.evertrust.io
```

NOTE

If you're looking to try out Stream's features, take a look at the [EVERTRUST Playground](#). It is a Docker Compose project bundled with demo values to get you started swiftly.

4.1. Docker Compose example

The simplest way to spin up an Stream instance is to let Docker Compose manage the required components :

- the database,
- the Stream instance
- and (optionally) the reverse proxy.

NOTE

The license and keyset should be correctly encoded for consumption as a YAML environment variable:

- the `\n` characters in the license should be removed, for instance by `cat stream.lic | tr -d '\n'`
- the `"` characters in the keyset should be escaped, for instance by `cat keyset.json | jq -R`

Copy the following `docker-compose.yaml` file and tweak it to match your needs :

```
version: "3.1"
services:
  stream:
    image: registry.evertrust.io/stream:2.0.x
    ports:
      - "9443:9443"
    networks:
      - stream
    environment:
      LICENSE: MI...
      APPLICATION_SECRET: tobechanged
      EVENT_SEAL_SECRET: tobechanged
```



```

KEYSET: "{\"primaryKeyId\": \"...\"}"
HOSTS_ALLOWED.0: .
MONGODB_URI: mongodb://mongo:27017/stream
depends_on:
  - mongo
healthcheck:
  test: [ "CMD", "curl", "-f", "http://localhost:7626/ready" ]
  interval: 10s
  timeout: 60s
  retries: 10
mongo:
  image: mongo:7
  restart: always
  volumes:
    - database:/data/db
  networks:
    - stream
volumes:
  database: {}
networks:
  stream: {}

```

You then only need to run the following in the directory where you created the previous file :

```
$ docker compose up
```

Stream should quickly become available on <https://localhost:9443>.

TIP A self-signed certificate will be generated at each reboot of Stream

4.2. Vanilla Docker example

Pull the latest Stream image:

```
$ docker pull registry.evertrust.io/stream:{page-version}.x
```

The Stream Docker image ships with sensible configuration defaults. Most can be configured by injecting environment variables when running the container, like so:

```

$ docker run \
  -e LICENSE="MI..." -e APPLICATION_SECRET="tobechanged" -e
  EVENT_SEAL_SECRET="tobechanged" -e KEYSET="{\"primaryKeyId\": \"...\"}" -e
  HOSTS_ALLOWED.0="." -e MONGODB_URI="" -p 9443:9443 \
  registry.evertrust.io/stream:{page-version}.x

```

4.3. Injecting extra configuration

The Docker image comes with a simple enough configuration to get started and test the software. However, it doesn't include any way to cluster the software with other instances or to edit other specific configurations. If you need to do so, you can mount custom configuration files, giving you full control over how Stream behaves.

The mounted folder :

- MUST contain an **pekko.conf** file configuring the Pekko cluster. See the [reference config](#) to get an idea over what's configurable.
- CAN contain a **application.conf** file containing any extra config options unrelated to clustering.

A typical Docker command would then be :

```
$ docker run \  
  -v [configurationPath]:/opt/stream/etc/:rw \  
  ...  
  registry.evertrust.io/stream:2.0.x
```

4.4. Custom startup scripts

Sometimes, you'll want to run scripts each time the container starts up in order to configure files in the container or set environment variables. To do so, you'll need to mount shell scripts into the **/docker-entrypoint.d/** directory in the container :

```
$ docker run \ -v [scriptsPath]:/docker-entrypoint.d/ \  
  ...  
  registry.evertrust.io/stream:2.0.x
```

Where **scriptsPath** is a directory containing one or multiple shell scripts that will be sourced before running Stream.

5. Troubleshooting

5.1. Stream Doctor

Stream doctor is a tool that performs checks on your Stream installation as well as its dependencies to ensure that everything is configured properly. Note that the tool requires root permissions to run.

5.1.1. Checks performed

At the moment, Stream doctor checks for :

OS checks

- Checks for installed Stream version, MongoDB version, Java version, Nginx Version and OS version.
 - If the OS is a RedHat distribution, checks for RedHat subscription
 - If Mongo is not installed locally, it notices it as an information log
- Checks for **SELinux**'s configuration
 - If selinux is disabled nothing has to be checked
 - If selinux is enforced checks the **httpd_can_network_connect** sebool value
- Checks for the status of the necessary services: **mongod**, **nginx** and **stream**.
- Checks how long the **stream service** has been running for.
- Checks if there is an **NTP service** active on the machine and checks if the system clock is synchronized with the NTP service.

Config checks

- Checks for existence and permissions of the **configuration** file: the permissions are expected to be at least 640 and the file is supposed to belong to stream:stream.
- Checks for existence and permissions of the **licence** file: the permissions are expected to be at least 640 and the file is supposed to belong to stream:stream.
- Checks for existence and permissions of the **keyset** file: the permissions are expected to be exactly 600 and the file is supposed to belong to stream:stream.
- Checks for existence and permissions of the Stream directory (default : /opt/stream) : the permission is expected to be at least 755
- Checks for the existence of the **symbolic link** for **nginx configuration** and runs an **nginx -t** test.
- Retrieves the **Java heap size parameters** that were set for Stream and informs the user if the default ones are used (min = 2048 and max = 3072).
- Retrieves the **Stream DNS hostname** and raises an error if it has not been set.

- Retrieves the **MongoDB URI** (throws a warning if MongoDB is running on localhost; throws an error if MongoDB is running on an external instance but the *authSource=admin* parameter is missing from the URI).
- Parses the **licence file** to retrieve its expiration date.
- Checks for the existence of the file containing the initial administrator password and throws a warning if that file still exists (displays the password too)

Network checks

- Runs a **MongoDB ping** on the URI, then checks for the database used in the URI (throws a warning if the database used is not called *stream*; throws an error if no database is specified in the URI).
- Checks for **AKKA High Availability** settings: if no node hostname is set up, skips the remaining HA checks. If 2 nodes are set up, retrieves which node is running the doctor and checks for the other node. If 3 nodes are set up, retrieves which node is running the doctor and checks for the other 2 nodes. The check runs as:
 - if *curl* is installed, runs a *curl* request on the Node hostname at *alive* on the management port (default is 8558), and if alive runs another *curl* request on the Node hostname at */ready* on the management port. Both requests should return HTTP/200 if ok, 000 otherwise.
 - if *curl* is not installed, uses the built-in Linux TCP socket to run TCP SYN checks on both the HA communication port (default is 25520) and the management port (default is 8558) on the Node hostname.
- Checks for **firewall configuration**. Currently only supports *firewalld* (RHEL) and a netstat test.
 - The **netstat part** will run a *netstat* command to check if the JVM listening socket is active (listening on port 9000). If *netstat* is not installed, it will skip this test.
 - The **firewalld part** will check if the HTTP and HTTPS services are opened in the firewall and if it detected a HA configuration, it will check if the HA ports (both of them) are allowed through the firewalld. If *firewalld* is not installed or not active, it will skip this test.
- Checks if IPv6 is active on each network interface and raises a warning if it is the case (with the interface name).

TLS checks

- Checks for existence and permissions of the **Stream server certificate** file: the permissions are expected to be at least 640 and the file is supposed to belong to the nginx group.
- Parses the **Stream server certificate** file: it should be constituted of the actual TLS server certificate first, then of every certificate of the trust chain (order being leaf to root). It throws a warning if the certificate is self-signed or raises an error if the trust chain has not been imported. It otherwise tries to reconstitute the certificate trust chain via the *openssl verify* command, and throws an error if it cannot.
- Parses the **Stream server certificate** file and checks if the **Stream hostname** is present in the **SAN DNS names** of the certificate, throws an error if it is not there.

5.1.2. Log packing option

If the Stream doctor is launched with the *-l option*, it will pack the logs of the last 7 days (in */opt/stream/var/log*) as well as the startup logs (the */var/log/stream/stream.log* file) and create a tar archive.

The *-l option* accepts an optional parameter that should be an integer (1-99) and will pack the logs of the last n days instead, as well as the startup logs.

Note that the **Stream doctor** will still perform all of its check; the log packing is done at the very end of the program.

Example of call to pack the logs of the last 7 days :

```
# /opt/stream/sbin/stream-doctor -l
```

Example of call to pack the logs of the last 30 days :

```
# /opt/stream/sbin/stream-doctor -l 30
```

5.1.3. Saving the doctor's output

If the Stream doctor is launched with the *-o option*, it will perform all of its checks and save the output in the specified file instead of displaying it into the stdout (default is the commandline interface).

If you use the option, you must provide a filepath in a writable directory.

Example of call to save the output in a file named *stream-doctor.out* instead of the stdout :

```
# /opt/stream/sbin/stream-doctor -o stream-doctor.out
```

5.1.4. Direct fixes

The Stream doctor is able to fix the following issues directly by itself if you use the *--fix* flag with the script:

- If the application secrets (play secret and event seal secret) have not been changed, the doctor will generate random application secrets and provide them to Stream directly (requires you to manually restart Stream afterwards);
- If firewalld is not allowing HTTP and HTTPS traffic, the doctor will change the firewall settings to allow **both** protocols and then restart the firewall by itself;
- If some permissions for the configuration file, the license file or the keyset file are not what they should be, the doctor will change these permissions (file owner and rwx permissions) to be what they should.

5.1.5. Help menu

To display Stream doctor's help menu, use the *-h option*.