EVERTRUST

EverTrust Horizon documentation v2.2

Installation Guide

EVERTRUST

Table of Contents

1. Introduction

1.1. Description

Horizon is EverTrust Certificate lifecycle management solution. This document is an installation procedure detailing how to install and bootstrap Horizon server on your infrastructure. It does not describe how to configure and operate a Horizon 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:

- A package-based installation on a server running **CentOS/RHEL 7.x/8.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 Horizon 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 Horizon.

2.1.1. System pre-requisites

The following elements are considered as system pre-requisites:

- A server running EL [7.x-8.x] x64 (CentOS / RHEL) with the network configured and **SELinux** disabled;
- Base and EPEL CentOS / RHEL [7.x-8.x] x64 repositories activated;
- An access with administrative privileges (root) to the server mentioned above;
- The IP address / DNS Name of an SMTP relay;
- The email address of the Horizon server administrator.

2.1.2. Software pre-requisites

The following elements are considered as software pre-requisites:

- The Horizon installation package: `horizon-2.2.X-1.noarch.rpm`;
- The MongoDB Community Edition package available from the MongoDB web site;
- EPEL repository activated.

NOTE

As a reminder, EPEL can be activated on CentOS / RHEL by doing the following:

```
$ yum install epel-release
```

2.2. Installation

2.2.1. Installing MongoDB

NOTE Mongo DB version 4.2.x to 5.0.x are supported by Horizon

Download the latest version of the following Mongo DB 5.x RPMs from the MongoDB web site:

- mongodb-org
- mongodb-org-mongos
- mongodb-org-server
- mongodb-org-shell

- mongodb-org-tools

Download the latest version of the Mongosh RPM from the mongosh github.

- mongodb-mongosh

Upload the downloaded RPMs through SCP on the server under `/root`.

Using an account with privileges, install the RPMs using 'yum'. For example, to install MongoDB version 5.0.1, run the following command from the folder containing the RPMs:

```
$ yum install mongodb-org*
$ yum install mongodb-mongosh
```

Enable the service at startup with the following command:

```
$ systemctl enable mongod
```

Start the mongod service with the following command:

```
$ systemctl start mongod
```

Verify that you can connect to the Mongo instance by running the mongo shell:

```
$ mongo
```

NOTE You can disconnect from the shell with ^D

2.2.2. Installing NGINX

1. Access the server through SSH 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 Horizon

Installation from the EverTrust repository

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

```
[horizon]
enabled=1
name=Horizon Repository
baseurl=https://repo.evertrust.io/repository/horizon-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 Horizon version:

```
$ yum install horizon
```

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

```
exclude=horizon
```

at the end of the `/etc/yum.repos.d/horizon.repo` file after installing Horizon.

Installing from RPM

Upload the file `horizon-2.2.X-1.noarch.rpm` through SCP under `/root`.

Access the server through SSH with an account with administrative privileges;

Install the Horizon package with the following command:

```
$ yum localinstall /root/horizon-2.2.X-1.noarch.rpm
```

NOTE

Installing the Horizon package will install the following dependencies:

- `dialog`
- `java-11-openjdk-headless`

> Please note that these packages may have their own dependencies.

2.2.4. Configuring the Firewall

Access the server through SSH with an account with administrative privileges;

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

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

Reload the firewall configuration with:

```
$ systemctl restart firewalld
```

2.3. Configuration

2.3.1. Initial Configuration

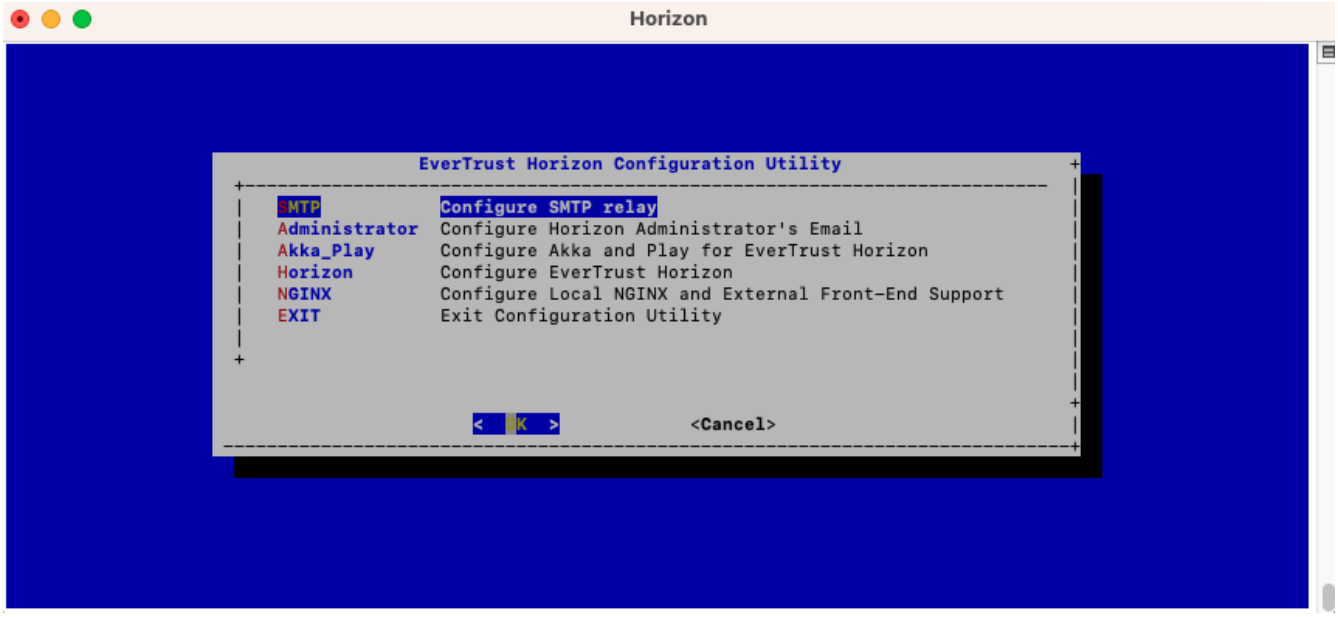
Configuring the SMTP Relay

Access the server through SSH with an account with administrative privileges;

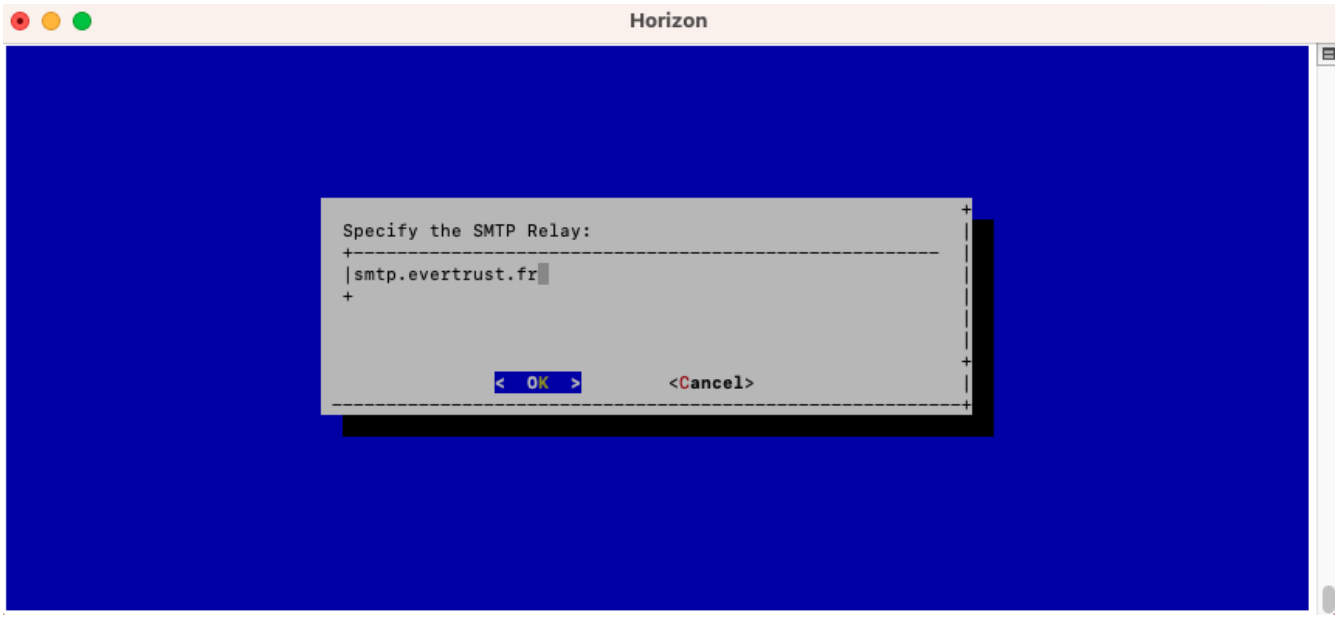
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

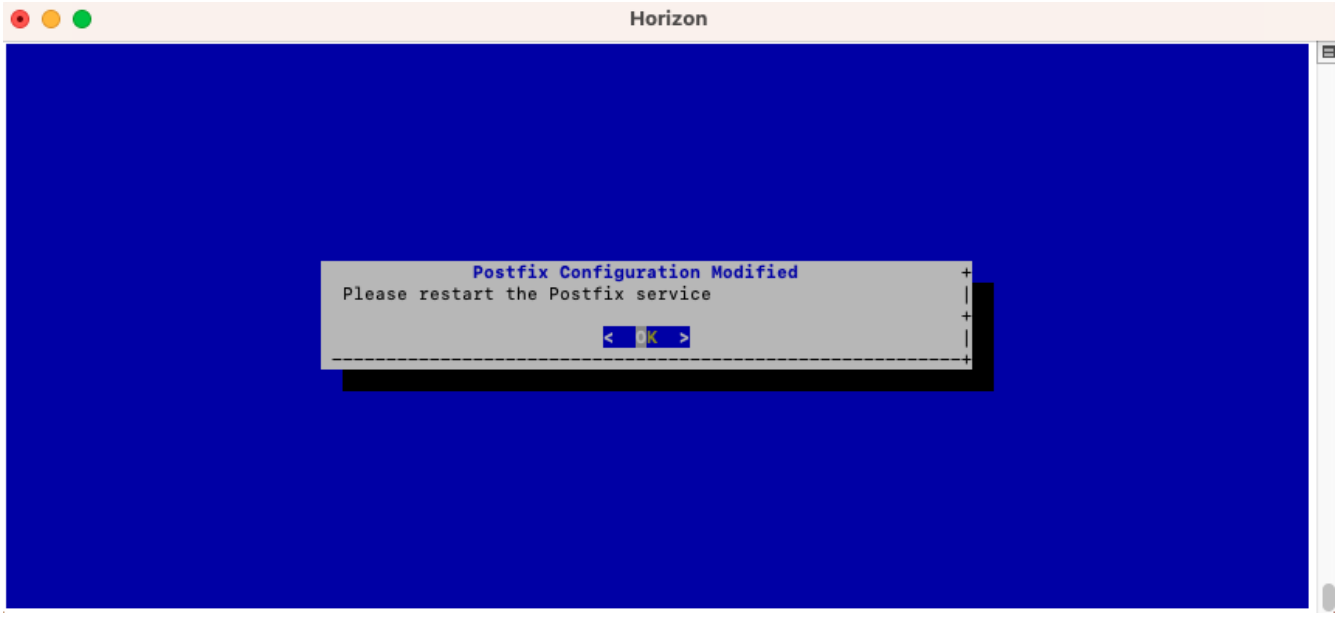
In the main menu, select '**SMTP**':



Specify IP address or the DNS name of the SMTP relay and validate:


Horizon
Specify the SMTP Relay:
smtp.evertrust.fr
< OK >
<Cancel>

The Postfix configuration is updated:



Exit the configuration utility and restart the Postfix service with the following command:

```
$ systemctl restart postfix
```

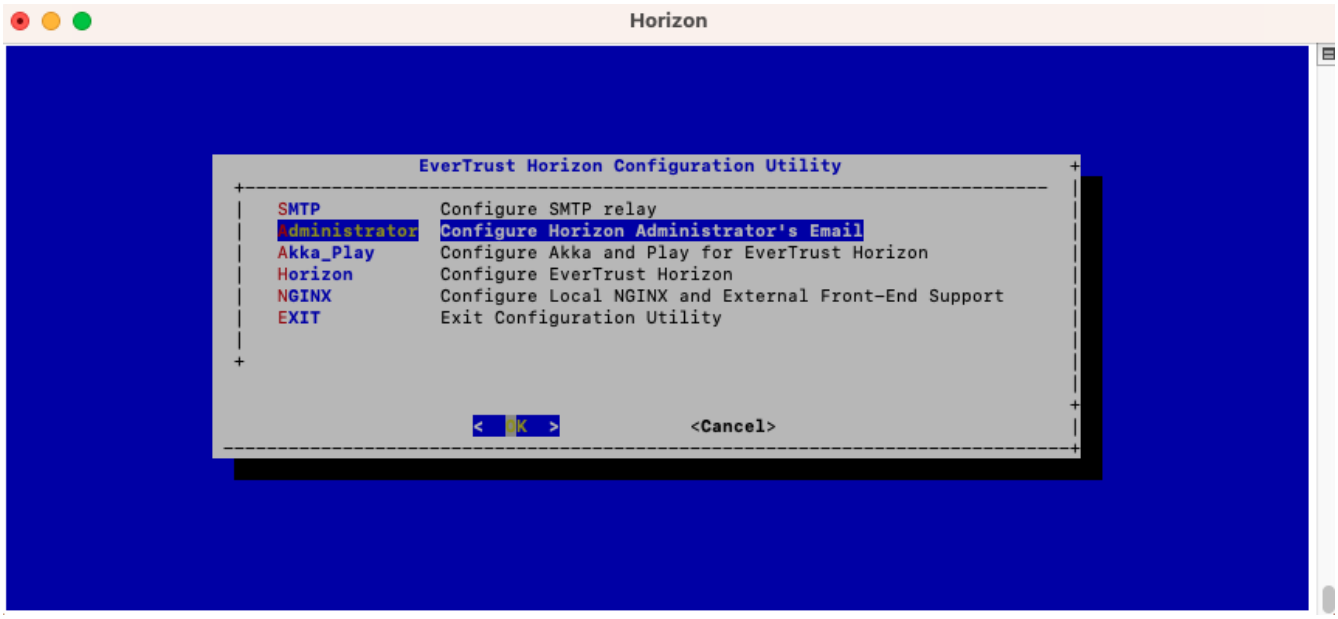
Configuring the Horizon Administrator's Email Address

Access the server through SSH with an account with administrative privileges;

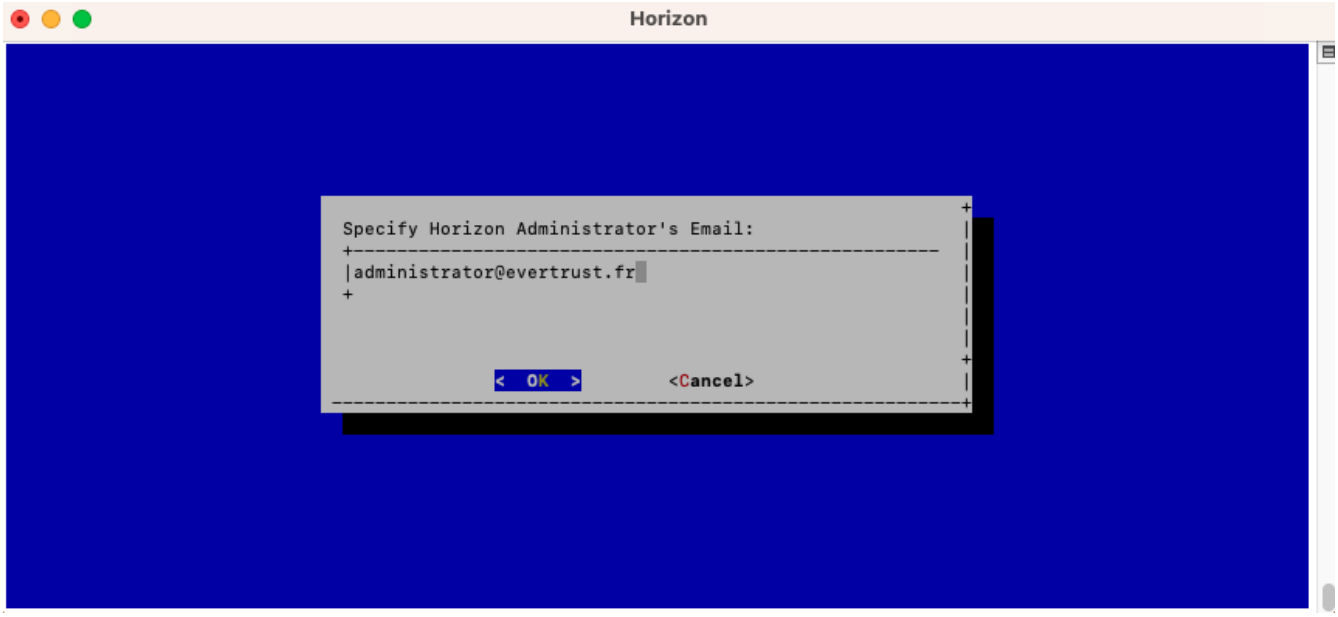
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

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



Specify the email address of the Horizon Administrator and validate:



Exit the Configuration Utility;

Validate the SMTP relay and Administrator Email Address with the following commands:

```
$ yum install mailx
$ mail -s "Hello Horizon root"
  > Hello From Horizon
  .
```

Ensure that the email receives the test email.

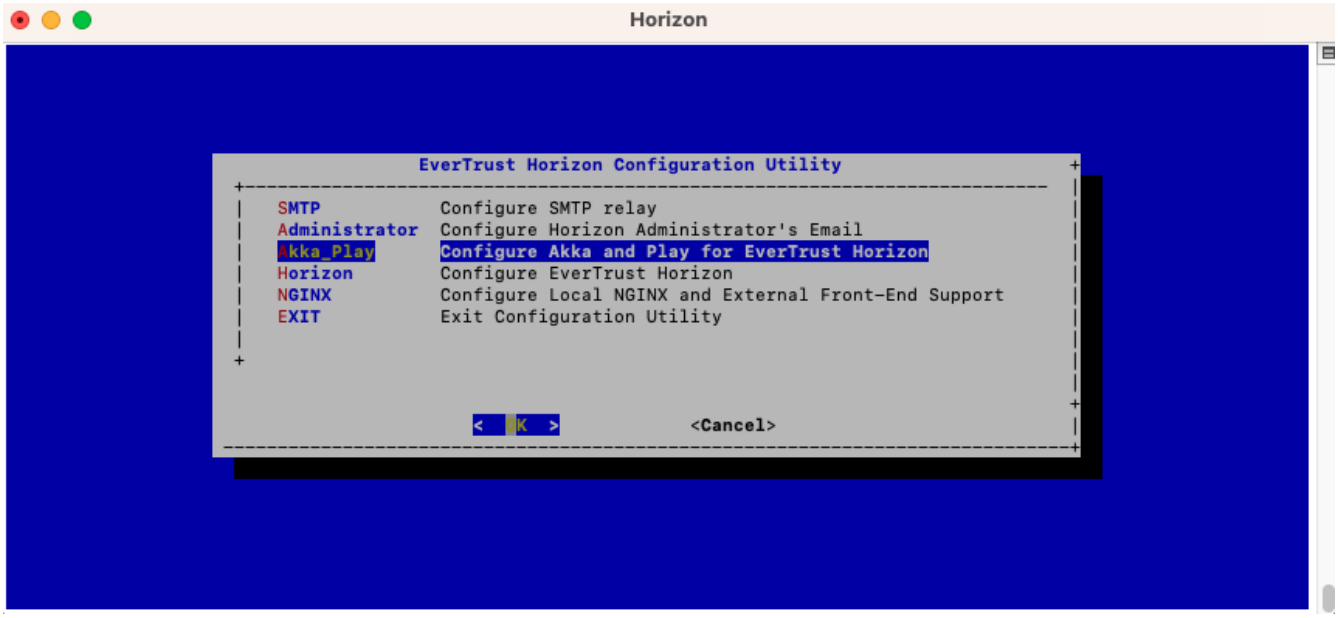
Generating a new Horizon Application Secret

Access the server through SSH with an account with administrative privileges;

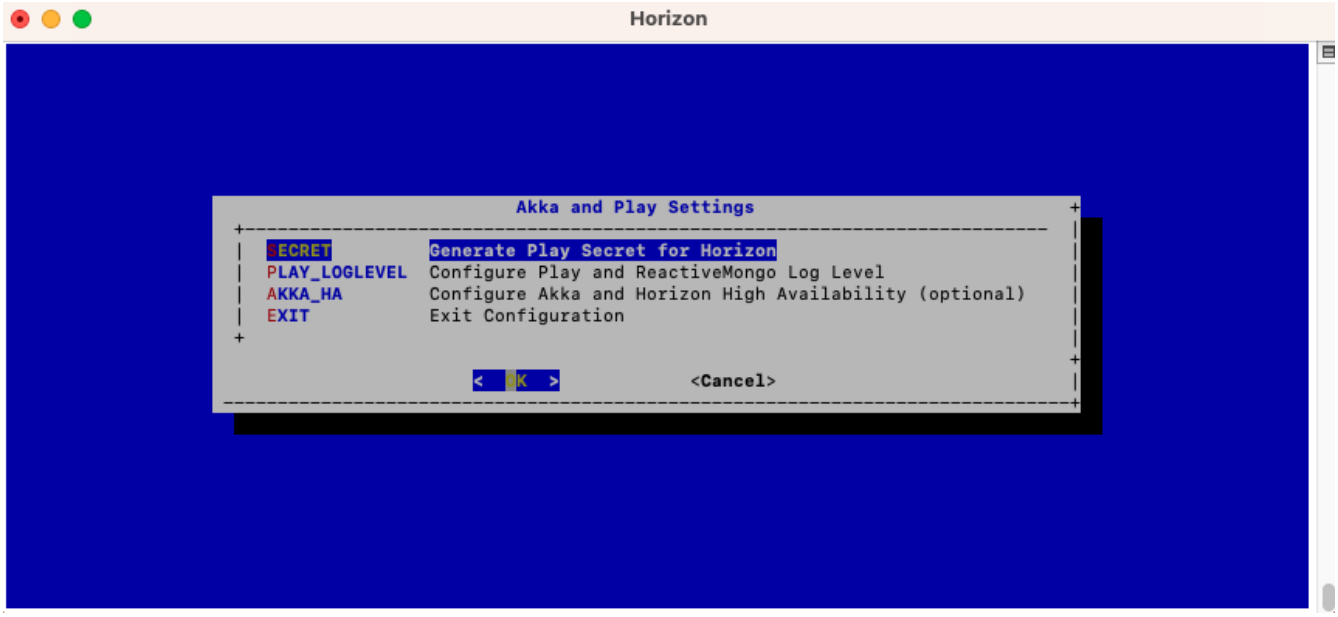
Run the Horizon Configuration Utility with the following command:

```
$/opt/horizon/sbin/horizon-config
```

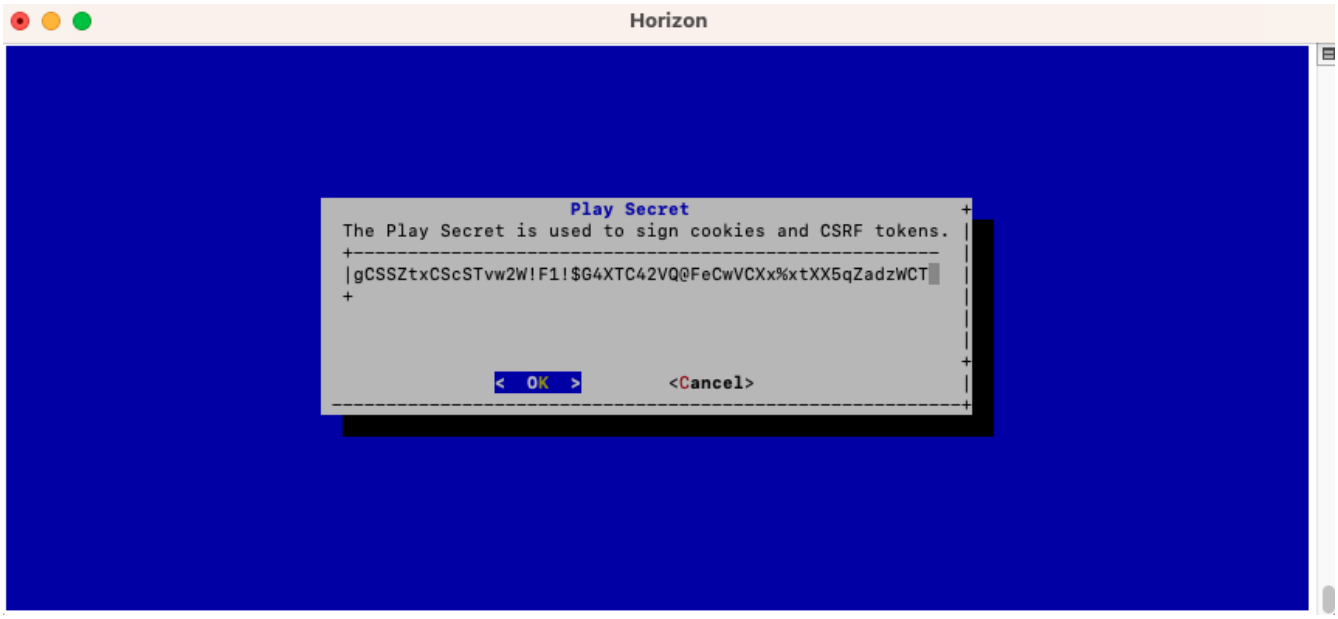
In the main menu, select '**Akka_Play**':



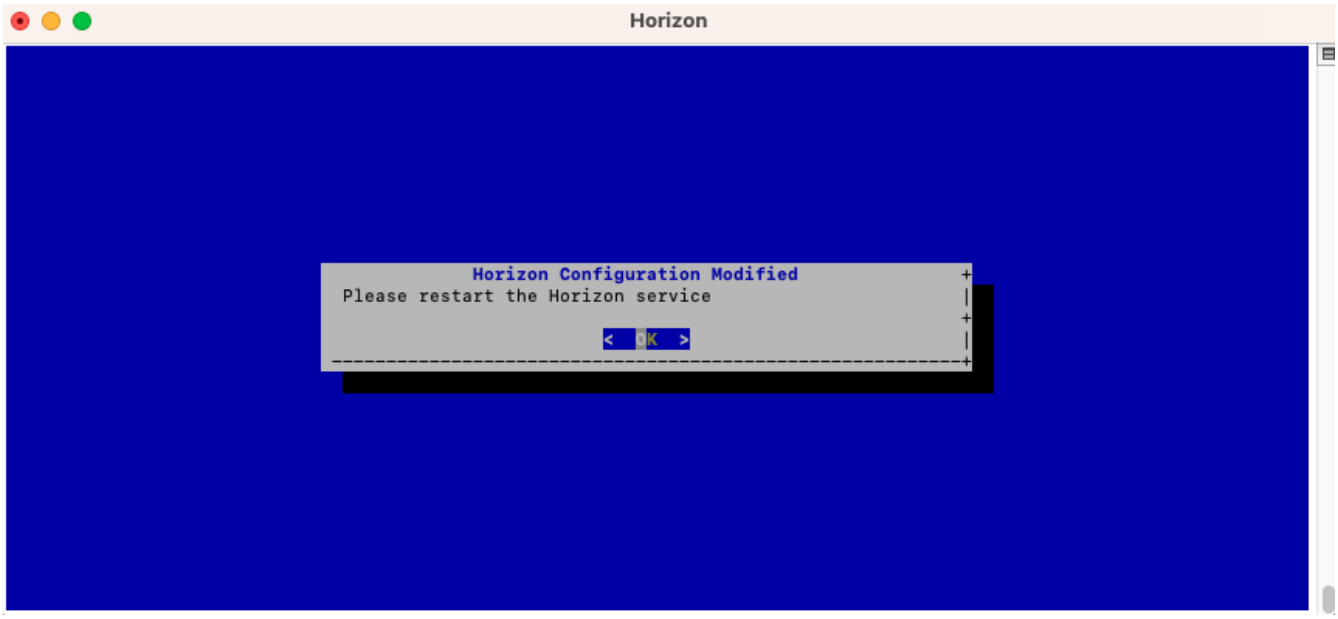
In the Akka_Play menu, select '**SECRET**':



Validate the new Horizon Application Secret:



The Horizon configuration is updated:


Horizon
Horizon Configuration Modified
Please restart the Horizon service
< OK >

JVM Configuration

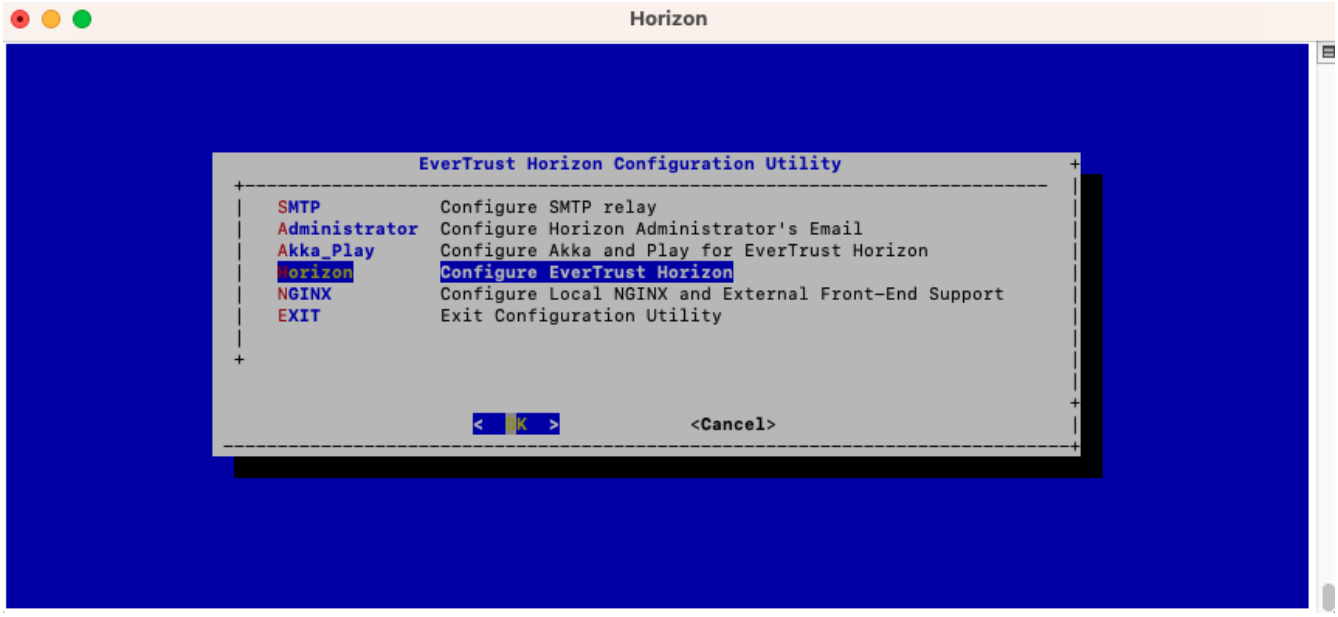
Horizon allows you to configure the ***xms*** (minimum memory allocation pool) and ***xmx*** (maximum memory allocation pool) parameters of the JVM running Horizon using the configuration tool.

Access the server through SSH with an account with administrative privileges;

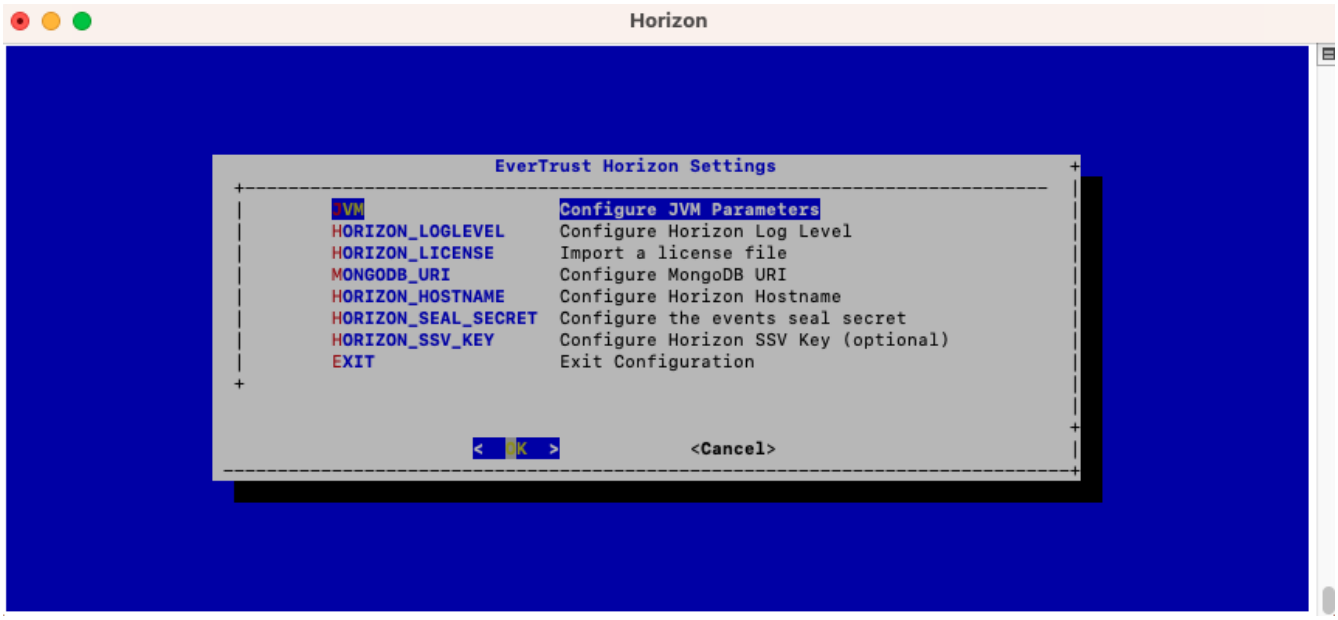
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

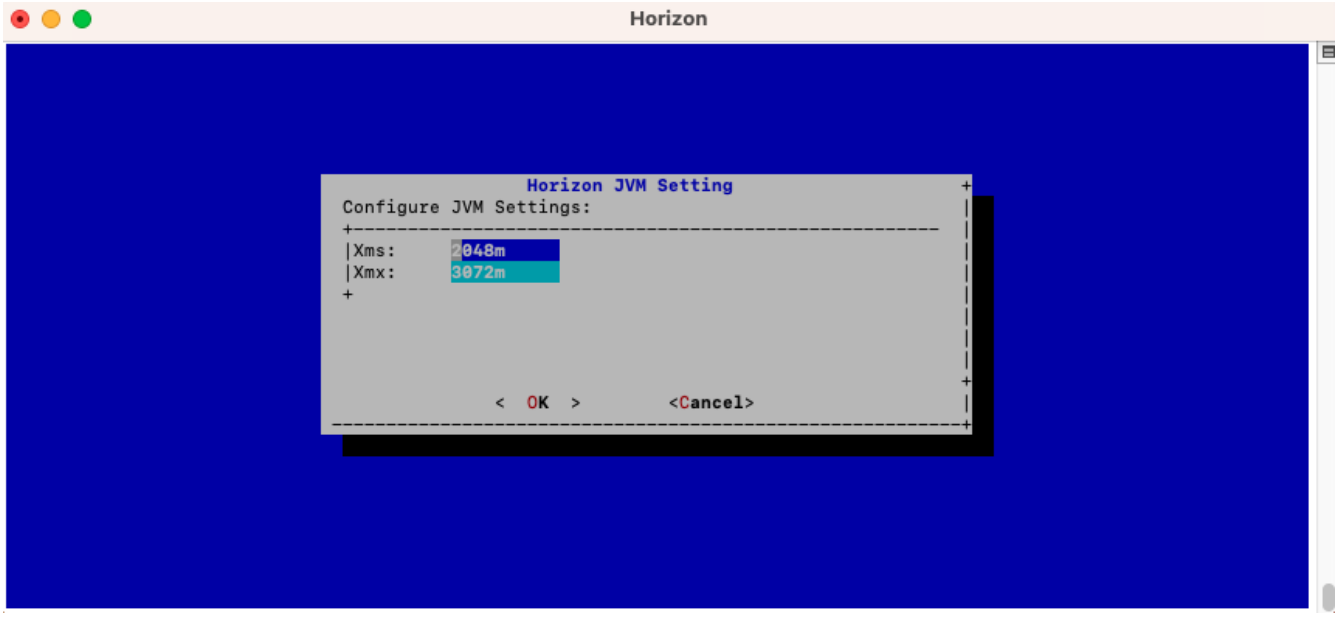
In the configuration menu, select '**Horizon**':



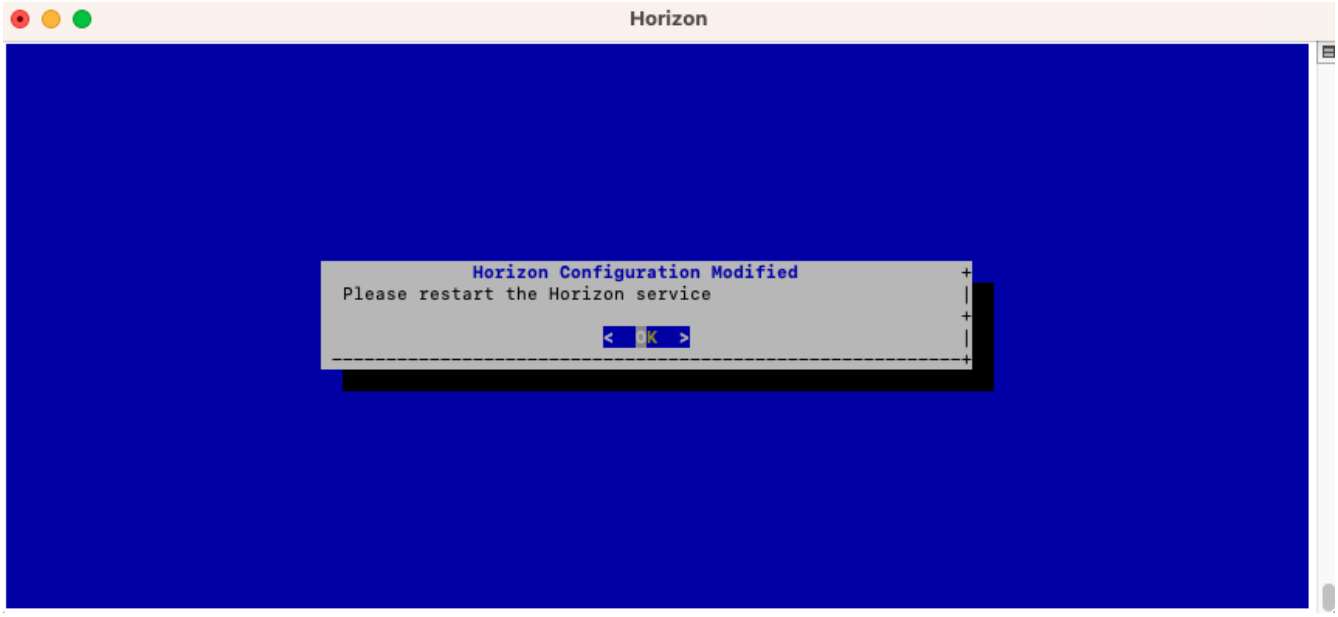
In the Horizon configuration menu, Select '**JVM**':



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



The new JVM parameters are configured:



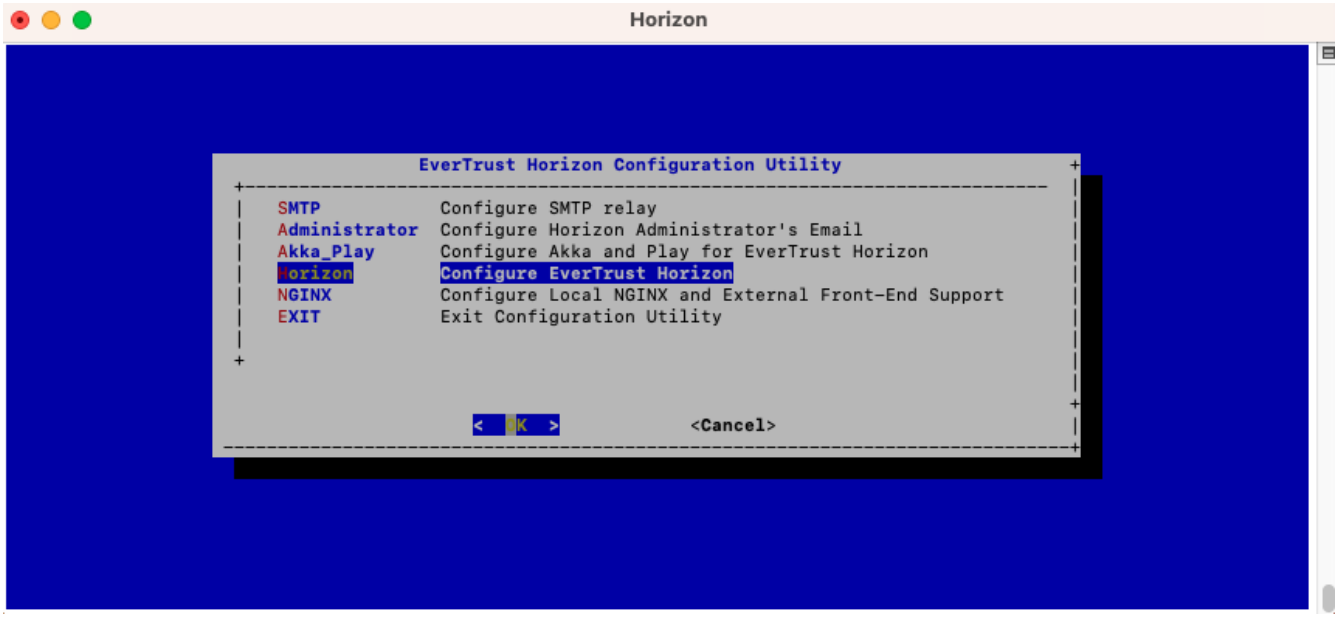
MongoDB URI Configuration

Access the server through SSH with an account with administrative privileges;

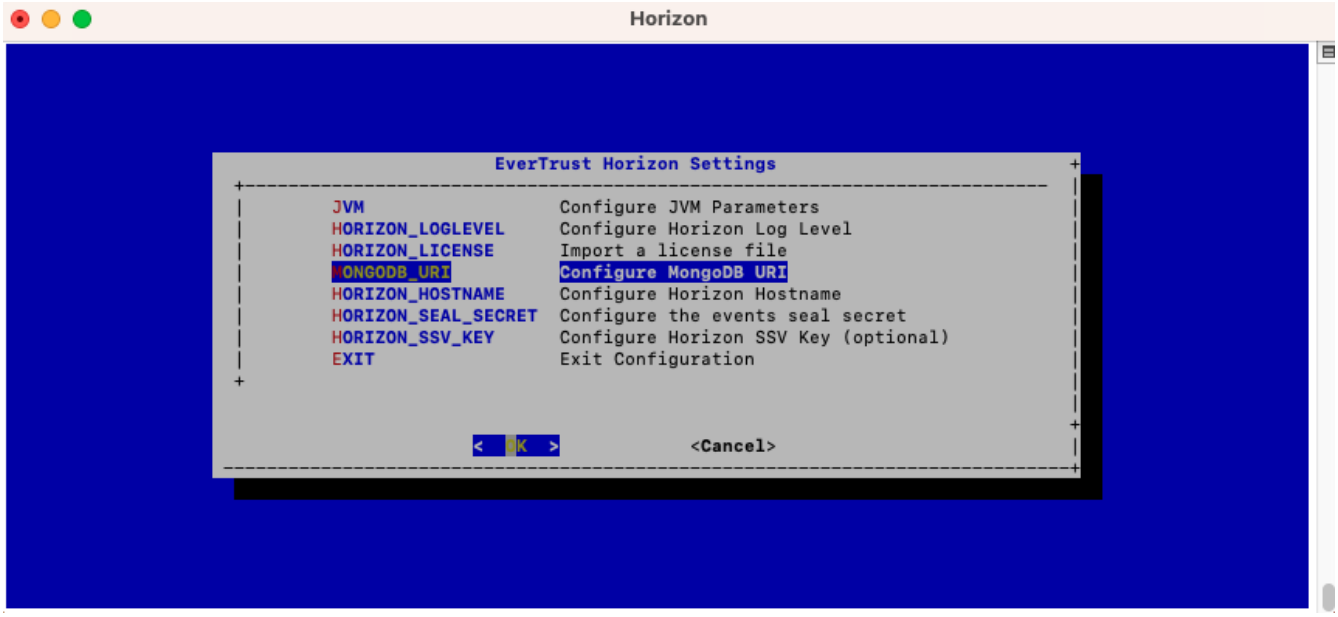
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

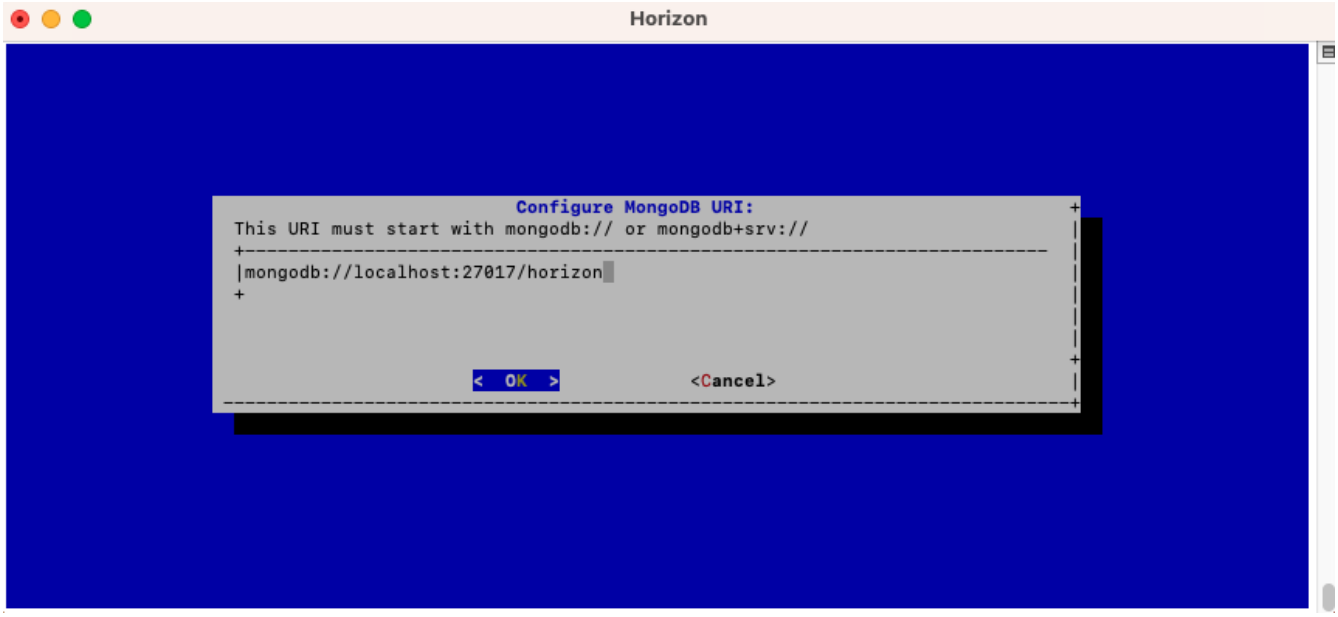
In the main menu, select **Horizon**:



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



Specify the MongoDB URI to target your MongoDB instance:



NOTE

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

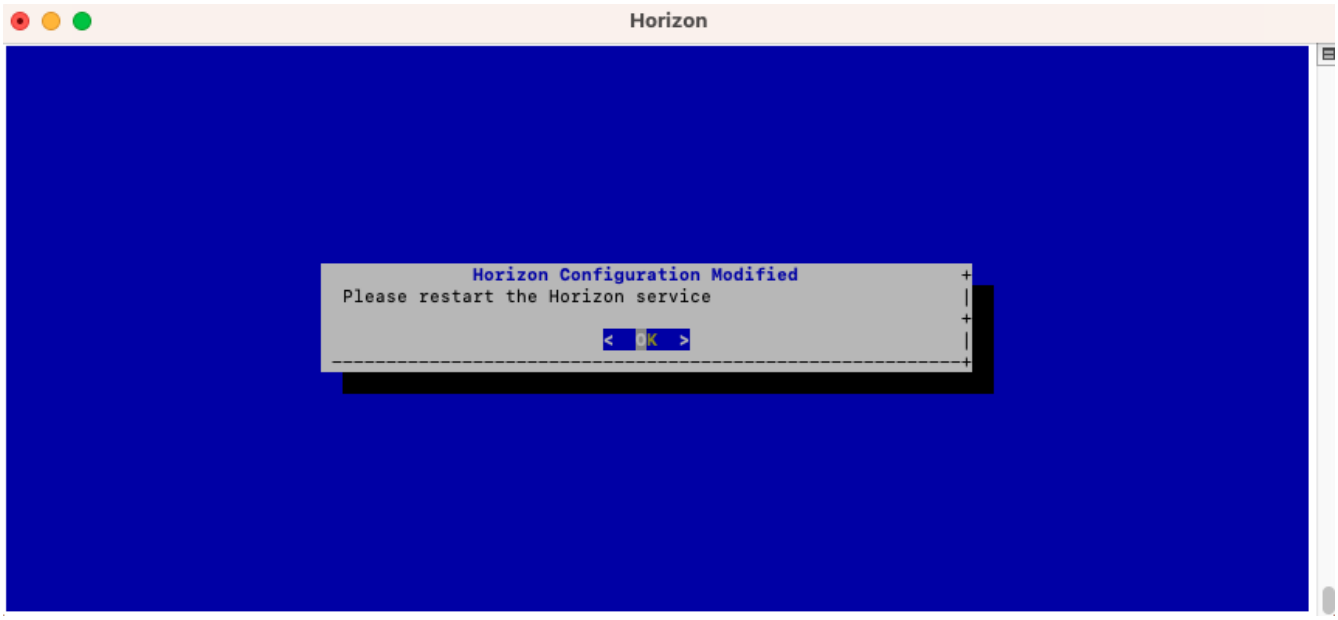
If you use an external MongoDB (such as MongoDB Atlas Database or dedicated On-premises database) instance:

- 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>@<Mongo-DB-hostname>:<Mongo-DB-Port>/horizon`

External MongoDB cluster of databases URI syntax: `mongodb+srv://<user>:<password>@<Mongo-DB-hostname-1>,<Mongo-DB-hostname-2>:<Mongo-DB-Port>/horizon?replicaSet=<Horizon-ReplicaSet-Name>&authSource=admin`

The MongoURI is configured:



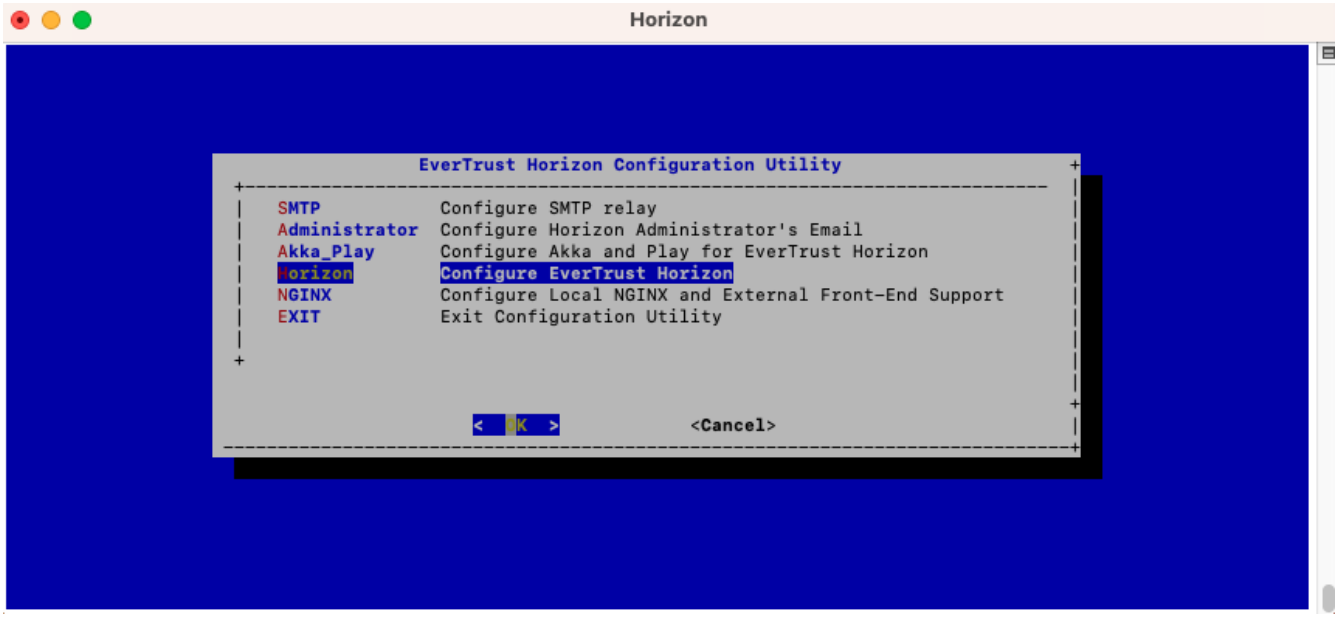
Horizon Hostname Configuration

Access the server through SSH with an account with administrative privileges;

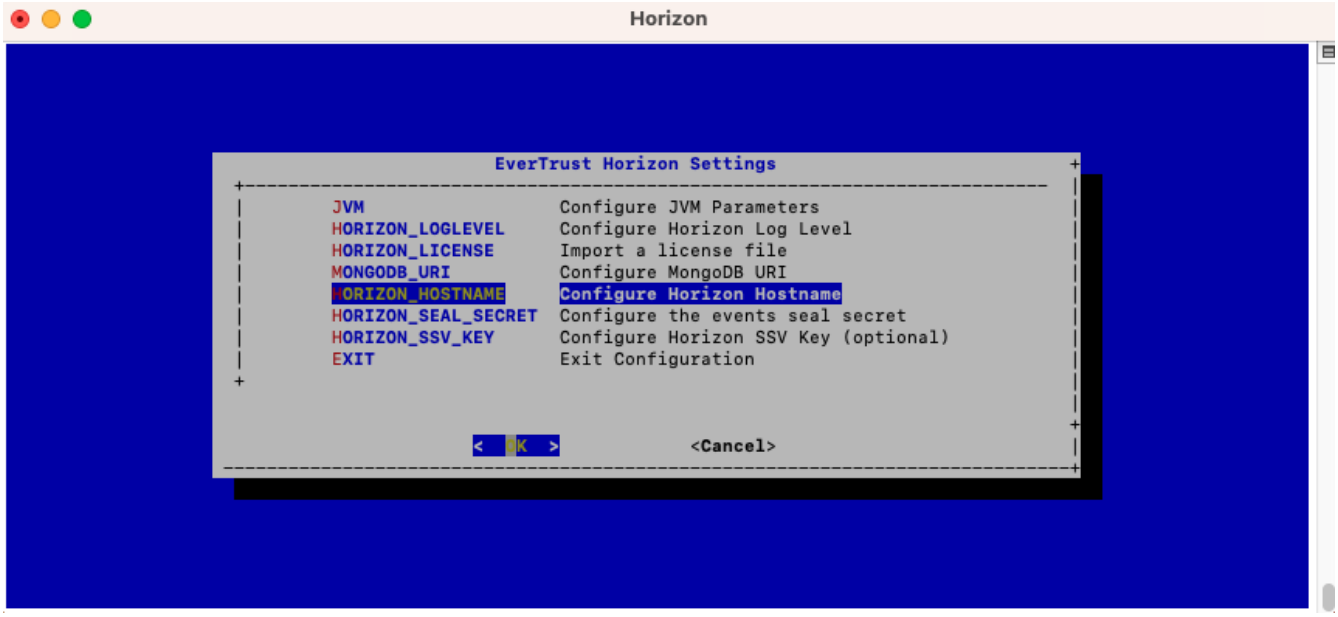
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

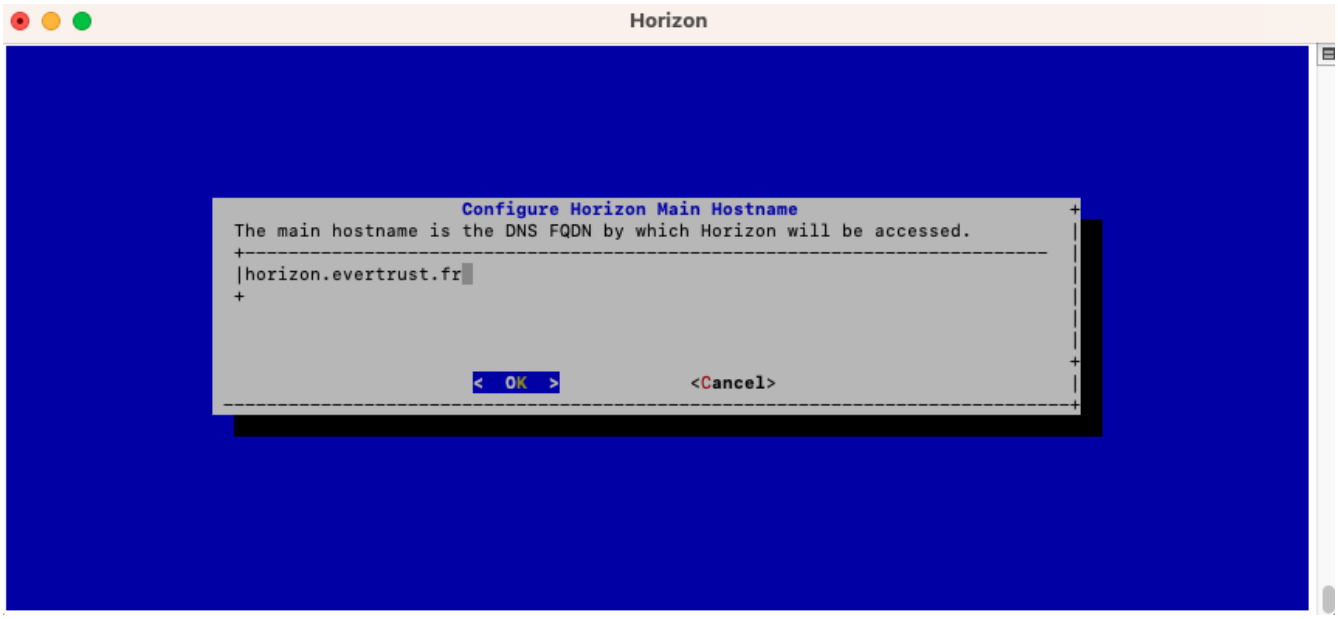
In the main menu, select '**Horizon**':



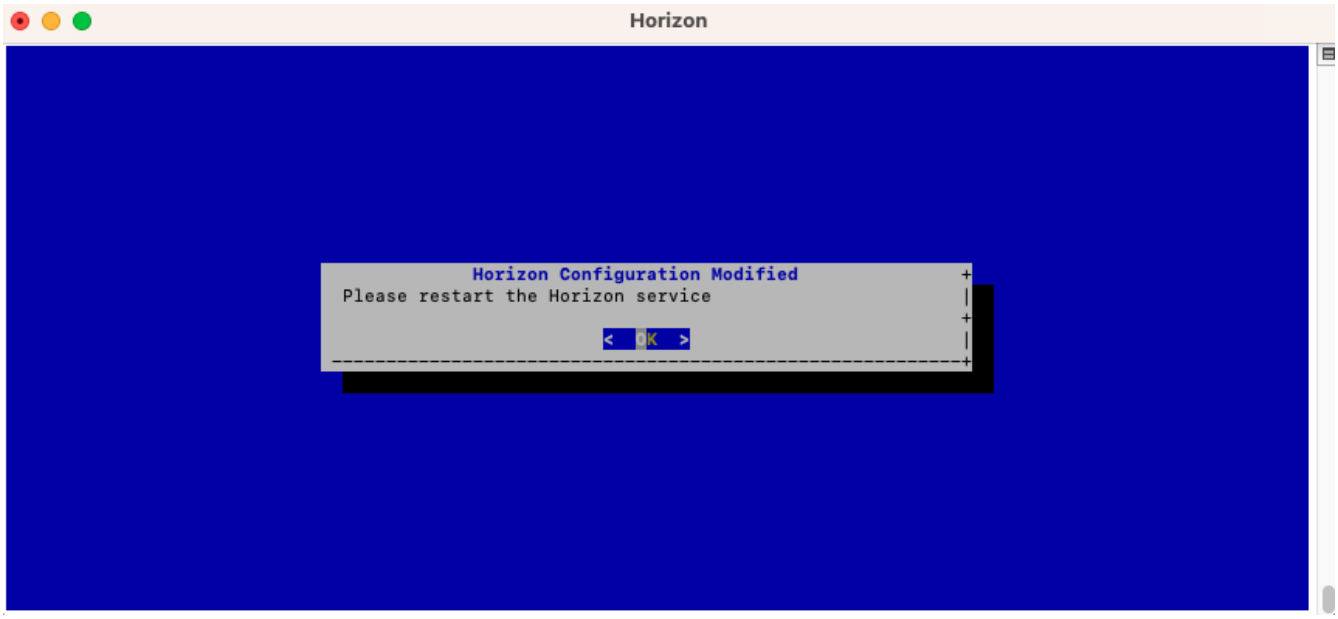
In the Horizon configuration menu, Select **HORIZON_HOSTNAME**:



Specify the DNS FQDN by which Horizon will be accessed:



The Horizon Hostname is configured:



Generating an event seal secret

Horizon will generate functional events when using the software.

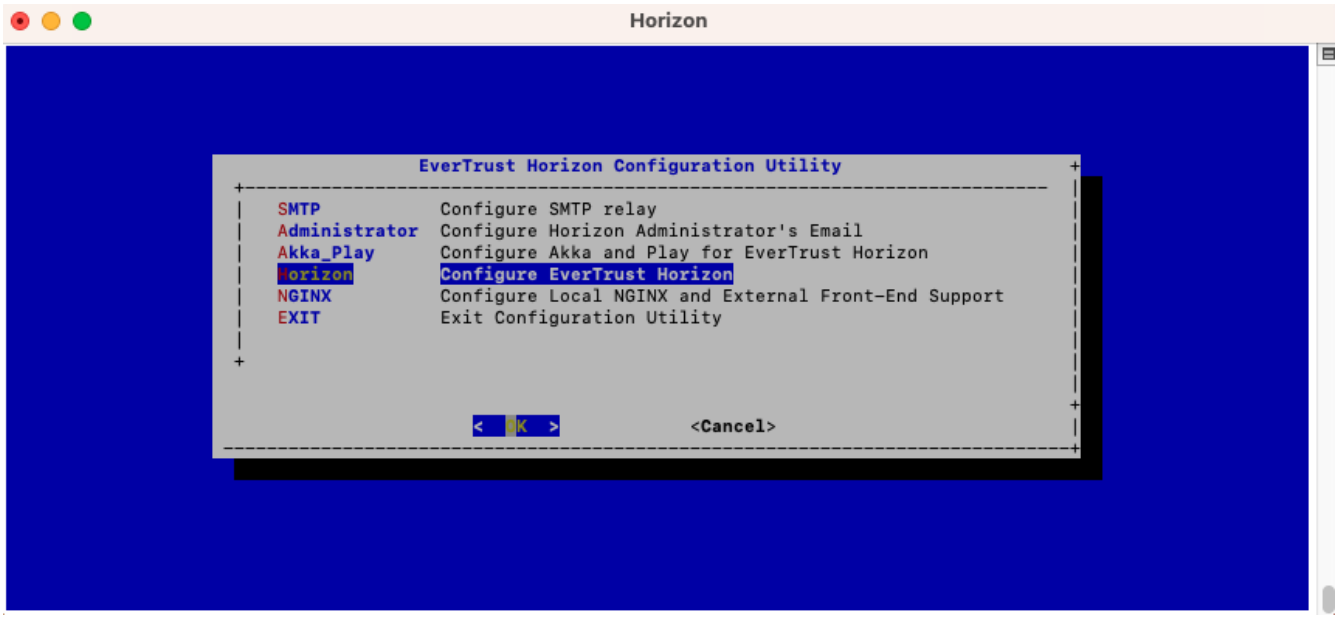
These events are typically signed and chained to ensure their integrity. Therefore, you must specify a sealing secret for this feature to work correctly.

Access the server through SSH with an account with administrative privileges;

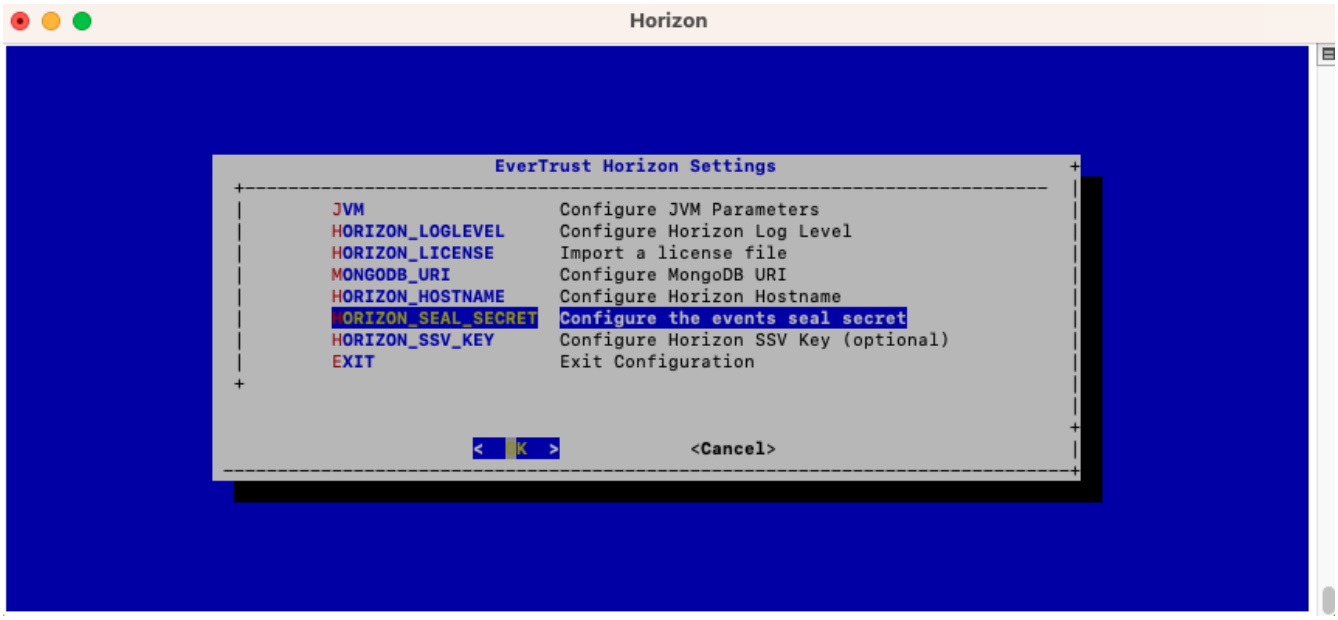
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

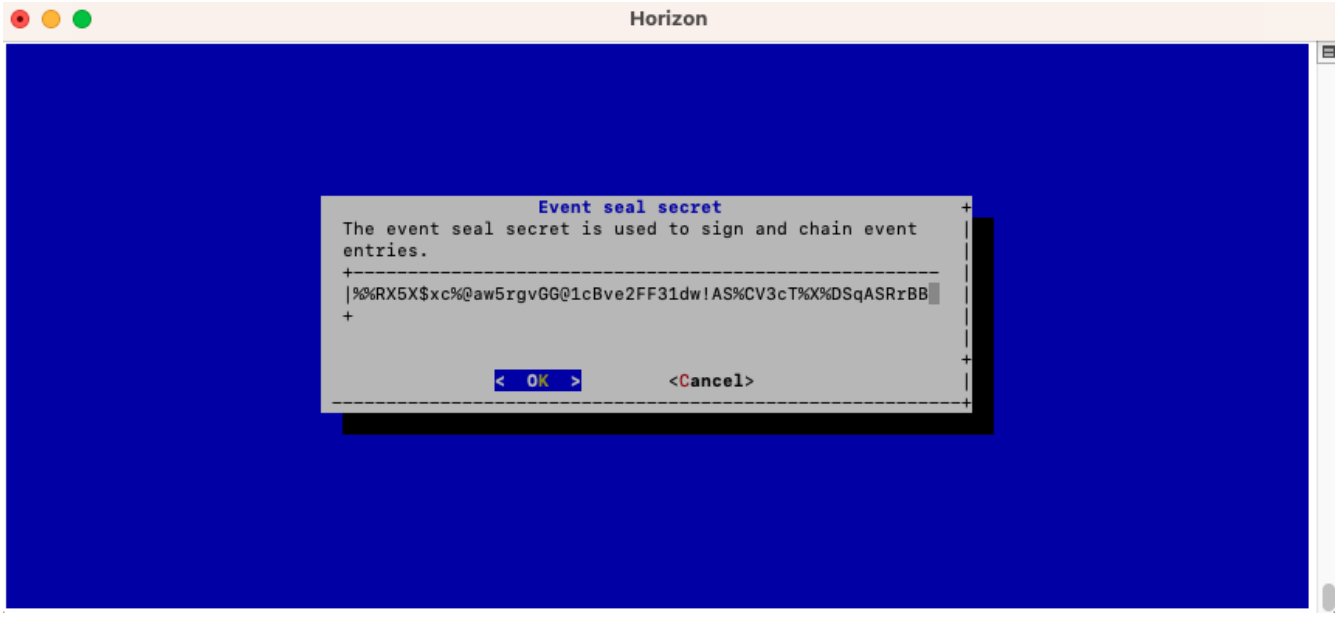
In the main menu, select '**Horizon**':



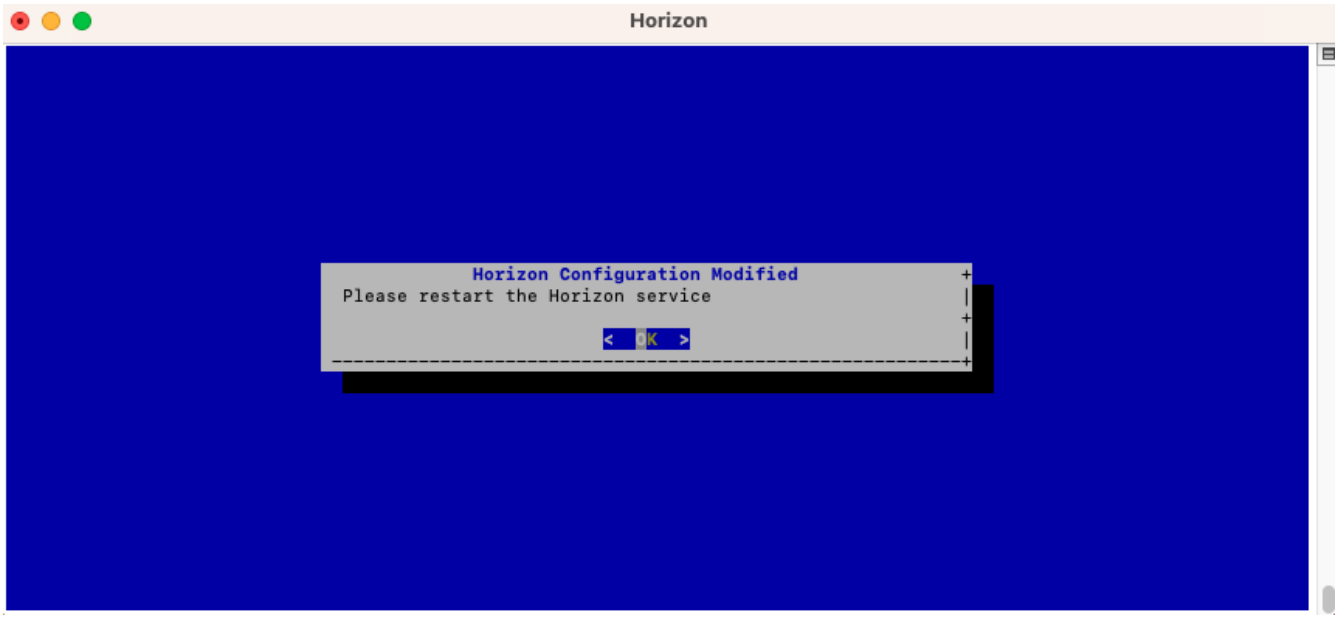
In the Horizon menu, select '**HORIZON_SEAL_SECRET**':



Validate the new event seal secret:



The event seal secret is now configured :



Installing the Horizon license

NOTE

You should have been provided with a '***horizon.lic***' file. This file is a license file and indicates:

- The horizon entitled module(s)
- The limitation in terms of holder per module if any
- A end of support date

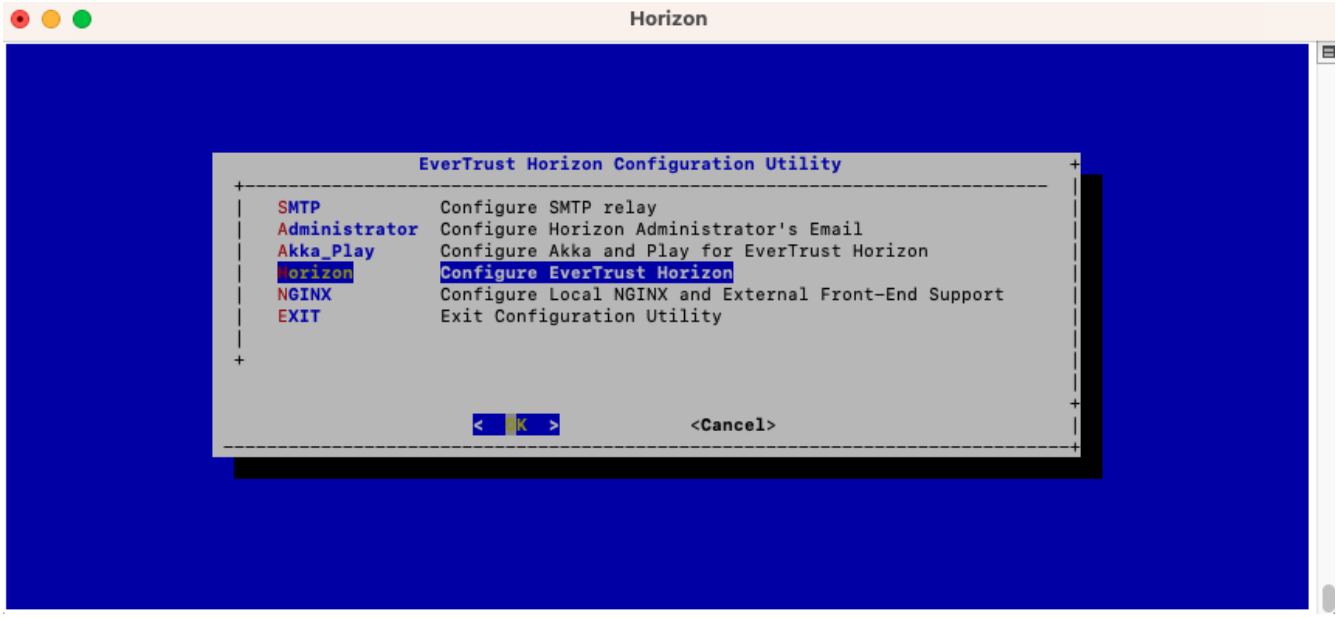
Upload the `horizon.lic` file through SCP under `/tmp/horizon.lic`;

Access the server through SSH with an account with administrative privileges;

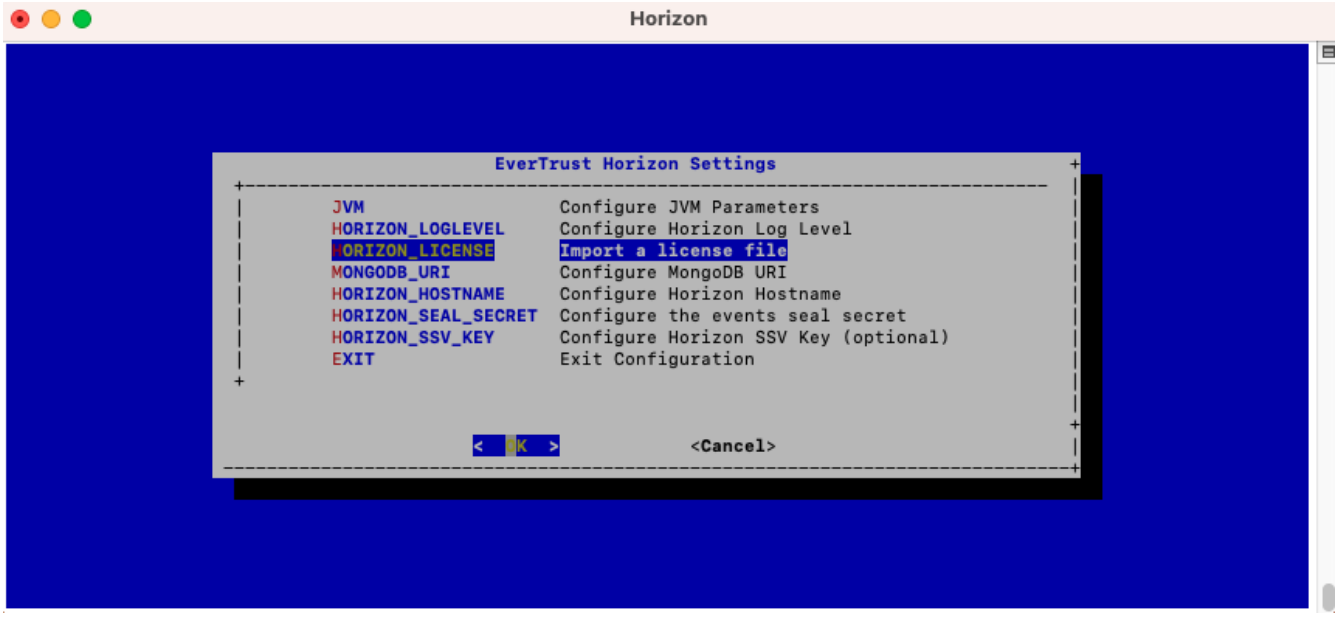
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

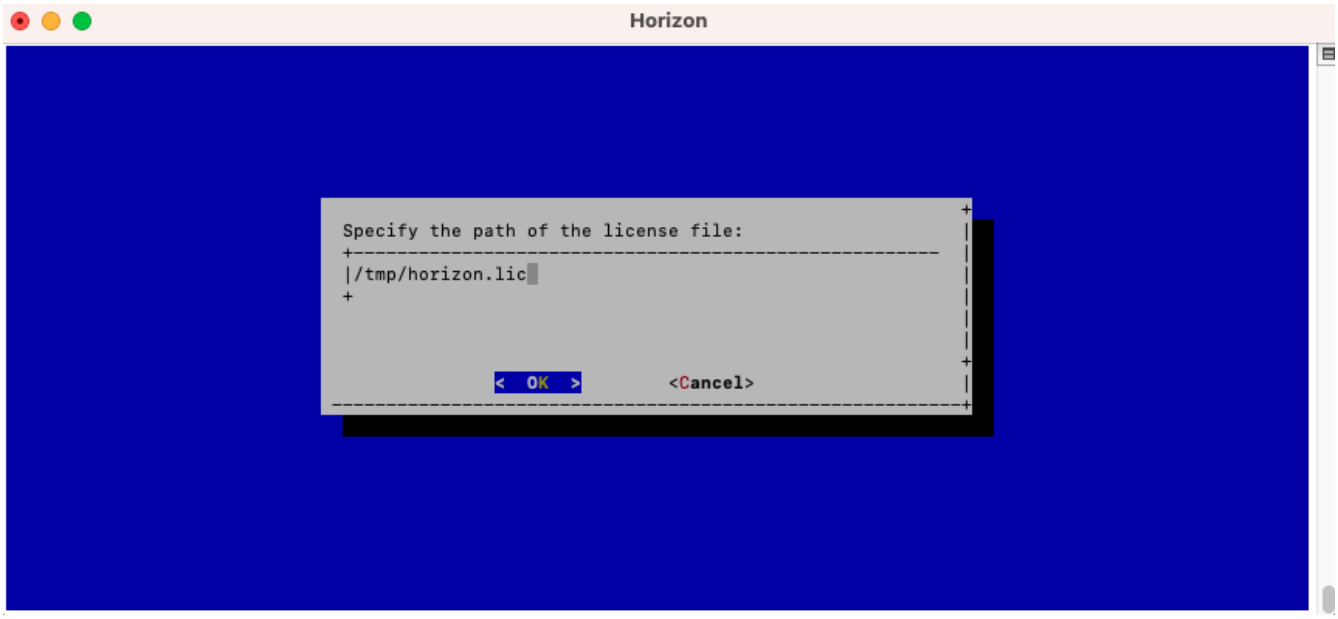
In the main menu, select '**Horizon**':



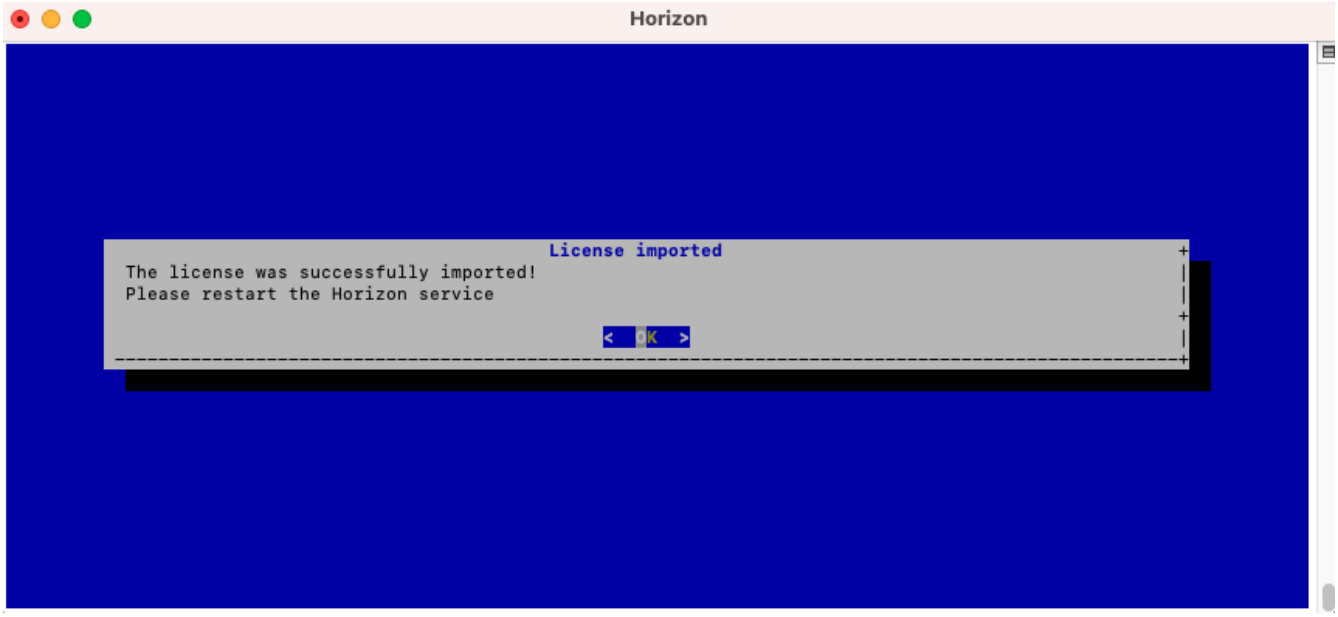
In the Horizon configuration menu, Select '**HORIZON_LICENSE**':



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



The Horizon License is configured:



Restart the horizon service using the following command:

```
$ systemctl restart horizon
```

Horizon Vault Key configuration

NOTE

Horizon stored sensitive data in a secure way using encryption.

Horizon masterkey can be derived from:

- Software key;
- HSM stored key using (PKCS#11 compatible HSMs are supported);
- Azure Key Vault stored key;
- Hashicorp vault stored key;
- FCMS vault stored key.

Please refer to the proper section according to your setup.

Horizon SSV key Configuration (Software)

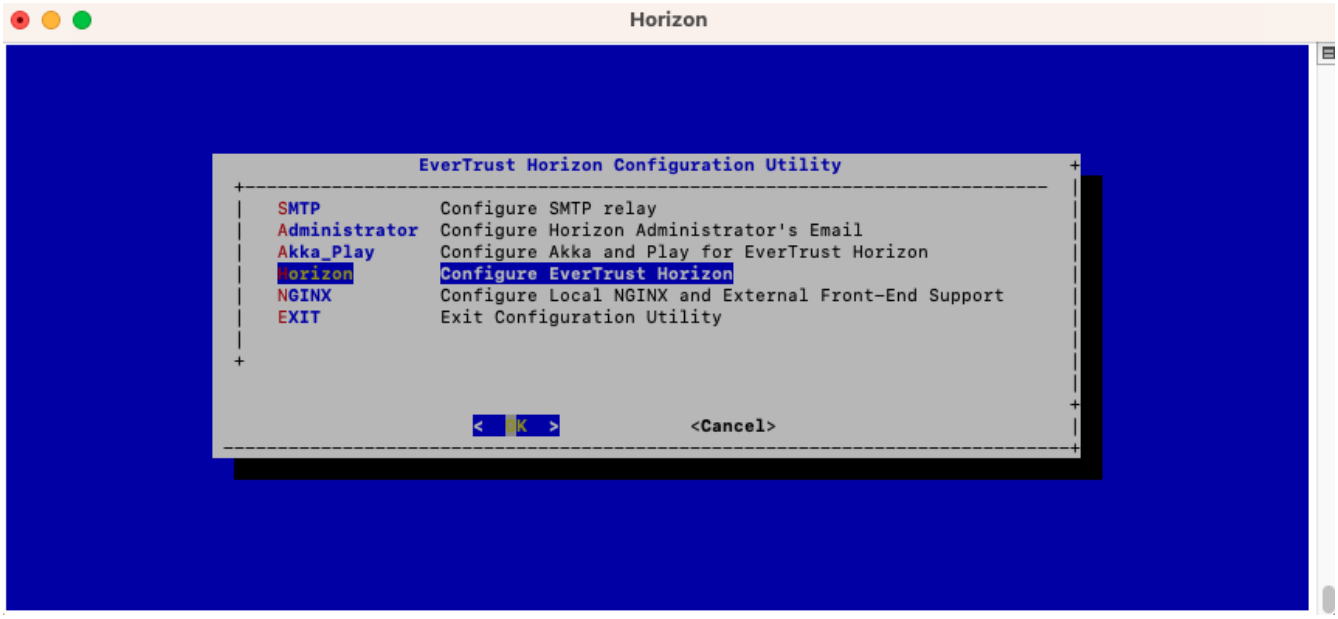
WARNING This section must not be followed if you use another vault than the default one.

Access the server through SSH with an account with administrative privileges;

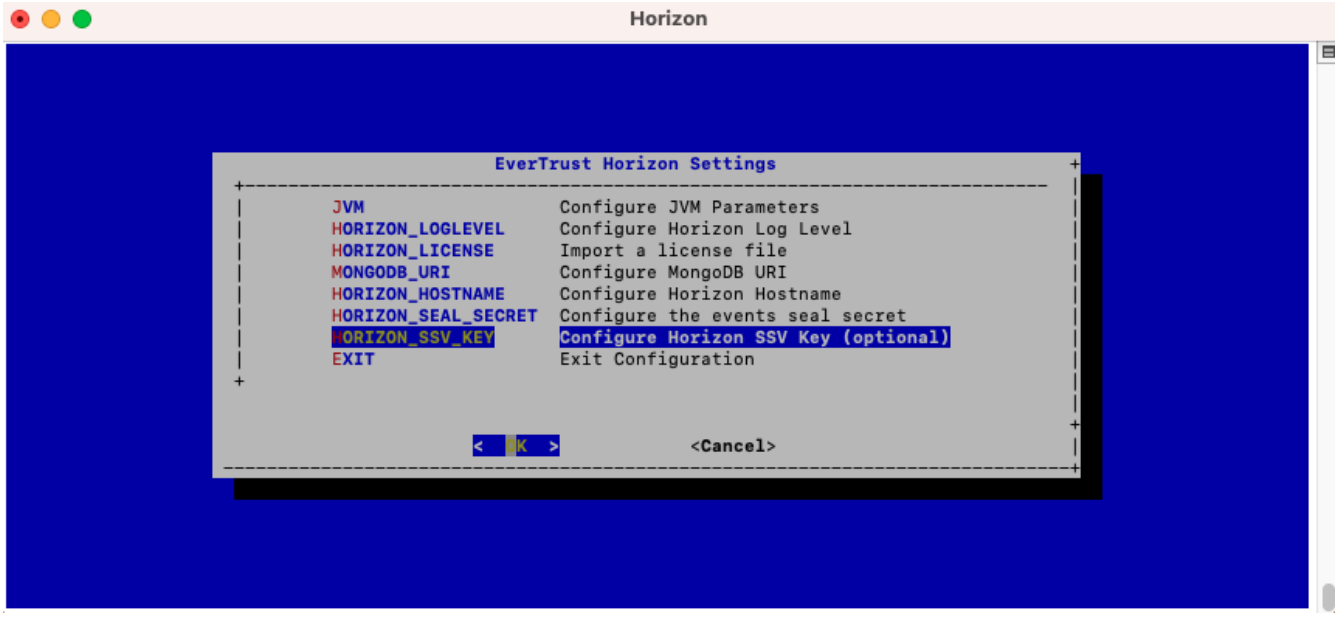
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

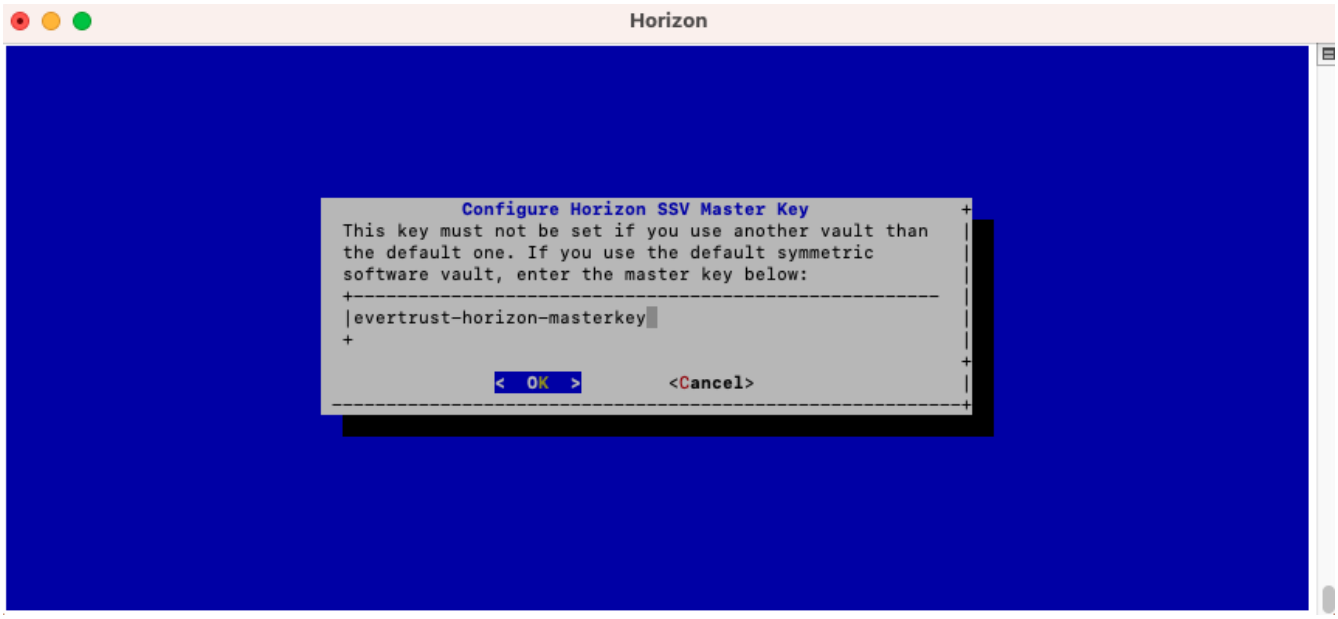
In the main menu, select '**Horizon**':



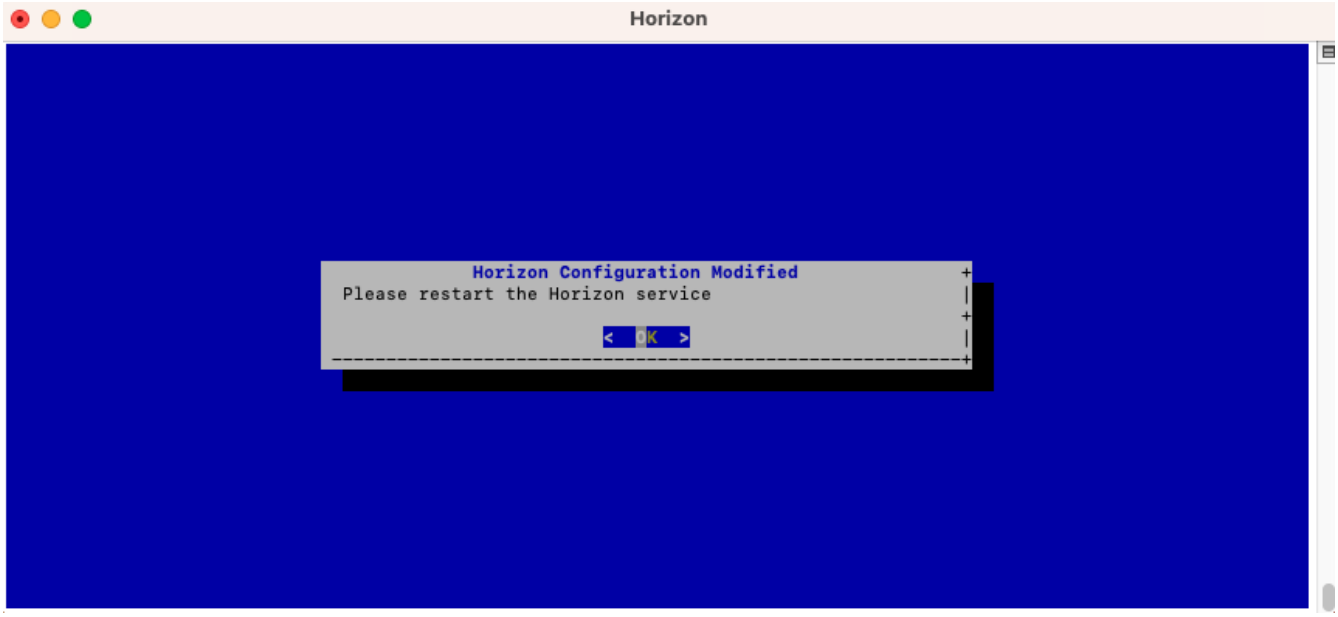
In the Horizon configuration menu, Select '**HORIZON_SSV_KEY**':



Specify the master key that will be used:



Horizon masterkey is configured:



Restart the horizon service using the following command:

```
$ systemctl restart horizon
```

HSM vault Configuration

Horizon supports **PKCS#11 compatible HSM vaults**.

WARNING	This section must not be followed if you use another vault than the HSM vault.

NOTE	HSM middleware should be properly installed and HSM slot initialization should be done using the tools provided by the HSM provider. "horizon" linux user should be member of the proper HSM linux management group to perform cryptographic operations ('nfast' for nCipher nShield HSM or 'hsmusers' for Luna HSM for example).

Access the server through SSH with an account with administrative privileges;

Create a `vaults.conf` configuration file in `/opt/horizon/etc/conf.d` directory with the following content to configure the HSM vault:

```
default {
  module_path = ""
  slot_id = ""
  pin = ""
  label = ""
  allow_master_key_gen = true
}
```

- `module_path`: The path to the PKCS#11 library (string between double quotes);
- `slot_id`: ID of the Slot on the PKCS#11 Module (string between double quotes);
- `pin`: The PIN used to authenticate to your HSM slot (string between double quotes);
- `label`: Label of key (string between double quotes);
- `allow_master_key_gen`: Allow the masterkey to be generated by Horizon if not found in the slot.

Set the permissions using the following commands:

```
$ chown horizon:horizon /opt/horizon/etc/conf.d/vaults.conf
```

Restart the horizon service using the following command:

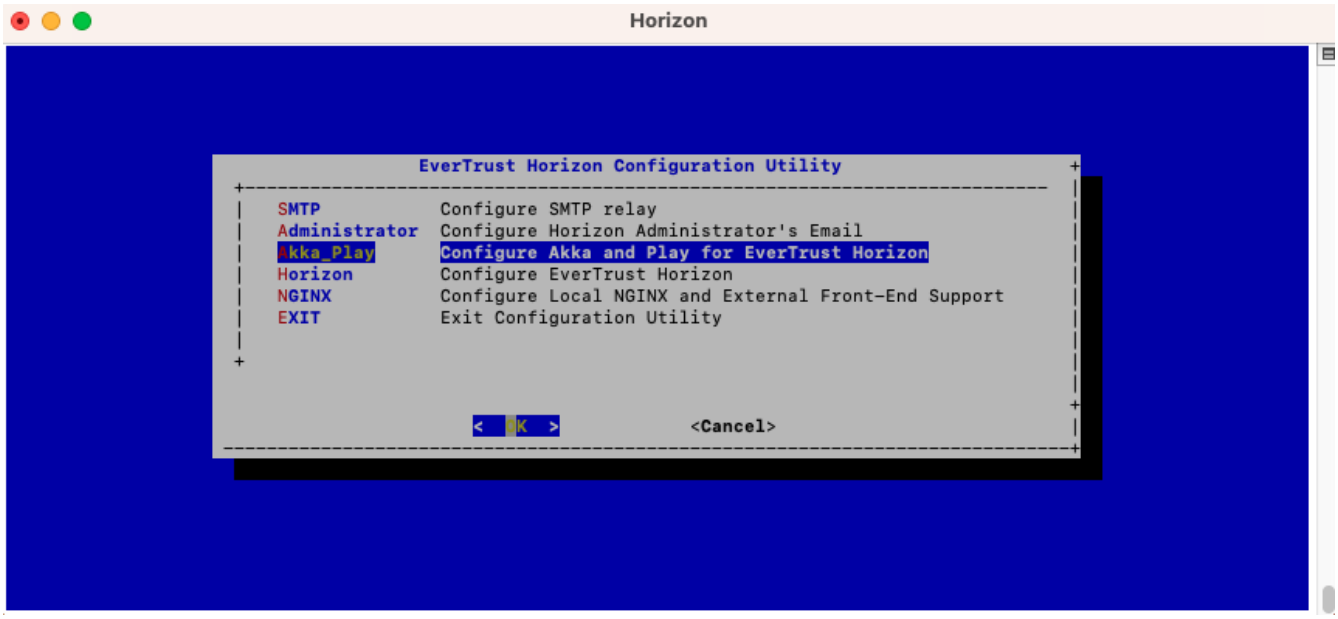
```
$ systemctl restart horizon
```

WARNING	At the end of the installation procedure: • Set `allow_master_key_gen` value to `false`. • Restart the horizon service.

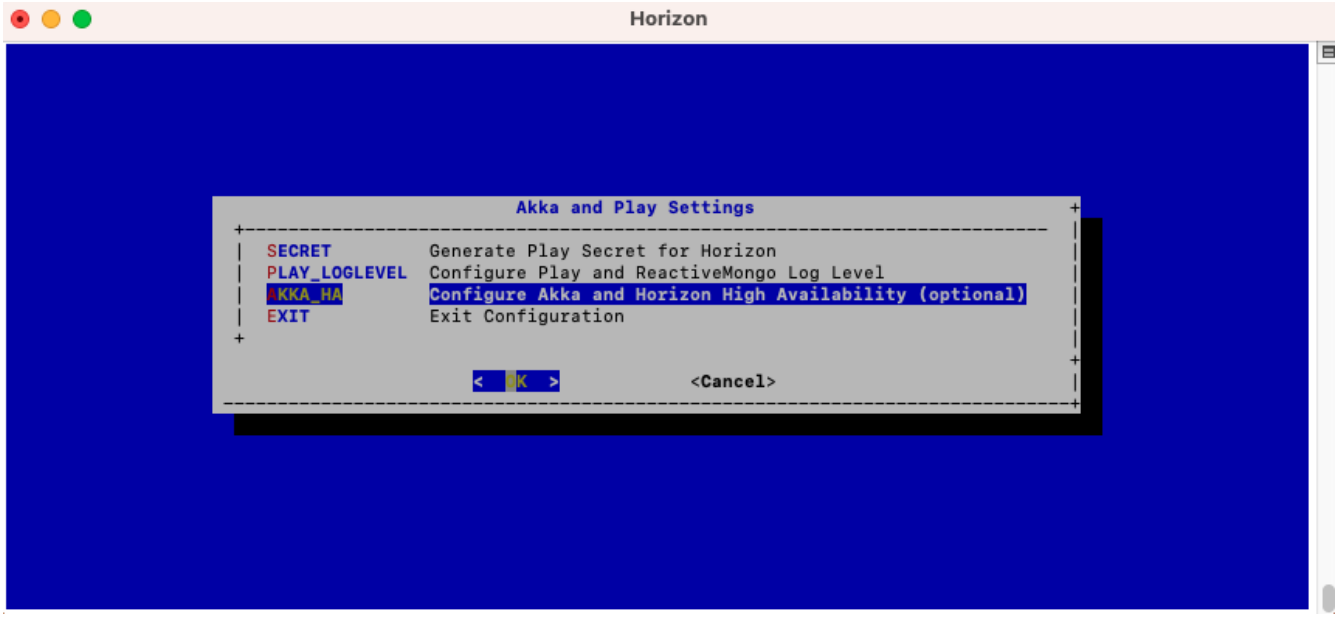
Installing Horizon on a cluster of servers

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

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

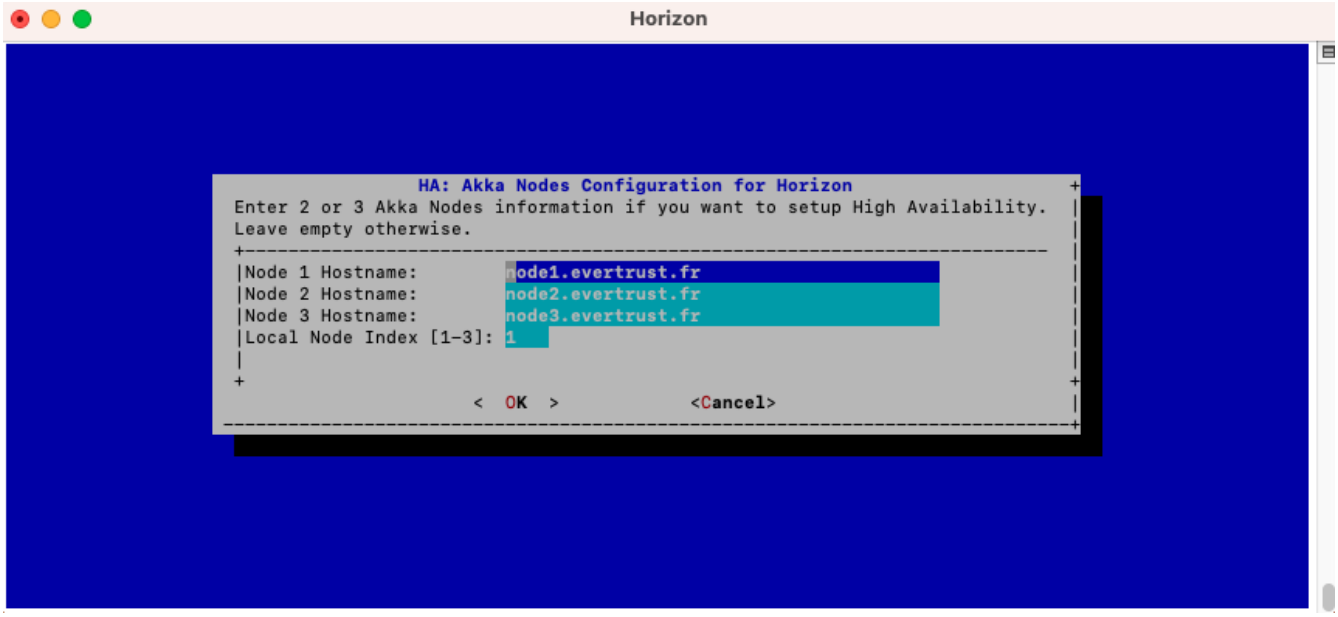


In the Akka_Play menu, select '**AKKA_HA**':



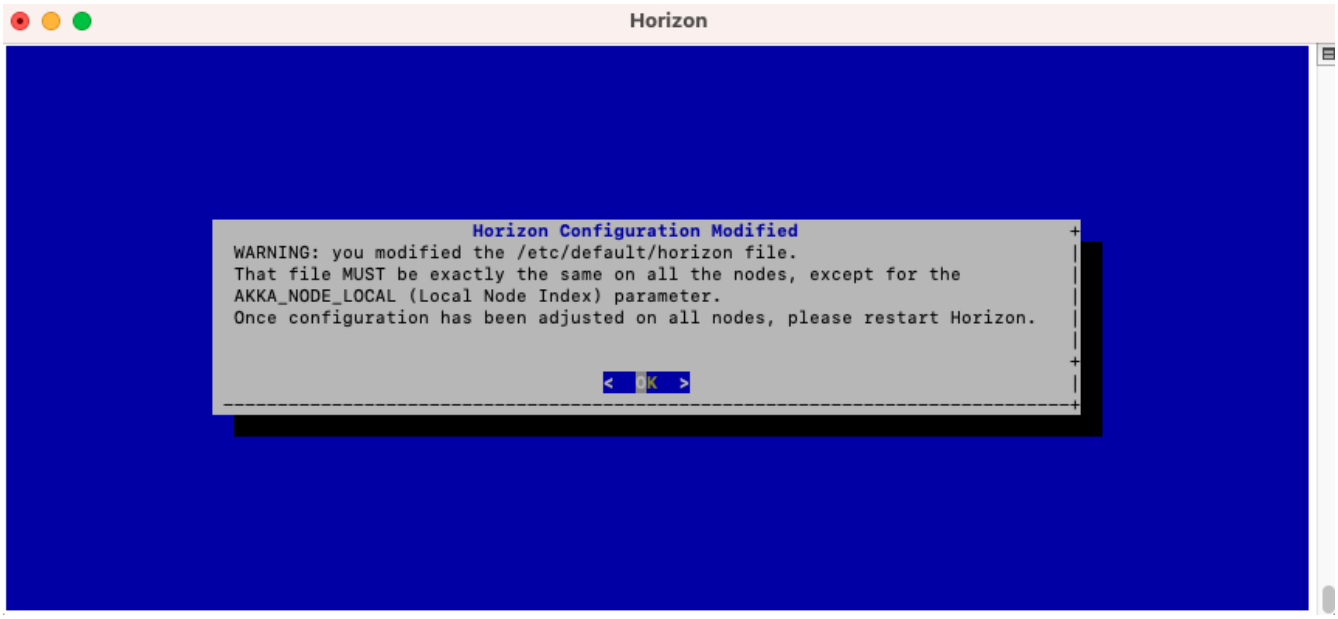
In this menu, specify either the IP address or the DNS name for each server that will be running Horizon on this cluster, as well as the local node index (the number of the node that you are configuring at that moment).

NOTE	Note that the local node index must match the Node Hostname parameter:



Save your changes from the menu.

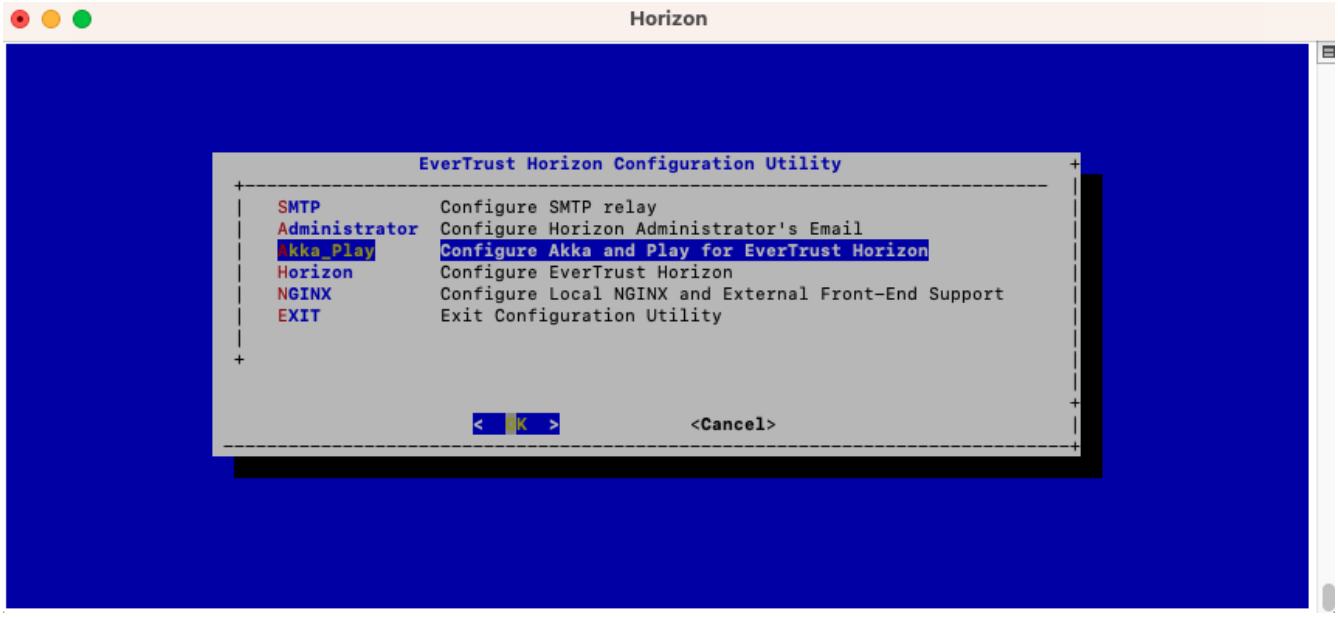
The High Availability mode is now configured on the current node :



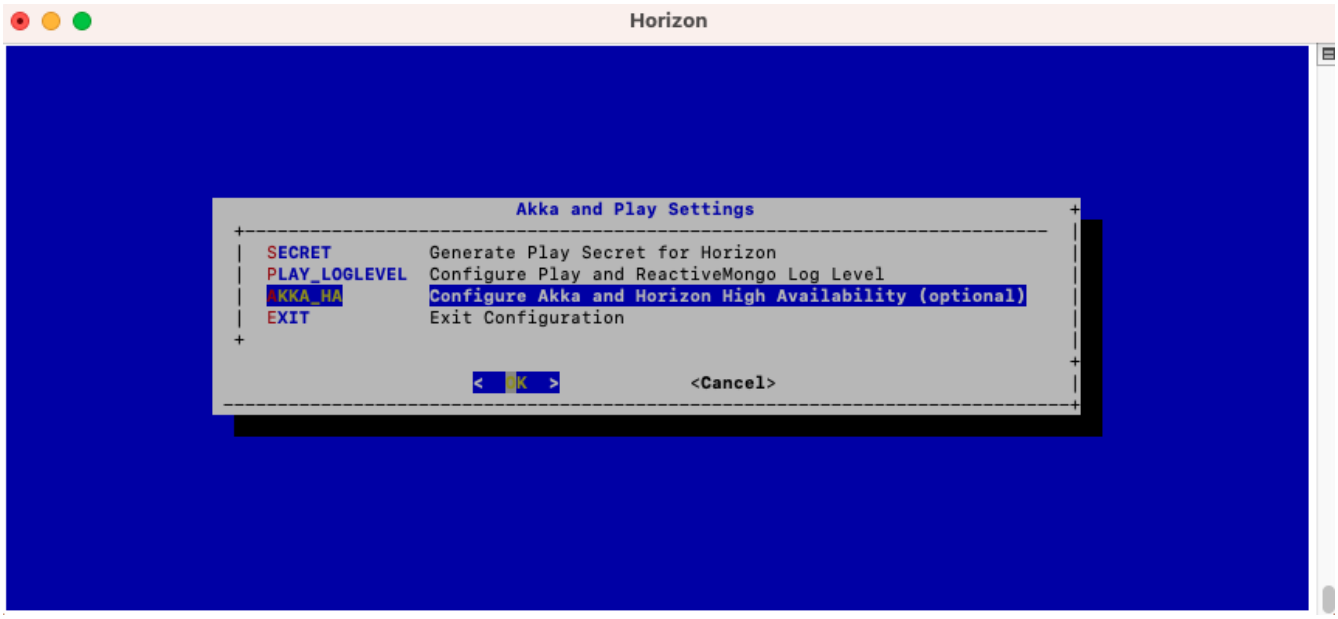
You must now configure your other nodes, but because they belong to the same cluster they need to share the **same secret, the same secret seal event, the same hostname and the same database**. In order to be able to do that, you need to copy the configuration file that was generated by the horizon-config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

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

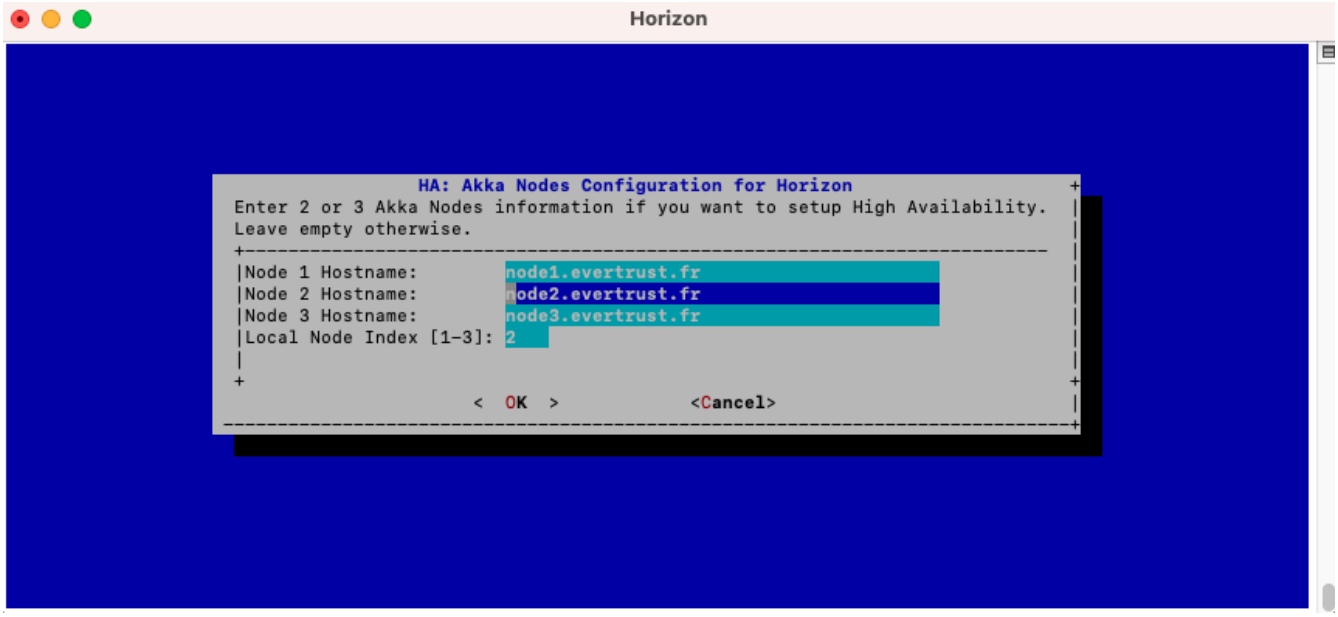
```
$ /opt/horizon/sbin/horizon-config
```



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:



WARNING	You will need to import the Horizon licence file on each node manually, following the guidelines of section Installing the Horizon 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 Horizon service on each one of the nodes:

```
$ systemctl restart horizon
```

2.3.2. Server Authentication Certificate

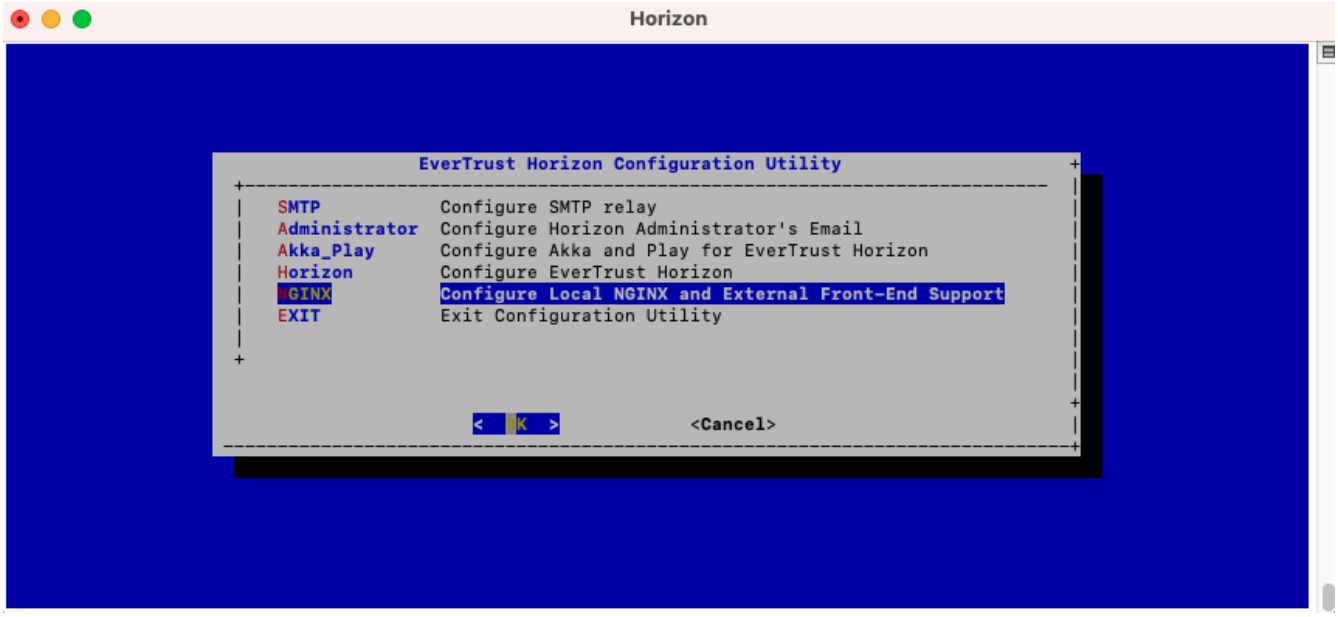
Issuing a Certificate Request (PKCS#10)

Access the server through SSH with an account with administrative privileges;

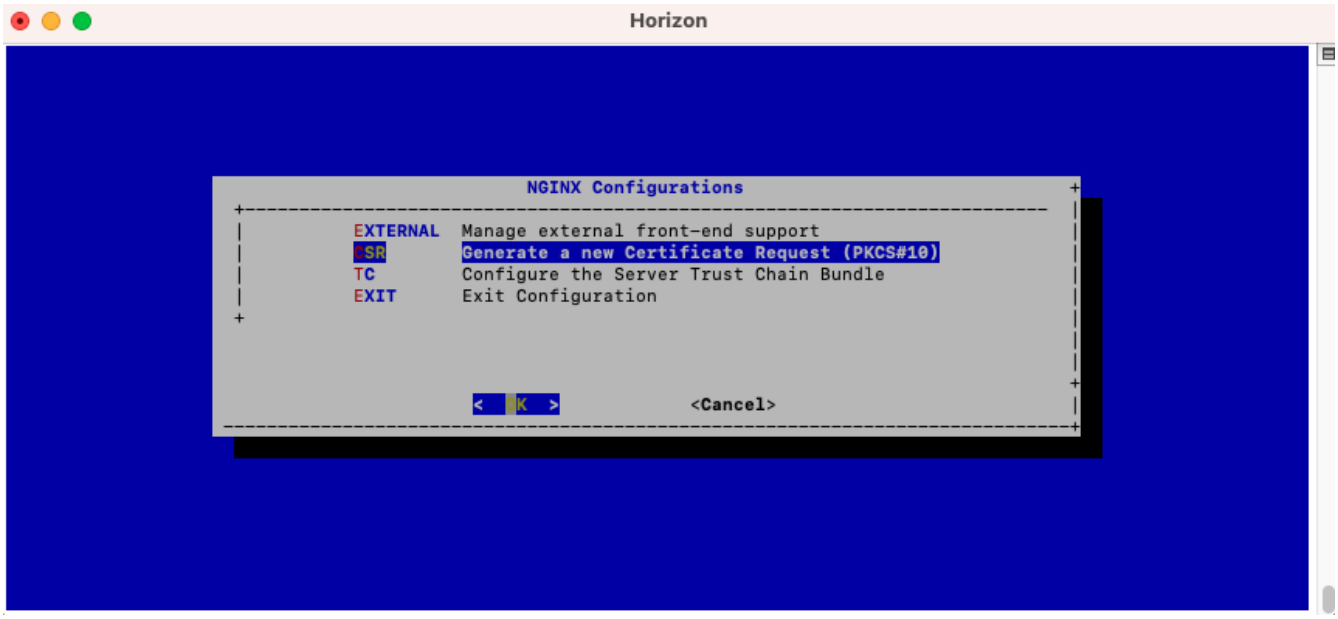
Run the Horizon Configuration Utility with the following command:

```
$ /opt/horizon/sbin/horizon-config
```

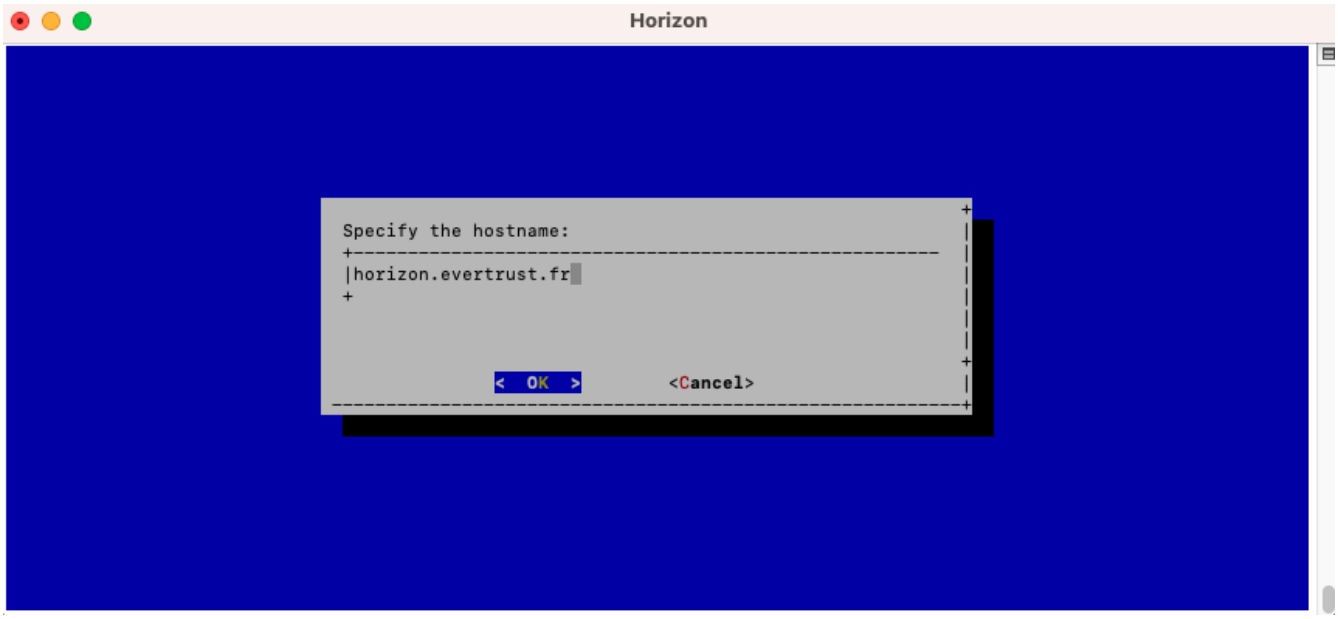
In the main menu, select '**NGINX**':



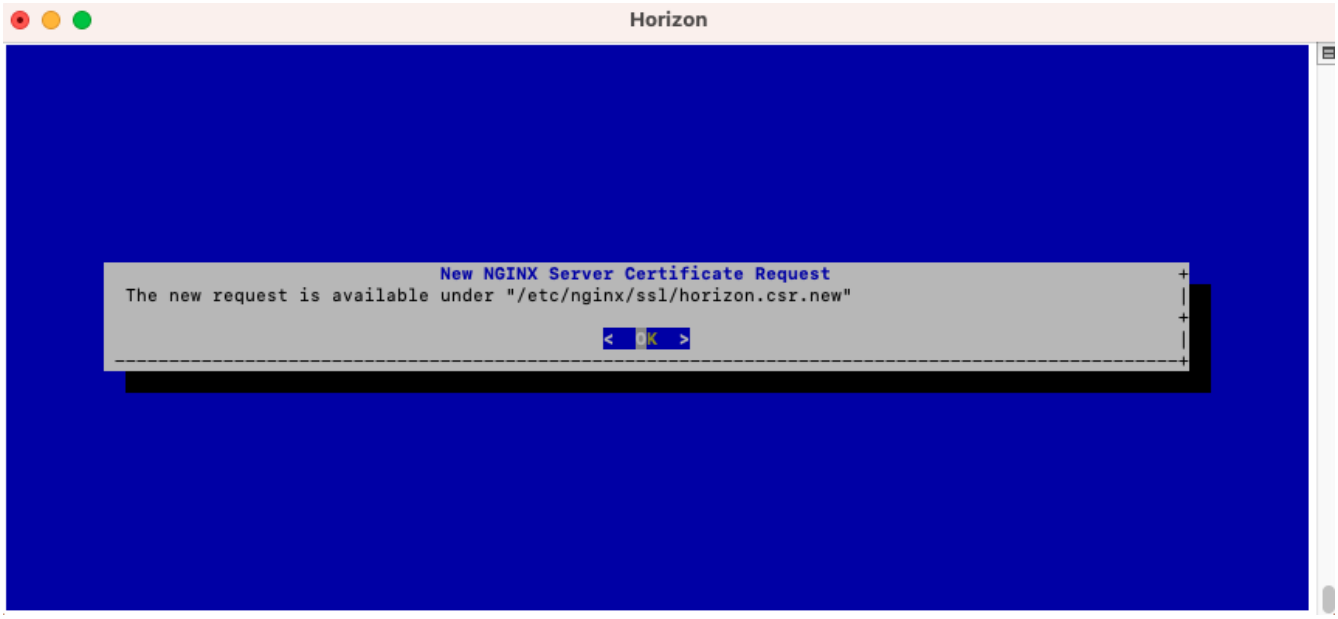
In the NGINX menu, select '**CSR**':



Specify the DNS Name of the Horizon server (by default, the config script takes the Horizon hostname if defined or the local machine hostname otherwise):



The certificate request is generated and available under `/etc/nginx/ssl/horizon.csr.new`:

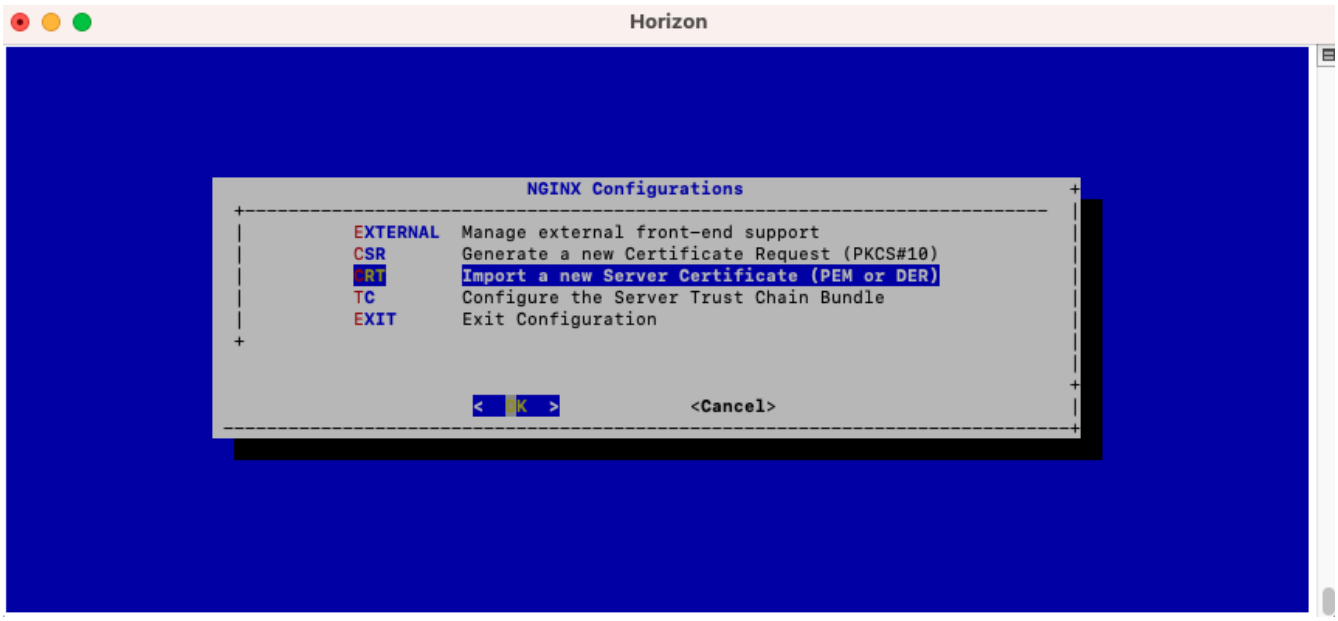


Sign the certificate request using your PKI.

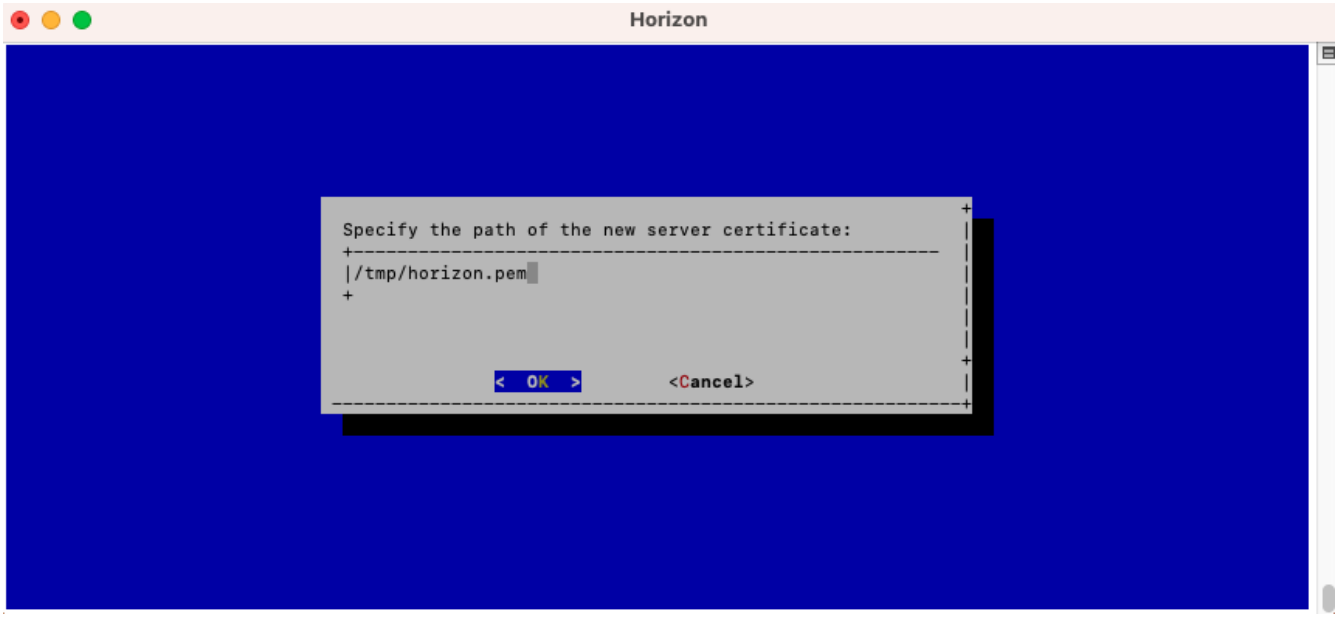
Installing a Server Certificate

Upload the generated server certificate on the Horizon server under `/tmp/horizon.pem` through SCP;

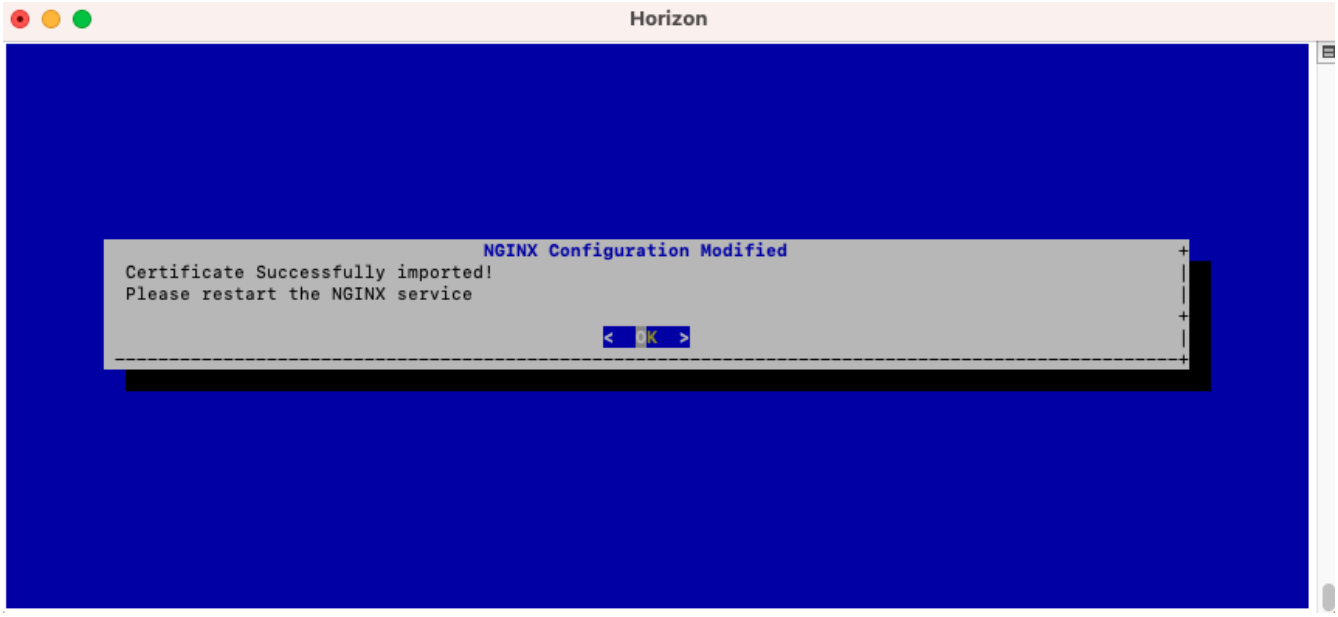
In the NGINX configuration menu, select '**CRT**':



Specify the path `/tmp/horizon.pem` and validate:



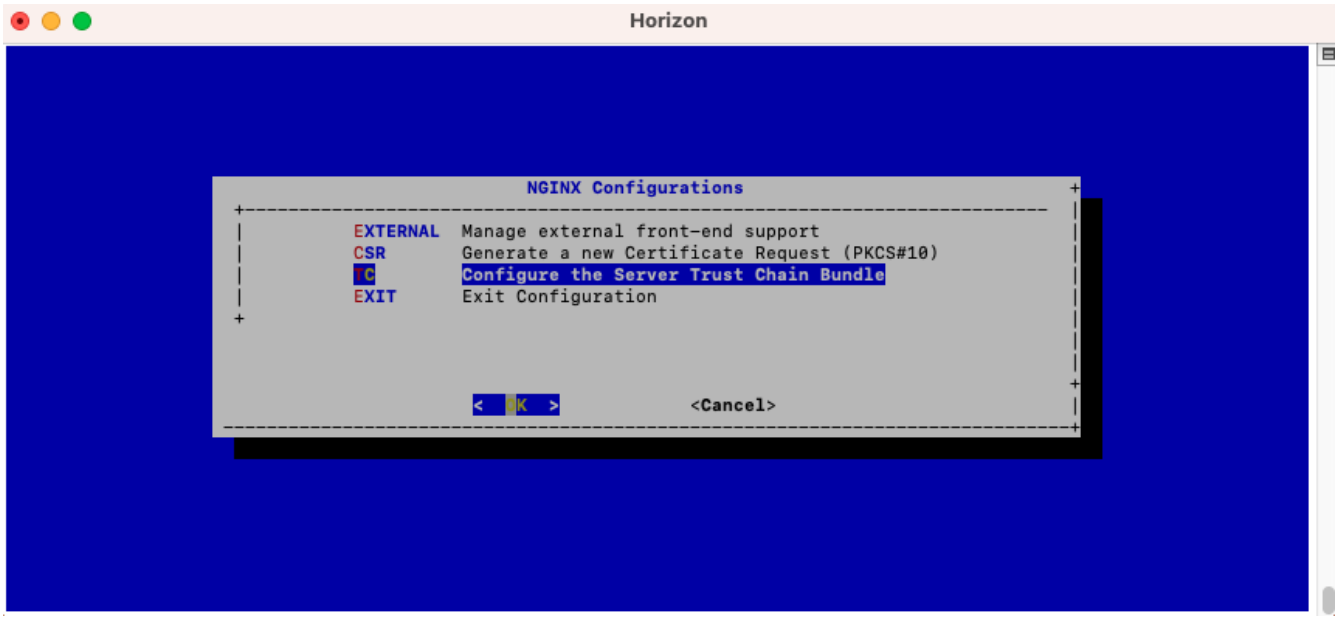
The server certificate is successfully installed:



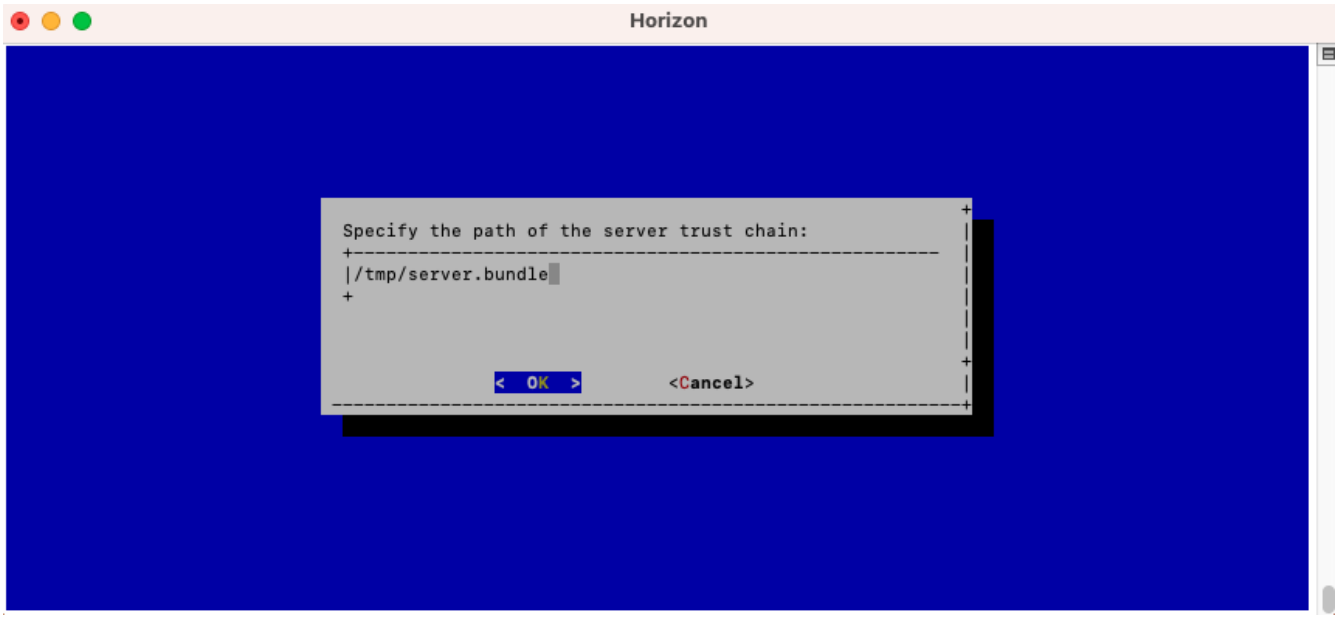
Installing the Server Certificate Trust Chain

Upload the server certificate trust chain (the concatenation of the Certificate Authority certificates in PEM format) on the Horizon server under `/tmp/server.bundle` through SCP;

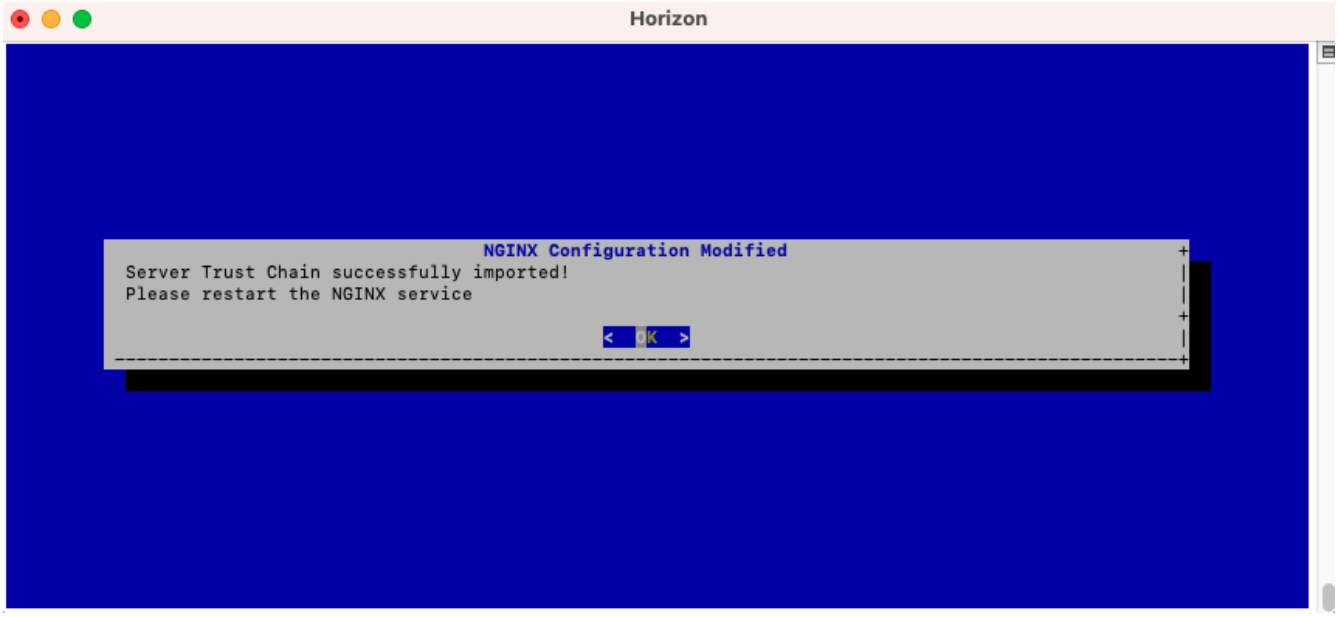
In the NGINX configuration menu, select '**TC**':



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



The server bundle is successfully installed:



Verify the NGINX configuration with the following command:

```
$ nginx -t
```

Restart the NGINX service with the following command:

```
$ systemctl restart nginx
```

2.4. Initial Horizon access

2.4.1. Starting the Horizon services

1. Access the server through SSH with an account with administrative privileges;
2. Start the horizon service with the following command:

   ```
   $ systemctl start horizon
   ```

3. Start the nginx service with the following command:

   ```
   $ systemctl start nginx
   ```

You can now continue to access.pdf to log in to the web UI.

2.5. Upgrade

2.5.1. Standard Procedure

NOTE	The current instructions refer to the standard upgrade procedure. Additional steps might be required, please refer to release notes.

Upgrade the horizon installation

You must retrieve the latest Horizon RPM from the EverTrust repository manually using the user credentials you were provided.

Access the server through SSH with an account with administrative privileges;

Install the Horizon package with the following command:

```
$ yum install horizon-2.2.X-1.noarch.rpm`
```

Upgrade the database schema

Some Horizon versions require that you run migration scripts against your database. Since version 2.1.0, Horizon comes bundled with an `horizon-upgrade` script that handles this migration logic.

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

Access the server through SSH with an account with administrative privileges;

Run the following command:

```
$ /opt/horizon/sbin/horizon-upgrade -t <target version>
```

In most cases, `horizon-upgrade` can detect the version you're upgrading from by checking the database. However, when upgrading from version prior to 2.1.0, 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 `horizon-upgrade` which version you are upgrading from. To do that, simply set the source version explicitly with the -s flag :

```
$ /opt/horizon/sbin/horizon-upgrade -t <target version> -s <source version>
```

Similarly, `horizon-upgrade` will try to use the MongoDB URI that was configured by the Horizon 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/horizon/sbin/horizon-upgrade -t <target version> -m "<mongo uri>"
```

NOTE	The upgrade script requires a MongoDB client to connect to your database (either `mongo` or `mongosh`). If no client is installed on the host where Horizon is running, consider installing the standalone `mongosh` client or running the upgrade script from another host that has access to the database.

2.5.2. Upgrading from a version prior to 2.1.0

These instructions should be followed if you upgrade from a version prior to 2.1.0 to any version greater or equal to 2.1.0.

These steps should be followed in addition to the common upgrade procedure found in iaas/upgrade/upgrade.pdf. None of these steps are automated by `horizon-upgrade`.

Setting an event seal secret

You must manually create an entry to pass an event seal secret to Horizon in the `/etc/default/horizon` file. `horizon-config` won't do that automatically.

To do so, open the `/etc/default/horizon` file with a text editor :

$ vi /etc/default/horizon

And add a new line under the `Horizon variables` section :

```
# Horizon variables
HORIZON_NOTIFICATION_SMTP_HOST=127.0.0.1
HORIZON_HOSTNAME=
HORIZON_DEFAULT_SSV_KEY=
HORIZON_EVENT_SEAL_SECRET=changeme # <- this one
```

Then, near the end of the file, after the `# Setting Horizon Mongo DB uri` section, create a new section for the event seal secret:

```
# Setting the Horizon event seal secret
JAVA_OPTS="$JAVA_OPTS -Dhorizon.event.seal.secret=${HORIZON_EVENT_SEAL_SECRET}"
```

Horizon won't boot if the `HORIZON_EVENT_SEAL_SECRET` is set to `changeme`. Therefore, you should set your secret to something hard to guess. Refer to the iaas/setup/horizon.pdf guide to learn how to generate a seal secret with `horizon-config`.

2.6. Backup and Restore

This section details how to back-up and restore Horizon. Back-up and restore operation can be performed using the back-up and restore tool available under `/opt/horizon/sbin/horizon-backup`. It is designed to be used only in RPM-Based deployments.

For Docker or Kubernetes based deployments, the configuration should be managed by the Docker/Kubernetes management platform, and the database should be backed-up using MongoDB tools.

2.6.1. Backup Procedure

This section details how to back up Horizon configuration elements.

Several elements can be backed up:

- The Horizon configuration files.
- The Horizon MongoDB.

The backup tool allows backing up these elements independently.

```
$ /opt/horizon/sbin/horizon-backup --help
 usage: horizon-backup [-cdho:qs]
    -c | --conf          Backup the configuration files
    -d | --db            Backup the MongoDB database
    -h | --help          Display the 'horizon-backup' help
    -o | --output [path] Specify the backup output folder (default:
'/opt/horizon/var/backup')
    -q | --quiet         Quiet mode
```

To back up the configuration files, run the following command:

```
$ /opt/horizon/sbin/horizon-backup -c
```

The configuration files backup consists of a compressed archive (`.tar.gz`) located under `/opt/horizon/var/backup/`.

To back up the MongoDB database, run the following command:

```
$ /opt/horizon/sbin/horizon-backup -d
```

The MongoDB database backup consists of a compress file (`.gz`) located under `/opt/horizon/var/backup/`.

To run a complete backup, execute the following command:

```
$ /opt/horizon/sbin/horizon-backup -c -d
```

NOTE

- The backup output folder can be overridden using the `-o | --output` parameter
- The backup tool can operate in quiet mode (when scheduled in a cron job) using the `-q | --quiet` parameter

2.6.2. Restoration Procedure

This section details how to restore horizon configuration elements.

WARNING	This restore procedure only applies to the exact same application version as the backup file.

Restoration operation should be performed while the Horizon service is not running. Stop the Horizon service with the following command:

```
$ systemctl stop horizon
```

To restore a configuration backup, run the following command:

```
$ tar xzpvf [horizon configuration backup archive path] -C/
```

To restore the MongoDB database, run the following command:

```
$ mongorestore --uri="[MongoDB URI]" --drop --gzip --archive=[horizon MongoDB backup
archive path]
```

NOTE	The MongoDB URI can be retrieved from the `/etc/default/horizon/_*` configuration file, as `MONGODB_URI` parameter.

The Horizon service can now be started with the following command:

```
$ systemctl start horizon
```

2.7. Uninstallation

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

NOTE	Uninstalling Horizon consists of uninstalling: • The Horizon service; • The MongoDB service; • The NGINX service.

2.7.1. Uninstalling Horizon

Access the server through SSH with an account with administrative privileges;

Uninstall Horizon with the following commands:

```
$ systemctl stop horizon
$ yum remove horizon
$ rm -rf /opt/horizon
$ rm -rf /var/log/horizon
$ rm -f /etc/default/horizon
```

2.7.2. Uninstalling NGINX

Access the server through SSH 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
```

2.7.3. Uninstalling MongoDB

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

3.1.2. 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/horizon
NAME                    CHART VERSION   APP VERSION DESCRIPTION
evertrust/horizon       0.5.3           2.2.2       EverTrust Horizon Helm chart
```

3.1.3. Configuring the namespace

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

```
$ kubectl create namespace horizon
```

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

- A license secret containing your Horizon license file
- An image pull secret, allowing Kubernetes to authenticate to the EverTrust's container

repository

Creating the license secret

You should have a license file for your Horizon installation, most probably named `horizon.lic`. To convert this file to a Kubernetes secret, run:

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

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 Horizon 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 horizon
```

3.1.4. 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:

```yaml
image:
  pullSecrets:
    - evertrust-registry

license:
  secretName: horizon-license
  secretKey: license
```

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

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

Please allow a few minutes for the Horizon instance to boot up. You are now ready to go on with the :access.pdf. This instance will allow you to test out if Horizon is working correctly on your cluster.

However, this installation is not production-ready. Follow our k8s/production.pdf to make sure your instance is fit to run in your production environemnt.

3.2. Production checklist

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

3.2.1. Operating the database

All persistant data used by Horizon 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:
  uri:
    valueFrom:
      secretKeyRef:
        name: <secret name>
        key: <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.

3.2.2. Managing secrets

Storing secrets is a crucial part of your Horizon installation. On cloud-native installations like on Kubernetes, we recommend using SSV (Secure Software Vault) to encrypt sensitive data: a master passphrase will be 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
`vaults.*.master_password`	SSV password used to encrypt sensitive data in database.	Highest impact: database would be unusable
`events.secret`	Secret used to sign and chain events.	Moderate impact: events integrity would be unverifiable
`externalDatabase.uri`	External database URI, containing a username and password.	Low impact: reset the MongoDB password
`appSecret`	Application secret use to encrypt session data.	Low impact: sessions would be reset
`mailer.password`	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:
  valueFrom:
    secretKeyRef:
      name: <secret name>
      key: <secret key>
```

WARNING Always store auto-generated secrets in a safe place after they're generated. If you ever uninstall your Helm chart, the deletion of the SSV secret will lead to the impossibility of recovering most of your data.

3.2.3. 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 Horizon 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 Horizon cluster Pods among multiple nodes in different availability zones to reduce the risk of Single Point of Failure.

3.2.4. Configuring ingresses

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

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

We support autoconfiguration for major ingress controllers: Kubernetes Ingress NGINX and Traefik. Autoconfiguration is the recommended way of configuring your ingress as it will handle configuration quirks for you. To enable autoconfiguration, set the `type` key to your ingress controller in the ingress definition. Accepted values are `nginx` and `traefik`.

```
ingress:
  enabled: true
  type: "" # nginx or traefik
  clientCertificateAuth: true
  hostname: horizon.lab
  tls: true
```

`clientCertificateAuth` can be used to control whether to ask for a client certificates when users access Horizon.

If you wish to manually configure your ingress or use another ingress controller, head to the Manual ingress configuration section.

Manual ingress configuration

If you do not wish or cannot use autoconfiguration, you should ensure your ingress controller is correctly configured to enable all Horizon features.

- When requiring client certificates for authentication, the web server should not perform checks to validate that the certificate is signed by a trusted CA. Instead, the certificate should be sent to Horizon through a request header, base64-encoded. The header name used can be controlled using the `clientCertificateHeader`.
- Some endpoints should not be server over HTTPS, in particular those used for SCEP enrollment. You may want to create an HTTP-only ingress for serving paths prefixed by `/scep` and `/certsrv`, and prevent those from redirecting to HTTPS.

3.3. Upgrade

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

3.3.1. Upgrading the chart

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

```
$ helm repo update evertrust
```

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

```
$ helm search repo evertrust/horizon
NAME                        CHART VERSION   APP VERSION DESCRIPTION
evertrust/horizon           0.5.3           2.2.2       EverTrust Horizon Helm chart
```

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

```
$ helm upgrade <horizon> evertrust/horizon \
 --values override-values.yaml \
 --version 0.5.3
```

The chart will automatically create a `Job` that runs an upgrade script when it detects that the Horizon 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 Horizon, 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.

3.3.2. Specific chart upgrade instructions

Upgrading to 0.3.0

- Loggers are now configured with an array instead of a dictionary. Check the `values.yaml` format and update your override `values.yaml` accordingly.
- The init dabatabase parameters (`initDatabase`, `initUsername` and `initPassword`) have been renamed and moved to `mongodb.horizon`.

Upgrading to 0.5.0

- The ingress definition has changed. The `rules` and `tls` keys have been removed in favor of a more user-friendly `hostname` that will autoconfigure the ingress rules, and a boolean `tls` key that will enable TLS on that ingress. Check the Ingress section.

3.4. Uninstall

To uninstall Horizon from your cluster, simply run :

```
$ helm uninstall horizon -n horizon
```

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

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

4. Installing with Docker

Log in to the EverTrust Docker registry:

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

Pull the latest Horizon image:

```
$ docker pull registry.evertrust.io/horizon:2.2.X
```

The Horizon 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="" \
    -e MONGODB_URI="" \
    -e APPLICATION_SECRET="" \
    -e HOSTS_ALLOWED.0="" \
    -e HOSTS_ALLOWED.1="" \
    -p [port]:9000 \
    registry.evertrust.io/horizon:2.2.X
```

Here's a full list of environment variables used by the default config:

Variable	Description
LICENSE	A valid Horizon license string, base64-encoded.
APPLICATION_SECRET	Application secret used by Horizon
MONGODB_URI	A valid MongoDB URI. See mongo_uri_config.
SSV_PASSWORD	If empty, the APPLICATION_SECRET value will be used.
HOSTS_ALLOWED	Array of hosts. Append the array index after a dot (the nth allowed host variable name would be HOSTS_ALLOWED.n).
SMTP_HOST	
SMTP_PORT	
SMTP_SSL	
SMTP_TLS	
SMTP_USER	
SMTP_PASSWORD	

NOTE Your license usually contains newline characters, that you must replace by '\n' when

setting it through the environment.

Should you need to have full control over the config in the container, and assuming that you have a valid configuration according to the **Administration Guide** and have a license file, you can mount a folder containing the configuration to /horizon/etc :

```
$ docker run \
    -v [configurationPath]:/horizon/etc:rw \
    -p [port]:9000 \
    registry.evertrust.io/horizon:2.2.X
```

NOTE

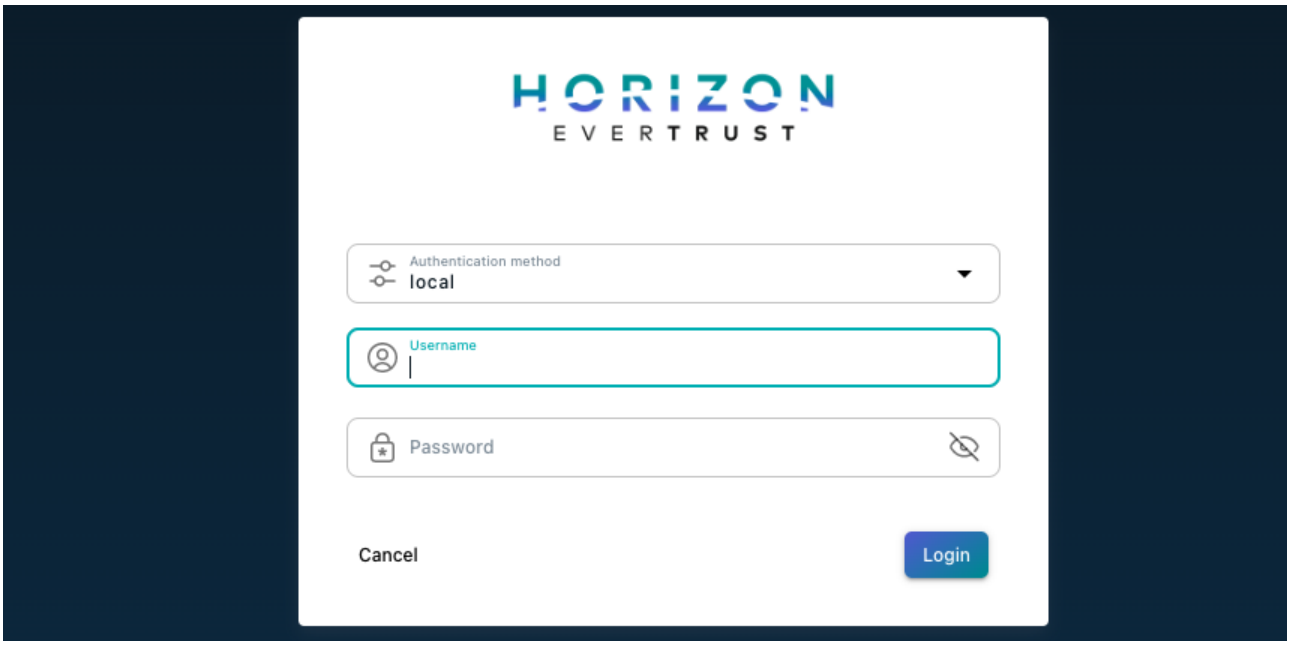
You need to use the configuration folder's absolute path.

The configuration folder must contain the `horizon.lic` file.

5. First login

5.1. Log in

1. Launch a web browser
2. Browse to `https://[Horizon IP or FQDN]/ui#/login`:



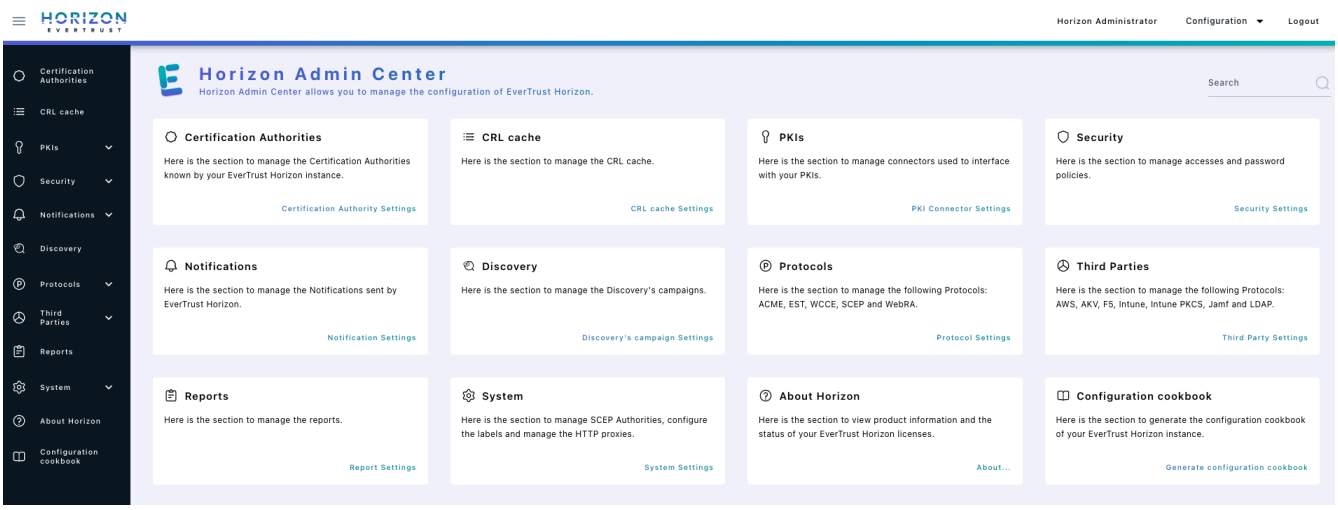
If you don't see any Authentication method (you should see **Local** if the database has been setup), you must Manually create the initial user.

NOTE

The default administration credentials are:

- Login: `administrator`
- Password: `horizon`

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



CAUTION

It is **highly recommended** to create a dedicated administration account and

delete the default one, or at least modify the default administrator password.

5.2. Manually create the initial user

Launch a MongoDB shell to access your database and run the following command to create the initial administrator:

```
use horizon;
db.security_local_identities.insert({"identifier":"administrator","hash":"$6$8JDCzmb9X
DpOwtGQ$7.kRdgIjPYR/AxPbzKsdkBH3ouCgFbqyH9csjcr5qIoIXK/f2L6bQYQRhi9sdQM4eBm8sGUdEkg.TV
OQ1MRsA/","name":"Horizon Administrator"});
db.security_principals.insert({"identifier":"administrator","permissions":[{"value":"c
onfiguration:*"},{"value":"lifecycle:*"}],"roles":[]});
db.security_identity_providers.insert({"enabled":true,"type":"Local","name":"local","e
nabledOnUI":true});
```

6. Troubleshooting

6.1. Horizon Doctor

Horizon doctor is a tool that performs checks on your Horizon installation as well as its required dependencies to ensure that everything is configured properly. The tool is targeted towards troubleshooting during installation or update procedures. Note that the tool requires root permissions to run.

6.1.1. Performed checks

At the moment, Horizon Doctor checks for:

OS checks

- Checks for installed Horizon version, MongoDB version, Java version, Nginx version, OS Version.
  - If the OS is a RedHat distribution, checks if the RedHat subscription is active.
  - If Mongo is not installed locally, it notices it as an information log.
- Checks for **SELinux**'s configuration: throws a warning if it is enabled, says ok if it is on permissive or disabled.
- Checks for the status of the necessary services: **postfix**, **mongod**, **nginx** and **horizon**.
  - If the **postfix** service is running, tries to connect via a TCP SYN on the port 25 of the **relayhost** specified in the */etc/postfix/main.cf* file and throws an error if it can't.
- Checks how long the **Horizon** 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 horizon:horizon.
- 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 horizon:horizon.
- Checks for existence and permissions of the **vault** file: the permissions are expected to be at least 640 and the file is supposed to belong to horizon:horizon.
- Checks for the permission of the Horizon directory (default: /opt/horizon): 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 Horizon and throws a warning if the default ones are used (min = 2048 and max = 3072).

- Retrieves the **Horizon DNS hostname** and stores it for a later test (throws an error if it has not been set).
- Checks for the **Horizon Play Secret** and **Horizon Event Seal Secret**: these are the Horizon application secrets and should be different from default value thus Horizon Doctor throws an error if either of them is equal to the default value (*changeme*).
- 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 **Horizon licence file** to retrieve its expiration date as well as the licence details (number of holders per category).

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 *horizon*; 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 in every network interface and throws a warning if it is the case (specifying the interface with IPv6 turned on).

TLS checks

- Checks for existence and permissions of the **Horizon 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 **Horizon 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 **Horizon server certificate** file and checks if the **Horizon hostname** is present in the **SAN DNS names** of the certificate, throws an error if it is not there.

6.1.2. Log packing option

If the Horizon doctor is launched with the *-l option*, it will pack the logs of the last 7 days (in */opt/horizon/var/log*) as well as the startup logs (the */var/log/horizon/horizon.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 **Horizon 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:

```
$  horizon-doctor -l
```

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

```
$ horizon-doctor -l 30
```

6.1.3. Saving the doctor's output

If the Horizon 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 command line 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 `horizon-doctor.out` instead of the stdout :

```
$ horizon-doctor -o horizon-doctor.out
```

6.1.4. Help menu

To display Horizon doctor's help menu, use the `-h` option.

6.2. Additional checks

- Ensure that you are using an up-to-date web browser when trying to access the Horizon web interface
- Ensure that Javascript in turned on in your web browser
- Ensure that your user machine can access the server where Horizon was installed
- If several hostnames have been set up for the Horizon interface, ensure that every single one of them is present in the TLS certificate SAN DNS names