Handling secrets in ESS Pro

Overview

The `matrix-stack` Helm chart provides flexible secret configuration options:

- **Automatic generation** via the `init-secrets` job.
- **In-Helm values** for simple, inline secret definitions.
- **External secrets** for integration with existing secret management systems.

A key feature of the chart is the **init-secrets job**, which automatically generates and stores secrets in a Kubernetes secret named `generated-secrets`. This simplifies the setup of sensitive configurations without manual intervention. This is supported only for secrets internal to the system.

The chart also supports **custom secret configurations** via either inline values or existing Kubernetes secrets.

The init secrets job

The **init-secrets job** is a Kubernetes job that runs once during the chart deployment as a helm `pre-install/pre-upgrade` hook to generate a secret named `generated-secrets`. This secret contains the necessary keys and configurations for the ESS Pro components.

Permissions Required: To ensure the job can create the `generated-secrets` secret, the Kubernetes user must have **permissions to manage RBAC** in the target namespace. The helm chart will create a service account with appropriate RBAC roles.

Use Case: This job is ideal for automated secret generation, especially when deploying the ESS Pro for the first time. It avoids manual configuration of sensitive data and ensures consistency across deployments.

If you do not want to use the job, you will have to set :

```
initSecrets:
  enabled: false
```

The chart will then require all secrets to be defined in the values.

Configuring secrets using in-Helm values

You can directly define secrets in the `values.yaml` file using the secret `value` field. This is useful for simple configurations or when secrets are not stored externally.

Example :

```
synapse:
  registrationSharedSecret:
    value: "your-secret-value"
```

This method is not considered the safest are the secrets will be stored in clear text in the `values.yaml` file.

Configuring secrets using external secrets

For advanced use cases, you can reference **existing Kubernetes secrets** to inject values into the ESS Pro components. This is ideal when secrets are managed elsewhere (e.g., in a secret management system).

Example :

```
synapse:
  registrationSharedSecret:
    secret: existing-secret
    secretKey: key-in-secret
```

Requirements:

- The `existing-secret` must be a valid Kubernetes secret containing the key `key-in-secret`.
- This method allows seamless integration with external secret management tools (e.g., HashiCorp Vault, Azure Key Vault).

Use Case: This approach is preferred when secrets need to be rotated or managed externally, ensuring compliance with security policies and reducing the risk of hardcoded credentials.

Revision #4
Created 2025-05-21 15:43:49 UTC by Gaël Goinvic

Updated 2025-05-28 11:51:16 UTC by Gaël Goinvic