Genelec

Documentation

Genelec Cloud Service Status

Date: December 18, 2023

Version: 1.1

Table of Contents

1. Introduction

The Ingress Monitor software stands as a vital tool for both users and administrators within a cluster environment, providing holistic insights into ingress and empowering efficient management.

2. Importance and Problems Addressed

Importance:

The software's significance lies in its consolidated view of all ingresses within a local cluster, granting users immediate visibility into each ingress's status. The ability to monitor ingresses at a glance and assess their health, along with detailed information presented in a convenient manner, facilitates quick decision-making and proactive measures.

Problems Addressed:

- **Ingress Visibility:** Prior to this software, tracking ingresses and their statuses was fragmented, making it challenging for users to oversee their health collectively. This software solves this by offering a centralized platform for monitoring.
- **Ease of Access:** Accessing vital information like website URLs, preview pictures, certificate validity, and last updated details was time-consuming. The software solves this by consolidating this information into an easily accessible popup.
- **Communication and Control:** For administrators, communicating crucial messages to end users and managing ingress settings were tedious tasks. The software addresses this by providing a dedicated admin interface for streamlined actions, such as publishing messages, managing ingress display settings, and session management.

3. Software Architecture

Overview:

The software architecture diagram provides a visual representation of the system's structure, illustrating the components and their relationships within the Ingress Monitor software.

Description:

The architecture is designed as a multi-tier system comprising three primary components: the frontend, backend, and database layers. Each layer plays a crucial role in delivering seamless functionality and data management.

Components:

- **Frontend Layer:** This layer serves as the user interface, facilitating user interaction with the application via web browsers. It retrieves data from the backend layer for display and interacts with users through the graphical interface.
- **Backend Layer:** Responsible for data processing and retrieval, the backend interacts with the local cluster environment to fetch ingress data. It processes and stores this data into the database layer for efficient access by the frontend.
- **Database Layer:** Utilizing a NoSQL MongoDB database, this layer stores all ingress-related information fetched by the backend. The database acts as a centralized repository, enabling quick retrieval and management of ingress data.

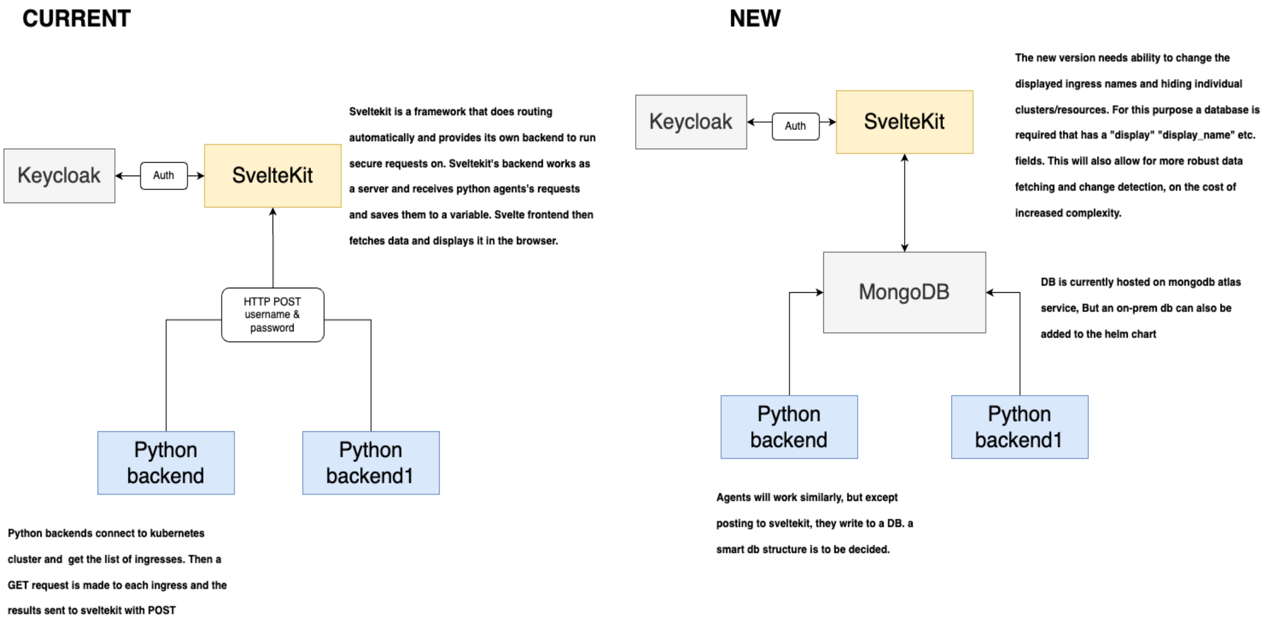
Interactions:

The front-end layer interacts with the backend through API calls, fetching data for display while the backend retrieves, processes, and stores ingress data into the MongoDB database. The database layer serves as the central hub for storing and managing all ingress-related information.

Benefits of the Architecture:

- **Scalability:** The architecture allows for easy scaling of individual components to accommodate increased data or user demand.
- **Modularity:** Each layer operates independently, promoting easier maintenance and updates without affecting the entire system.
- **Efficiency:** The system architecture ensures efficient data retrieval and presentation, offering users real-time access to updated ingress information.

Software Architecture Diagram:



Sveltekit is a framework that does routing automatically and provides its own backend to run secure requests on. Sveltekit's backend works as a server and receives python agents's requests and saves them to a variable. Svelte frontend then fetches data and displays it in the browser.

Python backends connect to kubernetes cluster and get the list of ingresses. Then a GET request is made to each ingress and the results sent to sveltekit with POST

The new version needs ability to change the displayed ingress names and hiding individual clusters/resources. For this purpose a database is required that has a "display" "display_name" etc. fields. This will also allow for more robust data fetching and change detection, on the cost of increased complexity.

DB is currently hosted on mongodb atlas service, But an on-prem db can also be added to the helm chart

Agents will work similarly, but except posting to sveltekit, they write to a DB. a smart db structure is to be decided.

This visual representation illustrates the interaction between the frontend, backend, and database layers, highlighting their interdependencies and functions within the Ingress Monitor software.

4. User Features and Functionality

- ### Overview

  The software offers users a comprehensive view of available ingresses, with intuitive visual indicators for quick status assessment.

- ### Ingress Monitoring

  Users can easily monitor the status of each ingress, marked by color-coded icons and text representations (OK, Warning, Error). The Ingress Monitor software utilizes color-coded icons, text representations, and corresponding HTTP status codes to convey the health status of ingresses within the system:

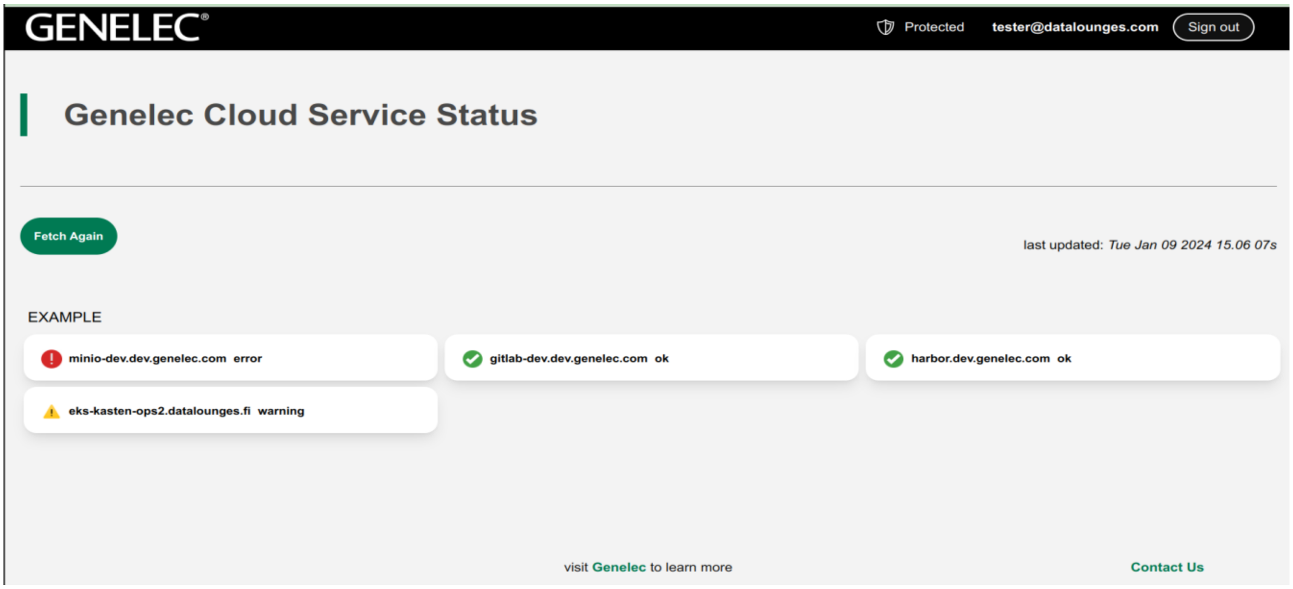
  - **OK Status:**
  - **Icon:** Green Arrow ✔
  - **Text Representation:** "OK" or "Healthy"
  - **HTTP Status Code:** 200 OK
  - **Description:** Indicates that the server has successfully handled the request and returned the appropriate content. The ingress is functioning correctly without any issues.

- **Warning Status:**
- **Icon:** Yellow Triangle ⚠
- **Text Representation:** "Warning" or "Issue"
- **HTTP Status Code:** 400 Bad Request or 500 Internal Server Error
- **Description:** Could represent various issues, such as a malformed request (400) or internal server problems (500). It indicates potential concerns or warnings with the ingress.

- **Error Status:**
- **Icon:** Red Error Sign ✕
- **Text Representation:** "Error" or "Critical"
- **HTTP Status Code:** 404 Not Found or 503 Service Unavailable
- **Description:** Indicates that the requested resource is not found (404) or the server is temporarily unavailable (503). It signifies critical errors or issues with the ingress.

These color-coded icons, text representations, and their associated HTTP status codes offer a quick visual and descriptive indication of the status of each ingress, allowing users to promptly assess their health and functionality within the Ingress Monitor software.

This section provides a clear breakdown of the status indicators used in the software, bridging the color-coded representations, text descriptions, and their underlying HTTP status codes for better understanding and reference.



Genelec Cloud Service Status

last updated: *Tue Jan 09 2024 15.06 07s*

EXAMPLE

! minio-dev.dev.genelec.com error

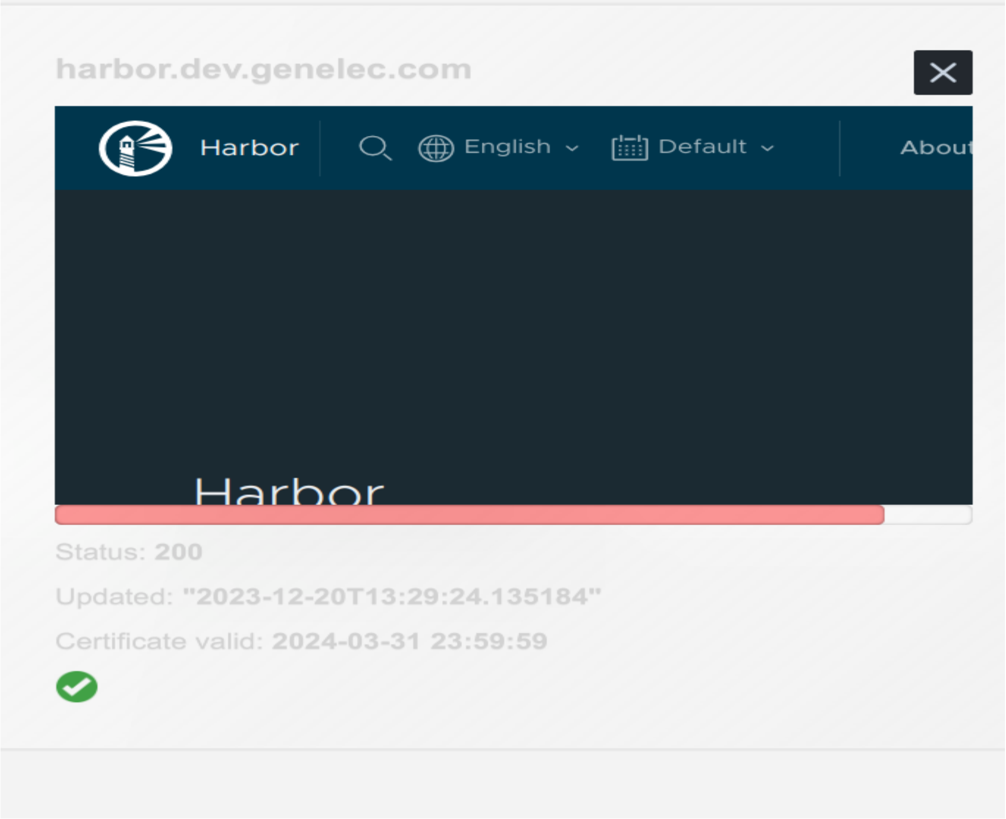
✓ gitlab-dev.dev.genelec.com ok

✓ harbor.dev.genelec.com ok

⚠ eks-kasten-ops2.datalounges.fi warning

- **Ingress Details Popup**

  Detailed information including website URLs, preview pictures, latest updated details, and certificate validity is conveniently available via a popup for each ingress.



harbor.dev.genelec.com

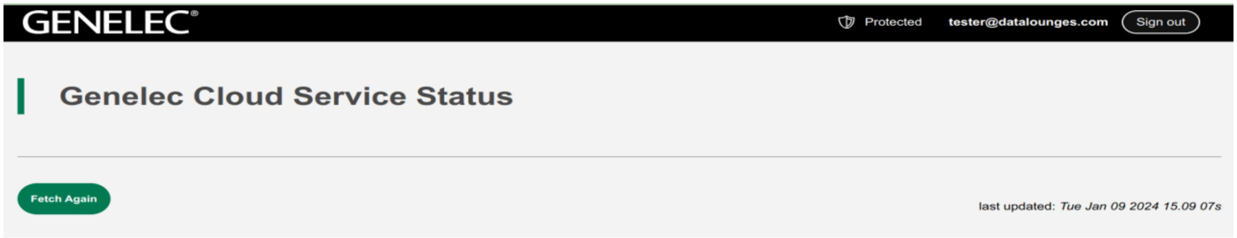
Status: 200

Updated: "2023-12-20T13:29:24.135184"

Certificate valid: 2024-03-31 23:59:59

- **Last Updated Information**

  Users can fetch updated ingress lists and view the timestamp of the last update, ensuring the displayed information is current.

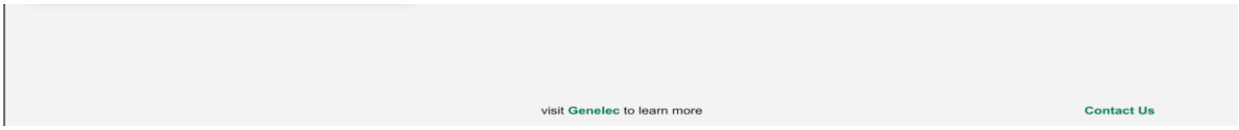


Genelec Cloud Service Status

last updated: *Tue Jan 09 2024 15.09 07s*

- **External Links**

  Users can swiftly access external links such as the company homepage and support contact for additional assistance.



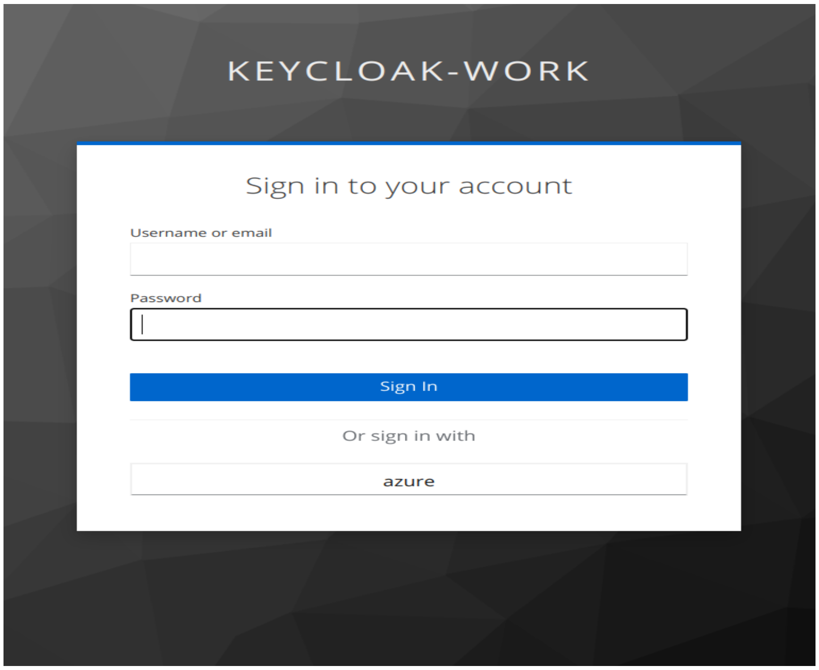
5. Admin Features and Functionality

- ### Overview

Administrators can access additional functionalities for managing ingresses and communicating with end users efficiently.

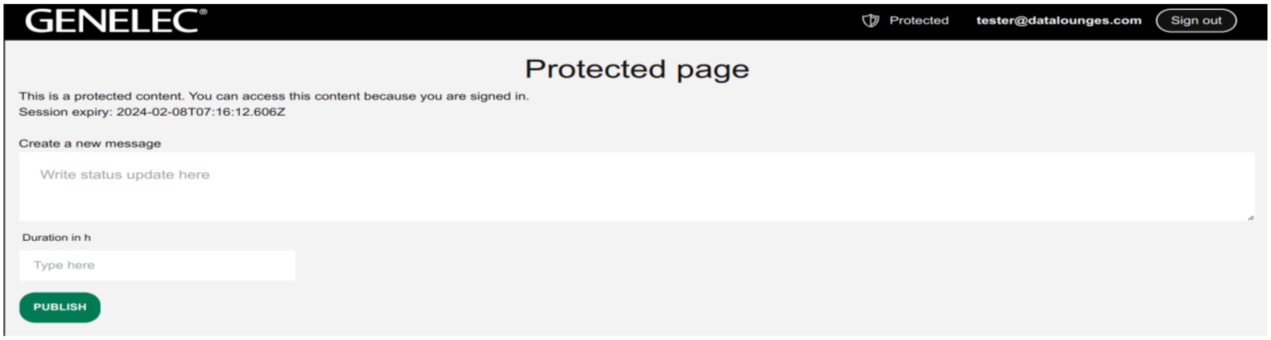
- ### Admin Login and Actions

Upon successful login, admins can publish messages for end users, manage ingress settings, and view session expiry details.



- ### Publishing Messages for End Users

Admins can publish time-bound messages for end users, which display on the homepage with countdowns and remain visible in the admin section for historical reference.



Protected page

This is a protected content. You can access this content because you are signed in.
Session expiry: 2024-02-08T07:16:12.606Z

Create a new message

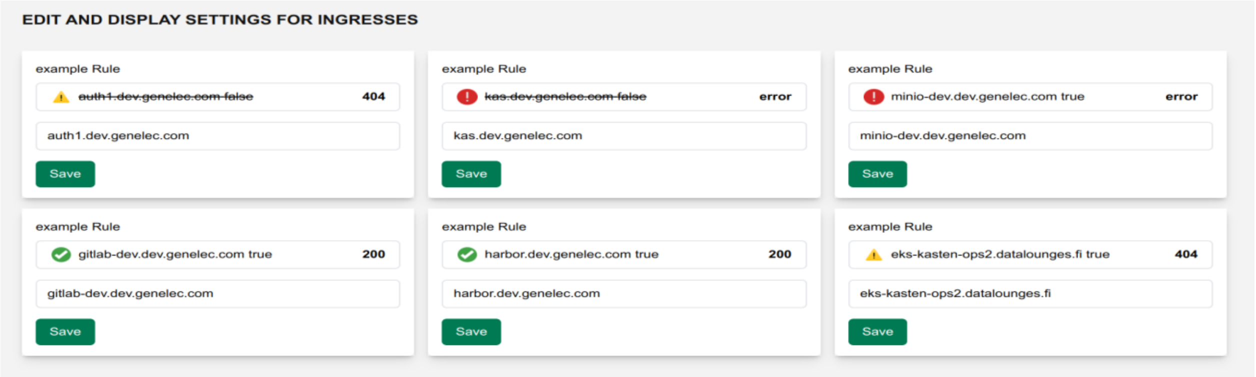
Write status update here

Duration in h

Type here

- **Edit and Display Settings for Ingresses**

  Admins can modify ingress display settings, rename ingresses, and control which ingresses are visible to end users, offering a tailored experience.



EDIT AND DISPLAY SETTINGS FOR INGRESSES

example Rule
⚠ ~~auth1.dev.genelec.com false~~ 404
auth1.dev.genelec.com

example Rule
! ~~kas.dev.genelec.com false~~ error
kas.dev.genelec.com

example Rule
! minio-dev.dev.genelec.com true error
minio-dev.dev.genelec.com

example Rule
✓ gitlab-dev.dev.genelec.com true 200
gitlab-dev.dev.genelec.com

example Rule
✓ harbor.dev.genelec.com true 200
harbor.dev.genelec.com

example Rule
⚠ eks-kasten-ops2.datalounges.fi true 404
eks-kasten-ops2.datalounges.fi

- **Session Management**

  Admins can view and manage session expiration details, ensuring secure access and controlled user sessions.

6. Installation Guide

System Requirements:

- **Hardware:**

  Minimum 2GB RAM, 1GHz processor.

  Disk space: 200MB for installation, additional space for database storage.
- **Software:**

  Operating Systems: Windows 7 or later, macOS 10.12 or later, Linux distributions.
- **Browser Requirements:**

Google Chrome, Firefox, or any modern web browser with JavaScript enabled.

Dependencies:

- **Frontend and Backend Deployment:** Deployed using Helm configuration.
- **Components:**

Frontend and Backend: Docker images pushed to Genelec Harbor.

Database: MongoDB for storing ingress data.

Security: Admin login is configured using genelec Keycloak configuration.

Step-by-Step Installation (Web Application):

- **Accessing the Web Application:**

  Open a compatible web browser (e.g., Google Chrome, Firefox).

  Enter the provided URL or IP address hosting the application.

- **Backend Deployment:**

  Pull the backend Docker image from Genelec Harbor.

  Run the backend using Helm configuration to fetch available ingress data and store it in the MongoDB database.

- **Frontend Deployment:**

  Pull the front-end Docker image from Genelec Harbor.

  Deploy the frontend to access the stored data from the MongoDB database.

- **Accessing the Web Application:**

  After deployment, navigate to the provided URL or IP address to access the Ingress Monitor web application.

Troubleshooting Tips:

- **Database Connection Issues:** Ensure the backend is properly configured to connect and store data in the MongoDB database.
- **Access Problems:** Verify network connectivity and firewall settings to access the application.

7. Configuration

Backend Configuration:

- The backend needs to be initially run to fetch and store ingress data into the MongoDB database.
- To update the database with new ingresses, re-run the backend component.

Frontend Configuration:

- No specific configuration required post-deployment.
- Frontend fetches data from the MongoDB database populated by the backend.

Additional Notes:

- For ongoing updates, schedule periodic runs of the backend to fetch and update the database with any new ingresses.
- Ensure proper network connectivity between the frontend, backend, and the MongoDB database for seamless data retrieval.

8. Best Practices and Tips

Best Practices:

- **Regular Backend Runs:** Schedule periodic backend runs to ensure the database stays updated with the latest ingress information.
- **Security Measures:** Implement access controls and secure configurations in keycloak for admin logins to maintain data integrity.
- **Backup and Restore:** Regularly backup the database to prevent data loss and have a restoration plan in place if needed.
- **User Training:** Conduct user training sessions to familiarize users with the application's features and functionalities.

Tips:

- **Browser Compatibility:** Ensure users utilize compatible browsers (e.g., Chrome, Firefox) for optimal performance.
- **Monitoring Routine:** Encourage users to frequently check the last updated information to ensure they are viewing the most recent data.
- **Admin Responsibilities:** Advise administrators to use messaging features judiciously and ensure timely removal of outdated messages.

9. Conclusion

The Ingress Monitor software stands as a pivotal solution in offering comprehensive ingress monitoring capabilities. With its user-friendly interface and administrator functionalities, it addresses the critical need for efficient management and visibility within cluster environments.

Conclusion Notes:

The software's ability to consolidate ingress monitoring, provide detailed ingress data, and offer essential administrative controls marks it as a vital tool for ensuring cluster health and streamlined communication between administrators and end users. Its deployment across various operating systems and browsers ensures accessibility and ease of use for all users.