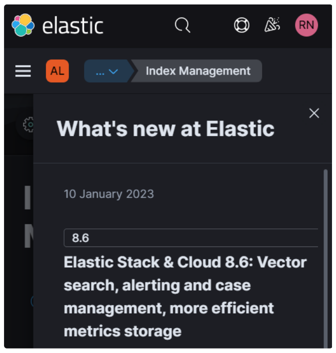
Change and Release Notifications

Introduction to the goals and the concepts

This following guide is meant for developers working on an app available on the Hogwarts's Analytical platform. Several applications such as **Elasticsearch** have implemented an in-application news feed detailing their updates and changes to their platform. In an effort to increase the visibility of the changes made to our applications, the notification feature allows application owners. This functionality was developed by the Assemblyline team. Therefore, most of the examples are taken from the Assemblyline's implementation of the Notification Feed.



What's new at Elastic

10 January 2023

8.6

Elastic Stack & Cloud 8.6: Vector search, alerting and case management, more efficient metrics storage

The Notification Feed

1. Creating the Feed file

The **Notification Feed** used on the Hogwarts's Analytical Platform follows the JSON Feed format. It is similar to other web feed such as RSS and Atom, but it was chosen was chosen because it is far easier to read, write and parse as opposed to the XML format.

JSON Feed Version 1.1

by Brent Simmons and Manton Reece

JSON Feed landing page

A TypeScript implementation of the The JSON Feed format can be found in the Template-UI common components. It provides all the fields a user might need for their news feed. The root type is the `Feed` that contains all the information about the feed and has the property *items* listing all the news feed element.

The `FeedItem` details the information about a single feed element. The *content* properties is where users write the content of their news feed. It supports three syntaxes that will be parsed accordingly in your application. Note that it checks each property sequentially to get the first value to display. Meaning, if markdown has content, only the markdown content will be displayed.

- **Markdown** : The `content_md` key allows users to write the feed using the markdown syntax.
- **HTML** : The `content_html` key allows for HTML, but it must be encoded or escaped for security.
- **Text** : The `content_text` key is simple text with no syntax.

Also note that if more properties are needed that don't follow the JSON Feed format such as the `_isNew` key in the `FeedItem`, start the name with an underscore to note the additional property.

The following models are the main JSON Feed types implemented in TypeScript.

```
type Feed = {
  version: string;
```

```
type FeedItem = {
  id: string;
```

```
type FeedAuthor = {
  name?: string;
  title: string;
  home_page_url?: string;
  feed_url?: string;
  description?: string;
  user_comment?: string;
  next_url?: string;
  icon?: string;
  favicon?: string;
  authors?: Array<FeedAuthor>;
  language?: string;
  expired?: boolean;
  hubs?: Array<{ type: string; url: string }>;
  items: Array<FeedItem>;
};
```

```
  url?: string;
  external_url?: string;
  title?: string;
  content_html?: string;
  content_text?: string;
  content_md?: string;
  summary?: string;
  image?: string;
  banner_image?: string;
  date_published?: Date;
  date_modified?: Date;
  authors?: Array<FeedAuthor>;
  tags?: Array<'new' | 'current' | 'dev' | 'servi
  language?: string;
  attachments?: Array<FeedAttachment>;
  _isNew: boolean;
};
```

```
  url?: string;
  avatar?: string;
};
```

The following is an example of a JSON Feed file.

```
{
  "version": "https://jsonfeed.org/version/1.1",
  "title": "Hogwarts development release news feed",
  "user_comment": "This feed provides news related to Hogwarts development releases",
  "description": "This feed provides news related to Hogwarts development releases",
  "language": "EN",
  "expired": false,
  "feed_url": "https://github.com/CybercentreCanada/hogwarts-discovery-apps",
  "home_page_url": "https://github.com/CybercentreCanada?q=hogwarts",
  "items": [
    {
      "id": "tag:github.com,2008:Repository/318279246/v1.1.1.dev1",
      "title": "Hogwarts 1.1.1.1 (DEV)",
      "date_modified": "2023-01-01T00:00:00Z",
      "date_published": "2023-01-01T00:00:0Z",
      "url": "https://github.com/CybercentreCanada/hogwarts-tui/releases/tag/v1.1.1_1",
      "tags": ["dev"],
      "content_html": "&lt;h3&gt;feature&lt;/h3&gt;\n&lt;ul&gt;\n&lt;li&gt;Added the notification feature to the Hogwarts landing page.&lt;/li&gt;\n&lt;/ul&gt;",
      "content_md": "### feature\r\n- Added the notification feature to the Hogwarts landing page.",
      "authors": [
        {
          "name": "cccs-author",
          "url": "https://github.com/cccs-author",
          "avatar": "https://avatars.githubusercontent.com/u/"
        }
      ]
    }
  ]
}
```

2. Hosting the files

Once the file has been created, it needs to be hosted on a platform that is accessible by any web client. The Notification component will issue a fetch request to access the file using its URL path. Therefore, the service providing access to the file must have the appropriate configuration to give access to that file. The following are some of the options available for a developer.

2.1 Azure Blob Service

The first option is to use the **Azure blob service** to serve the file. The Assemblyline News Feed uses it to serve its multiple feed files. It was chosen for its ease to setup and how it allows for automatic updating using a python script.

2.2 GitHub Repository

The second option is to use the Hogwarts Discovery Service's GitHub Repository to serve the feed files. That GitHub repository uses a tool called Flux CD to deploy automatically to a Eureka service. That service provides access to the files in the repository in their given path using the corresponding link to the appropriate environment. As such, you can make a Pull Request for that repository that includes your feed files and the service will serves those files accordingly.

2.3 Other Services

In short, it doesn't matter which service is being chosen as long as the file can be access by the web client. As such, that service must have the

where the file is located as long as the service serving the file is configure for all networks.

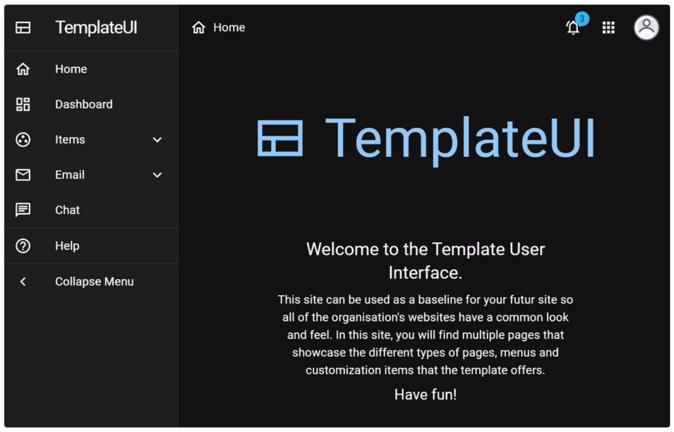
3. Updating the Hogwarts landing page and include the Notification component in your app

Once your feed file is created and accessible, you need to add the notification component in your web application and update the Hogwarts's landing page to include your feed.

3.1 Template-UI Notification Component

The Notification components in the Template-UI library are composed of the Notification button located at the top of the TopNav AppBar and the Drawer to display the notifications themselves. By default, the component gets its feed URLS from the preference configuration, but can be changed as needed. Note that the Notification button will only be displayed if an array of URLs is provided. The component is does not need other components to work. It will fetch, parse and display the notifications accordingly.

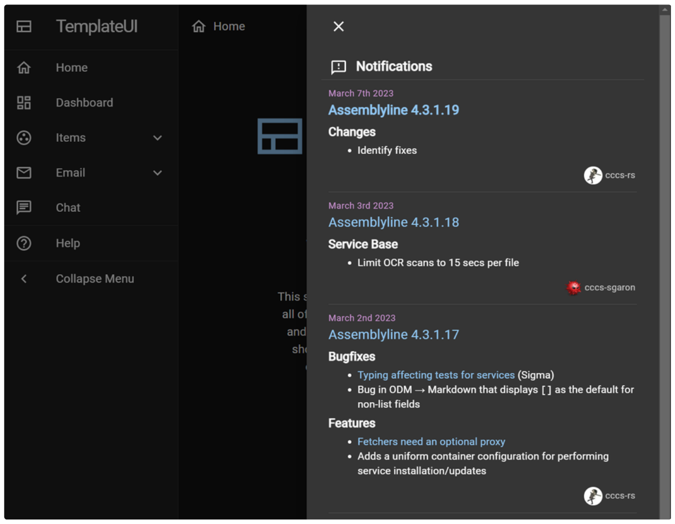
You can update your application to include the Notification components by adding these components in your common folder, calling the notification in the AppBar component and providing your list of notification URLs.



Welcome to the Template User Interface.

This site can be used as a baseline for your futur site so all of the organisation's websites have a common look and feel. In this site, you will find multiple pages that showcase the different types of pages, menus and customization items that the template offers.

Have fun!



March 7th 2023
Assemblyline 4.3.1.19
Changes
- Identify fixes

cccs-rs

March 3rd 2023
Assemblyline 4.3.1.18
Service Base
- Limit OCR scans to 15 secs per file

cccs-sgaron

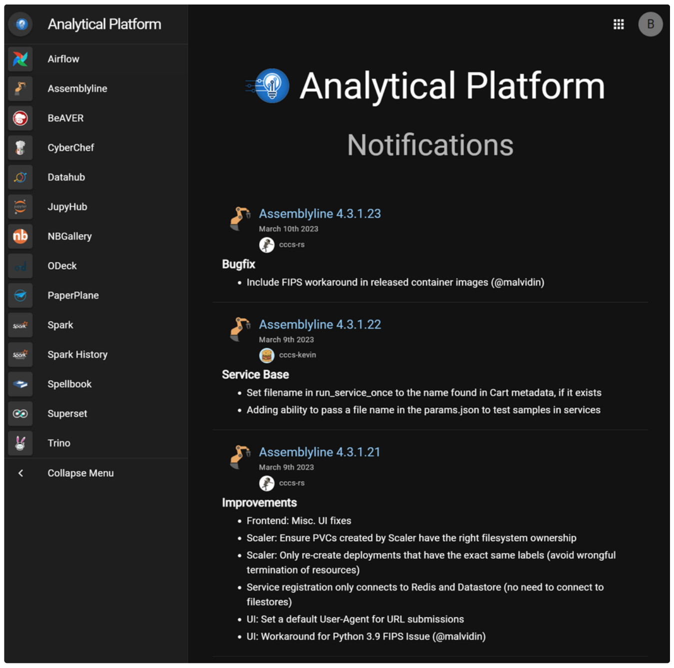
March 2nd 2023
Assemblyline 4.3.1.17
Bugfixes
- Typing affecting tests for services (Sigma)
- Bug in ODM → Markdown that displays [ ] as the default for non-list fields

Features
- Fetchers need an optional proxy
- Adds a uniform container configuration for performing service installation/updates

cccs-rs

3.2 Hogwarts's landing page

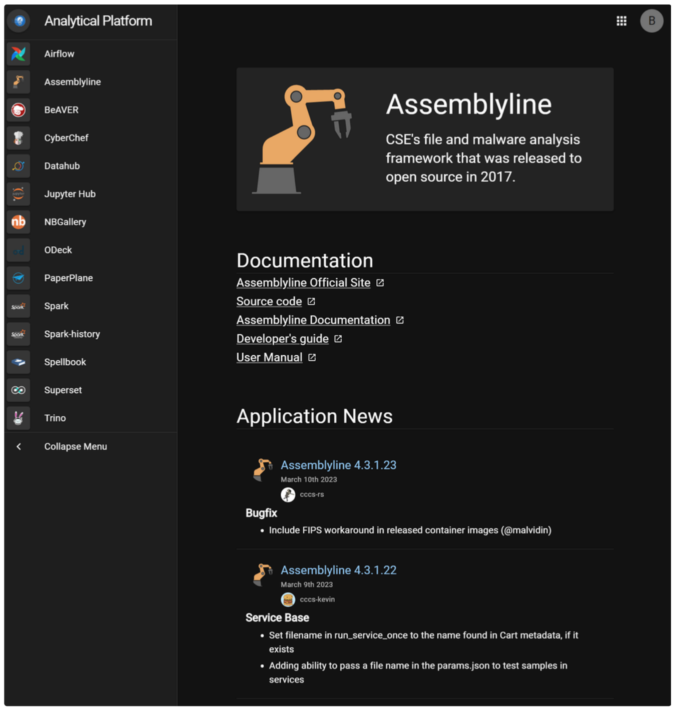
Once you have your Feed URLs, you need to update your application's page on the Hogwarts's landing site. The Hogwarts application will display all the notifications from all applications sorted chronologically. The specific application page will only display its notification.



Assemblyline 4.3.1.23
March 10th 2023
cccs-rs
Bugfix
- Include FIPS workaround in released container images (@malvidin)

Assemblyline 4.3.1.22
March 9th 2023
cccs-kevin
Service Base
- Set filename in run_service_once to the name found in Cart metadata, if it exists
- Adding ability to pass a file name in the params.json to test samples in services

Assemblyline 4.3.1.21
March 9th 2023
cccs-rs
Improvements
- Frontend: Misc. UI fixes
- Scaler: Ensure PVCs created by Scaler have the right filesystem ownership
- Scaler: Only re-create deployments that have the exact same labels (avoid wrongful termination of resources)
- Service registration only connects to Redis and Datastore (no need to connect to filestores)
- UI: Set a default User-Agent for URL submissions
- UI: Workaround for Python 3.9 FIPS Issue (@malvidin)



Assemblyline

CSE's file and malware analysis framework that was released to open source in 2017.

Documentation
Assemblyline Official Site
Source code
Assemblyline Documentation
Developer's guide
User Manual

Application News

Assemblyline 4.3.1.23
March 10th 2023
cccs-rs
Bugfix
- Include FIPS workaround in released container images (@malvidin)

Assemblyline 4.3.1.22
March 9th 2023
cccs-kevin
Service Base
- Set filename in run_service_once to the name found in Cart metadata, if it exists
- Adding ability to pass a file name in the params.json to test samples in services

4. Updating the file

During the lifecycle of your application, you will want to update your notification feed to better represent the changes made to your application and keep everyone aware about those changes. Currently, there is manual and an automatic solution to updating your feed files, but you could provide your own method.

4.1 Using GitHub Pull Request

If you have chosen to store your files on GitHub, you can manually update your files by creating a Pull Requests with your changes. Like mentioned before, once the FluxCD system detects a change in the repository, it will update the Discovery service with those changes.

4.2 Using a Spellbook Job

On Assemblyline, the process to update the feed files was automated using a Spellbook Job that runs each morning. Since the feeds files are located in an Azure blob storage, we can use the Azure storage library to connect to the client and to upload a new blob.

It is important to note that on Assemblyline this automated approach is only possible due to the member's contribution to its release process. Whenever a member wants to patch the application with their update, they create a new release on the main Assemblyline repository and document the content of their changes. This step is critical for our release process as it triggers the CD/CI system to create a new version of Assemblyline with the changes. However, for the Notifications, we can use GitHub’s API to fetch the release data to get all the necessary information to create the feed files. Note that the API’s release route is only available for public repository.

If your application does not have standard release procedure that documents the content of each update, you might want to use the GitHub’s manual approach to update the feed files or look into having a documented release process.

Summary

In conclusion, this guide explains the process of integrating the release notification in your application and in the Hogwarts’s landing page.