ETL solutions project

Created by:

Robert Fox

Xuanyu Lin

Lihong Zheng

Muhammad Nawaz

Shreya Jain

Here were some of the solutions that were considered:

- **Airbyte**: there was a free trial for this, but the service itself was paid-only.
- **Matillion**: here there was not a free option either, just a trial.
- **Fivetran**: again here there was no free or community version, only a free trial.
- **Apache Airflow**: this looked a good option – however there was some coding involved in use
- **Apache Nifi**: this appeared an interesting option, since it was both completely open-source, and involved no coding in use, though some system configuration was involved.

Ultimately Nifi was agreed as a good choice for the above reasons. It has proved initially to be fairly tough to learn the config setting, though once this is understood it seems an effective and user-friendly tool.

Key Features:

- Flow management
- Ease of use
- Security
- Extensible architecture
- Flexible scaling model

Installation:

Here are the docs: Getting Started with Apache NiFi

In brief:

- Installation on a local machine involved downloading and extracting a zipfile, and opening the extracted folder in an IDE such as VSCode.
- The appropriate Java Runtime Environment version needed to be also installed.
- A .bat file started the server, and a username and password were automatically generated, which needed to be retrieved from the logs in the IDE.
- Once running, the platform GUI can be accessed in the browser via localhost:8443/nifi/

It was discovered that Nifi can source data directly from a variety of endpoints. The team experimented with using http apis, locally stored files, NoSQL databases and AWS s3 to provide data.

Similarly, it was found that data can be outputted to any of these interfaces.

Fish-market platform:

A platform was built using an AWS s3 bucket as a source. A csv was chosen, 'fish-market.csv' that contained attributes of examples of various types of fish. Columns include fish species, weight, height and length.

Since Nifi 2.0 does not allow for importing of templates, a previous version was used, to allow exploration of pre-configured methods of converting csv to JSON format.

The platform effectively took the csv data and converted it to JSON that could be easily insert into a NoSQL datastore. The data was cleaned, configuring Nifi to ignore a record that gave a zero weight for a fish.

MongoDB was used as a final store for the cleaned and formatted data, which could be easily accessed, row-per-object, in the Mongo collection.

Live bitcoin data:

Another simple platform was built, to request bitcoin data from *coindesk.com*. This data is updated by the host every 30 seconds, so it provided a good opportunity to explore live updating using Nifi. The data landed in MongoDB as a single JSON object.

The scheduler on the InvokeHTTP processor was set to 10 secs, and the PutMongo to 'update'. The Mongo object was thus continuously updated automatically in real-time.

Additional Features:

- Lends well to visual creation and management of directed graphs of processors
- Is inherently asynchronous which allows for very high throughput and natural buffering even as processing and flow rates fluctuate
- Provides a highly concurrent model without a developer having to worry about the typical complexities of concurrency
- Promotes the development of cohesive and loosely coupled components which can then be reused in other contexts and promotes testable units
- The resource constrained connections make critical functions such as back-pressure and pressure release very natural and intuitive
- Error handling becomes as natural as the happy-path rather than a coarse grained catch-all
- The points at which data enters and exits the system as well as how it flows through are well understood and easily tracked

Github Compatibility:

In general, while NiFi does not have direct, built-in GitHub integration for flow management, the platform's compatibility with GitHub is achieved through utilizing GitHub as a version control system and leveraging GitHub-based tools for automation and deployment. This setup allows you to manage NiFi flows and configurations effectively within GitHub's ecosystem.

Security

- **System to System**

  A dataflow is only as good as it is secure. NiFi at every point in a dataflow offers secure exchange through the use of protocols with encryption such as 2-way SSL. In addition NiFi enables the flow to encrypt and decrypt content and use shared-keys or other mechanisms on either side of the sender/recipient equation.

- **User to System**

  NiFi enables 2-Way SSL authentication and provides pluggable authorization so that it can properly control a user’s access and at particular levels (read-only, dataflow manager, admin). If a user enters a sensitive property like a password into the flow, it is immediately encrypted server side and never again exposed on the client side even in its encrypted form.

- **Multi-tenant Authorization**

  The authority level of a given dataflow applies to each component, allowing the admin user to have fine grained level of access control. This means each NiFi cluster is capable of handling the requirements of one or more organizations. Compared to isolated topologies, multi-tenant authorization enables a self-service model for dataflow management, allowing each team or organization to manage flows with a full awareness of the rest of the flow, to which they do not have access.

Apache NiFi provides a strong set of security features that make it suitable for handling sensitive data in motion. Properly configuring and maintaining these security features will help you secure your NiFi instance and the data flowing through it.

Conclusion:

Apache Nifi was investigated by the team as a viable ETL platform solution. Being open-source and highly versatile, it was agreed that it would make an effective tool for collecting and preparing data for business analysis.

Potential use-cases might be scheduled sourcing of live data from an api, regularly updating a datastore, which in turn could be outputted to visualisation tools. The scheduler in the Nifi Fetch processor can be configured to access the data, at any desired time-interval.