GUIDE FOR INTEGRATION METHODS WITH THE DRUPAL RESTFUL API

Content

Integrating the Hadoop Distributed File System (HDFS) with RESTful APIs, including specific applications like Drupal API, involves various approaches that facilitate bidirectional data and command flows between HDFS and RESTful services, leveraging the capabilities of both systems. Below is a comprehensive list of such integration techniques:

1. WebHDFS REST API

WebHDFS provides a RESTful interface to HDFS, allowing user applications to interact with HDFS in a cluster via HTTP operations. This native support can be utilized for integration with any RESTful API, including Drupal API, enabling direct HTTP calls from Drupal to HDFS for file operations.

- **Functionality**: WebHDFS supports multiple operations achievable over HTTP, such as CREATE, ADD, READ, WRITE, DELETE, LIST, and SET PERMISSIONS. For example, reading data from HDFS involves a GET request, while creating or appending data uses a PUT request.
- **Architecture:** When a client sends an HTTP request to the WebHDFS REST API, the NameNode responds with a redirection to the DataNode containing the requested data, ensuring that data traffic does not burden the NameNode.

Advantages:

- Easy integration with web applications as it utilizes standard HTTP protocols.
- Allows HDFS to be accessible from a wide variety of client languages and systems that support HTTP.
- Requires no complex setup, as it comes as part of the Hadoop ecosystem.

Disadvantages:

- HTTP overhead can be significant, especially for large data transfers.
- WebHDFS REST API may not offer the same performance as native HDFS clients due to the stateless nature of HTTP.
- Security needs to be managed at the HTTP layer, and Hadoop's own security mechanisms like Kerberos may not always directly apply.

2. HttpFS

HttpFS is a service that provides a REST HTTP gateway supporting all HDFS file operations. It acts as a proxy server for HDFS, enabling interaction from remote hosts. This capability is particularly useful when integration with RESTful APIs needs to occur outside the Hadoop cluster network.

HttpFS offers similar functionality to WebHDFS regarding file system operations it

supports. However, it serves as a proxy server for the entire Hadoop file system API, not just HDFS, supporting other file systems integrated with Hadoop.

Architecture: HttpFS operates as a standalone service that proxies requests to the Hadoop cluster. It is designed to serve as a gateway for remote hosts to interact with HDFS and requires separate installation and maintenance.

Advantages:

- Allows access to HDFS outside the Hadoop cluster network, facilitating remote interactions.
- As a proxy, it can serve as a single access point, simplifying client-side configurations.
- Supports other file systems compatible with Hadoop, providing a more generalized solution.

Disadvantages:

- Additional overhead for maintaining a separate service solely for HTTP access to HDFS.
- Potential increase in response times due to the additional HTTP proxy layer between the client and HDFS.
- Scales horizontally to a certain extent but can become a bottleneck if not properly sized and configured for high-demand environments.

HttpFS thus serves as a crucial component in scenarios where remote access to HDFS via HTTP is necessary, offering flexibility and extending Hadoop's capabilities beyond the cluster boundaries while requiring careful consideration of its deployment and scalability implications.

3. Custom Wrapper for RESTful Services

A specifically developed service can act as an intermediary layer between HDFS and external APIs. This wrapper can translate RESTful calls into HDFS operations using the Hadoop Java API. It can also authenticate and redirect requests coming from the Drupal API to HDFS.

This service directly interacts with HDFS using the Java API, providing a REST interface capable of handling complex logic, input processing, and delivering customized outputs. It is highly adaptable and can be designed to expose only the necessary HDFS operations.

Architecture: Typically deployed as a standalone web application, this service can be hosted on a web server such as Apache Tomcat or Jetty. It acts as a mediator between HDFS and external REST APIs, processing HTTP requests, handling them, and executing corresponding HDFS operations via the Java API.

Advantages:

- Can be tailored to exact specifications, offering maximum flexibility in terms of functionality.
- Enables better optimization and control over interactions between HDFS and REST clients.
- Security logic and data validation can be custom-built to meet specific requirements.

Disadvantages:

- Development and maintenance can be resource-intensive, requiring dedicated time and expertise.
- Responsibility for security and performance optimizations lies entirely with the custom development team.
- Custom development means the solution is less generic, potentially leading to challenges in future scalability or integration with other systems.

This custom wrapper approach provides a powerful solution for integrating HDFS with RESTful APIs, offering precise control and customization capabilities, albeit requiring careful planning and ongoing support to ensure optimal performance and security.

4. Hadoop Connectors/Plugins for Drupal

Specific modules or plugins can be developed for Drupal that utilize the Hadoop API to communicate with HDFS. These plugins can handle authentication, reading, writing, and processing of data stored in HDFS. These modules provide Drupal-centric ways to interact with HDFS, such as storing files, importing/exporting data, and managing HDFS data through the Drupal interface. The exact features depend on the specific module or plugin being used.

Architecture: Connectors are typically PHP modules or libraries that use the Hadoop Java API or WebHDFS via PHP scripts. They are integrated into the Drupal architecture and managed as part of the Drupal ecosystem.

Advantages:

- Seamless integration with Drupal, providing a familiar interface for Drupal users.
- Reduced development time since connectors are pre-built for specific functionalities.
- Enables Drupal authentication and access control permissions to HDFS data.

Disadvantages:

- Functionality is limited to what the module or plugin offers; customization may be constrained.
- Relies on the community or third parties for module maintenance and updates.
- Potential performance challenges if plugins are not optimized for large-scale

HDFS operations.

Using Hadoop connectors or plugins within Drupal offers an efficient way to leverage HDFS capabilities directly from within the Drupal environment. While it simplifies integration and provides Drupal-specific functionality, careful consideration of customization needs and performance optimization is necessary to ensure smooth operation with HDFS.

5. Apache Knox

Apache Knox serves as a REST API Gateway for interacting with HDFS and other Hadoop services, providing a unified access point while enforcing security measures. Drupal systems can integrate with HDFS through Knox, which proxies RESTful API requests to HDFS and other services within the Hadoop cluster.

Knox offers a unified layer for RESTful access to Hadoop services, simplifying security for users who interact with services like HDFS, YARN, and Hive through a common gateway using REST API.

Architecture: Deployed as a standalone server, Knox sits at the edge of the Hadoop cluster and provides services such as authentication, authorization, auditing, and URL rewriting. It can proxy requests to various Hadoop services, including HDFS, thereby abstracting the complexity of direct HDFS integration.

Advantages:

- Centralized security enforcement facilitates access control management and security policy implementation.
- Simplifies client interactions with Hadoop services by abstracting complexities behind the REST API.
- Provides a more secure entry point to HDFS by integrating with enterprise-level security features.

Disadvantages:

- Requires additional infrastructure and configuration, adding complexity to the Hadoop environment.
- As a proxy, it may introduce additional latency in data access.
- The added layer necessitates monitoring and management, potentially increasing operational overhead.

Apache Knox enables Drupal systems to securely and efficiently interact with HDFS and

other Hadoop services through a standardized REST API interface. While it simplifies security and access management, careful consideration of its deployment and operational implications is crucial to maximizing its benefits within a Hadoop ecosystem.

6. Integrating HCatalog with Drupal

HCatalog provides access to HDFS data through its REST API. While primarily designed for accessing metadata in Hive, it can also be utilized for abstract file operations on HDFS. Drupal API can interact with the REST interface of HCatalog, indirectly enabling interactions with HDFS.

WebHCat (formerly Templeton), the REST API of HCatalog, offers RESTful interfaces to HDFS, Hive, Pig, and MapReduce. This simplifies access to HDFS data without needing to directly interact with Hadoop file system commands.

Architecture: WebHCat translates RESTful requests into HCatalog commands, which then interact with the Hadoop ecosystem. This provides a higher level of abstraction for accessing HDFS data, allowing users to work with tabular structures rather than files and directories.

Advantages:

- Higher level of abstraction makes it convenient for users familiar with SQL and tabular structures.
- Integrates well with Hive, enabling complex data operations such as queries and data management.
- Suitable for workflows involving data storage components.

Disadvantages:

- Not a direct method for performing file operations on HDFS; limited to the abstraction provided by HCatalog.
- Primarily designed for use with Hive, and its direct usefulness for raw HDFS operations may be limited.
- Adds an additional layer of complexity and potential failure points in the data access path.

HCatalog's integration with Drupal through its REST API (WebHCat) offers a streamlined approach to accessing HDFS data, particularly suitable for users comfortable with SQL and tabular data structures. However, its reliance on HCatalog and Hive means it may not directly address all use cases requiring raw file operations on HDFS. Careful consideration of its abstraction level and its fit within the overall architecture of data workflows is essential for maximizing its benefits.

7. Using OData (Open Data Protocol)

OData is a standard REST API protocol that can be used to expose HDFS as an OData service. Drupal or other RESTful API clients can then interact with HDFS data using standard OData queries.

8. File Transfer Mechanisms

Although not a direct API integration, RESTful services can interact with HDFS through file transfer mechanisms like SFTP, where files are uploaded to a staging area and then moved to HDFS using backend scripts or cron jobs.

9. Middleware Integration Platforms

Middleware platforms like Apache Camel, MuleSoft, or Talend can provide a bridge between RESTful APIs and HDFS. These platforms can orchestrate data flows, perform transformations, and mediate between different protocols and services.

10.Data Streaming Adapters

Tools like Apache Flume or Apache NiFi can be configured to stream data from RESTful API endpoints directly into HDFS. They can also handle data from HDFS to consumption by RESTful services, acting as continuous data pipelines.

11. HDFS clients with REST support

Development of HDFS clients that can be integrated into Drupal and support RESTful operations. These clients will translate Drupal API calls into HDFS actions using the WebHDFS REST API.

Each of these approaches varies in complexity, performance implications, and access control levels. The choice of a specific integration method will depend on several factors, such as security requirements, data volume and speed, as well as specific use cases involving the HDFS-RESTful API interaction. Moreover, when dealing with sensitive data, integration should ensure compliance with data management standards and personal privacy, which can impact the design and implementation of the integration layer.

The following table provides a summary of the pros and cons of each approach for integrating HDFS with RESTful APIs, including the Drupal API:

Approach	Advantages	Disadvantages
WebHDFS REST API	Native HDFS support; Direct HTTP access to HDFS	Limited by HDFS security features; HTTP network overhead
HttpFS	Suitable for remote access; Supports all HDFS operations	An additional layer can add latency; Requires separate management
A custom wrapper for RESTful services	Flexible, customizable features; Can be optimized for specific use cases	Development and maintenance overhead; Possible security risks if not implemented correctly
Hadoop Connectors/Plugins for Drupal	Seamless integration with Drupal; Drupal-oriented management features of HDFS	May not cover all; It depends on the plugin community for updates
Apache Knox	Security features like authentication; Unified API gateway for Hadoop services	Complex setup; Additional management component
HCatalog integration	Simplifies access to HDFS data; Good for Hive related operations	Not a direct way to interact with HDFS; Better suited for Hive than raw HDFS
Using OData	Standard protocol widely supported; Enables flexible requests	OData service layer overhead; Can introduce a performance hit for large datasets
Mechanisms for file transfer	Easy to implement; Works with existing infrastructure	Batch-oriented, not real-time; Requires manual or scripted file processing
Middleware Integration Platforms	Powerful data integration features; Can handle complex data streams	It can be expensive; Often require specialized knowledge
Data streaming adapters	Real-time data integration; Flexible and scalable	Setup can be complicated; May be too much for simple use cases
HDFS clients with REST support	Drupal-compliant interactions; Can use WebHDFS directly	Custom client development costs; It needs updates to stay in sync with HDFS changes

Each of these integration strategies offers a unique set of advantages and disadvantages, and the optimal choice will depend on the specific requirements and limitations of the system in question. It is imperative to consider factors such as expected data volume, latency requirements, security policies, and the level of control required for data operations when choosing the appropriate approach for integrating HDFS with a RESTful API.

Reference

1. WebHDFS REST API; https://hadoop.apache.org/docs/r1.0.4/webhdfs.html
2. WebHDFS FileSystem APIs; 2022; https://learn.microsoft.com/en- us/rest/api/datalakestore/webhdfs-filesystem-apis
3. Hadoop HDFS over HTTP-Documentation Sets; https://hadoop.apache.org/docs/stable/hadoop-hdfs-httpfs/index.html
4. Apache Knox Gateway 2.0.x User's Guide; https://knox.apache.org/books/knox-2-0-0/user-guide.html
5. Walker Rowe; What is Apache HCatalog? HCatalog Explained; 2017; https://www.bmc.com/blogs/what-is-apache-hcatalog-hcatalog-expla


EURO²



EuroHPC
Joint Undertaking