ORNL/TM-165912

Research Software Engineering Efforts for DataFlow: FY2021 Developments

Marshall McDonnell
Addi Malviya-Thakur
Greg Shutt
Suhas Somnath
Dale Stansberry
Olga Kuchar

September 2021

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT-BATTELLE LLC FOR THE US DEPARTMENT OF ENERGY

DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Website: www.osti.gov/

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone: 703-605-6000 (1-800-553-6847)
TDD: 703-487-4639
Fax: 703-605-6900
E-mail: info@ntis.gov
Website: http://classic.ntis.gov/

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone: 865-576-8401
Fax: 865-576-5728
E-mail: report@osti.gov
Website: http://www.osti.gov/contact.html

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-165912

Research Software Engineering Efforts for DataFlow: FY2021 Developments

Computer Science and Mathematics Division

Marshall McDonnell

Addi Malviya-Thakur

National Center for Computational Sciences

Greg Shutt

Suhas Somnath

Dale Stansberry

Olga Kuchar

September 2021

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AWS	Amazon Web Services
CADES	Compute and Data Environment for Science
CADES-OR	Compute and Data Environment for Science - Open Research
CI	Continuous Integration
CNMS	Center for Nanophase Materials Sciences
EC2	Elastic Compute Cloud
GCS	Globus Connect Server
GCSv4	Globus Connect Server version 4
GCSv5	Globus Connect Server version 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaC	Infrastructure-as-Code
JSON	JavaScript Object Notation
ORNL	Oak Ridge National Laboratory
UI	User Interface

ABSTRACT

DataFlow is a web application that helps scientific data to flow from one source location to another destination location. DataFlow helps scientists easily capture scientific metadata associated with an experiment and transmit both metadata and experimental data to a designated, centralized data storage resource. This report describes the software engineering efforts and architecture of the project for the fiscal year 2021 developments. We hope it effectively communicates findings from our work, challenges we have overcome, and how we will continue our development of DataFlow in the future.

1. Introduction

DataFlow is a web application that helps facilitate scientific data to flow from one source location to another destination location. DataFlow helps scientists easily capture scientific metadata associated with an experiment and transmit both metadata and experimental data to a designated, centralized data storage resource.

Currently, the target destination is the Compute and Data Environment for Science, or CADES (1), `$HOME` storage area for a User (2). This is accomplished by providing a simple web interface for Users to upload data, annotate with metadata, and then transfer that data from the source to the destination location.

One key use case for DataFlow is to help migrate data from off-network instruments to a destination that is both on-network and more easily accessible by a research team. Off-network refers to the fact that ORNL has an internal network not accessible to the outside internet but also to some machines physically within ORNL but are just not allowed on the ORNL network. This may be due to hardware that is running a required operating system that is not supported to run on the network or specific instrument hardware that cannot be managed by the current set of monitoring tools on the network. Thus, off-network implies that there is no connection to the ORNL "on-network" and all of the ORNL services (such as CADES OpenStack cloud or other compute infrastructure accessible only from within ORNL's network).

Moving data from off-network instruments to on-network locations is accomplished by deploying an instance of DataFlow that is accessible by the off-network instrument via HTTP / HTTPS but also to the on-network destination for the data transfer. Thus, a physical server running the DataFlow software is accessible by the off-network instruments via an approved HTTP / HTTPS port to the specific internet protocol (IP) address of this server only. The DataFlow server is on-network and has access to on-network ORNL services. The experimentalist at the off-network instrument can upload the experimental data, annotate with metadata, and have the datasets transferred to an on-network, destination location. From the destination location, the rest of the research team can perform data reduction, data analytics, or a variety of computational tasks on the data either manually or via automated processes. One note for this use case is that it does require a physical server to be installed in the off-network enclave for DataFlow to be deployed on, which requires an upfront cost to use the service.

This report describes the software engineering efforts and architecture of the project for the fiscal year 2021 developments. The current developments in this report cover the progress on adding new features to the core DataFlow service, initial work on containerization of DataFlow and its different services, GCS endpoint operations, new infrastructure for the project, and overall improvements to the DataFlow project.

2. Software Overview and Architecture

In Fig. 1, a high-level software architecture is presented for DataFlow. The core of DataFlow is a web application using the Ruby on Rails framework (3) that orchestrates the user interface interactions, upload of files, and transfer of files to their destination. Also included in the core application is a Phusion Passenger (4) application server and a NGINX (5) web server with capabilities for large file upload (6) to the application server. This core application communicates with multiple dependent services, such as a source location GCS endpoint, to initiate the transfer of data to the destination GCS endpoint location (7), a PostgreSQL database (8) for the core service data such as User credential storage, and a Solr database (9) for indexing of the data. The core application and all these services together are the DataFlow application as a whole.

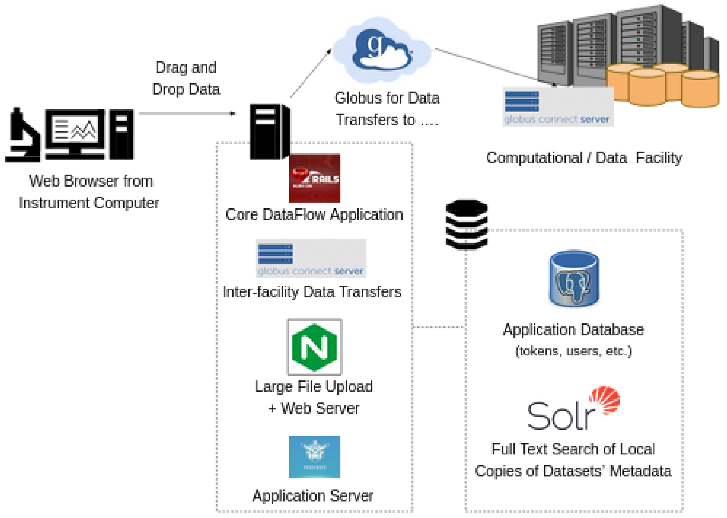
Drag and Drop Data
Globus for Data Transfers to ....
Computational / Data Facility
Web Browser from Instrument Computer
Core DataFlow Application
Inter-facility Data Transfers
Large File Upload + Web Server
Application Server
Application Database (tokens, users, etc.)
Solr
Full Text Search of Local Copies of Datasets' Metadata

Figure 1. Current software architecture of the DataFlow project.

3. Progress on Software Developments

3.1 DataFlow Service

DataFlow as a whole is made of up many different services as previously described. Yet, there is a core DataFlow service which is the Ruby on Rails (3) web application in charge of the overall orchestration of actions for DataFlow using the many different services. Below are the software developments specifically related to the DataFlow service.

3.2 New Features

Below is a list of new features introduced over the year for the DataFlow core service.

- Added the dataset name to the directory created on the destination location
- Corrected the metadata JavaScript Object Notation (JSON) file written to the destination

- Allowed the upload of an entire directory
- Sorted the view of the datasets in the User Interface (UI)
- Switched to human-readable file sizes in the UI
- Added a navigation bar for the UI
- Added an initial landing page to the UI instead of immediate redirect to Globus for authentication
- Documentation added to the core DataFlow service. Added to UI as a link so User can access at any time to help explain DataFlow's purpose and guide the User.
- Various other updates to the UI for design changes. Written in markdown and stored in the application code repository.

3.3 Globus Library Dependency

The core DataFlow service has a dependency on an "in-house" Ruby library for handling Globus interactions. While working on deploying a development version of DataFlow, the DataFlow core service showed authentication errors stating no credentials were provided during Globus transfers. It was clear they were being submitted in the request by the higher-level DataFlow core service code. Yet, it was found that authentication credentials were being "dropped" in the underlying Globus Ruby library. This authentication requirement was fixed in the underlying library and confirmed to work during manual integration testing.

This Globus library code repository is found at: https://code.ornl.gov/cades-gateway/globus

3.4 Globus Connect Server Service

Below are the software developments specifically related to the Globus Connect Server Version 4 (GCSv4) (10) endpoint setup as a required third-party service for data transfer in DataFlow.

3.4.1 Configurable Setup

The initial release of DataFlow was for the specific Gateway machine setup for the Center for Nanophase Materials Sciences (CNMS) User Facility (11) and sent data to the GCSv4 endpoint destination location at CADES Open Research (CADES-OR) (2). The code for DataFlow contained the hard-coded information for both the source (CNMS Gateway machine) and destination (CADES-OR) GCS endpoints.

These options were changed to configurable parameters specified at build-time and deployment of DataFlow. The data transfer controller and GCS endpoint controller required refactoring, along with additions to configuration files.

This allowed for the ability to change the GCS endpoints and, for future work, will help enable the following:

- Allow research teams outside of ORNL to use their own specified GCS endpoints for both source and destination.

- Set the source and destination endpoints at run-time via the UI.
- Change the endpoints for testing GCS endpoints setup for DataFlow integration testing.

3.4.2 Containerization

In efforts to containerize the entire DataFlow application, the GCS service was the only service besides the main application that did not have a readily available container image from DockerHub. Thus, we created a container image definition for this service. The target was the GCS version 4 (GCS v4) and not the current GCS version 5 (GCS v5) endpoint. From direct discussion with Globus team members on the Globus Google Groups forum, containerization for the GCSv5 is not yet available but is a work-in-progress currently (12).

When containerizing a GCS v4 endpoint, there were many challenges found that were non-trivial.

First, it was discovered that the `data_interface` in the GCS configuration file was not being set. This is not set because branching logic (`if` statement) in the Globus Python library was mislabeling the container to be an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance, flagging it as `True` for `is_ec2`. If the `[GridFTP]` section's `DataInterface` is explicitly set to the IP address of the instance, the default behavior of flagging as an EC2 instance is overridden.

This same issue arose in the DataFed setup of a new DataFed data repository. It was communicated across the projects how to provide this simple fix and it also remedied the problem.

Second, it was discovered that the MyProxy server (13) running inside the container for the GCSv4 endpoint was unable to be activated from the running application. Interestingly, the prescribed port for MyProxy was open between the container and the host so MyProxy network traffic should be able to connect to the service running in the container. However, only after setting up the container to run in shared network mode with the host could the MyProxy service connect to the DataFlow application in this container setup. This is achieved by adding the `--net=host` option when creating the Docker container.

Third, in some instances, like for the container setup, the endpoint will be generated as the application is spun up for each service. Yet, DataFlow needs to know the source endpoint ahead of time before it spins up as a service. Thus, on startup, the option was added to either use the `GLOBUS_SOURCE_ENDPOINT` if it is defined or to use the `globus-cli` tool to complete an OAuth request (14) to extract the new source endpoint UUID.

Fourth, it was found that pseudo-tty had to be added to the GCSv4 container during run-time.

4. Progress on Operations

4.1 Demonstration Service

In working with the ORNL Biosciences Division on a potential collaboration for their off-network instruments, a demonstration of the DataFlow application was requested. This was a fair point since, as noted previously, this use case does require physical hardware purchases and installation costs in the off-network enclave for DataFlow to be deployed on, which requires an upfront cost to use the service.

This led to the DataFlow team standing up a demonstration service site that would be available to all of ORNL to use. This would allow for demonstrating DataFlow to any team interested in this capability.

The website https://dataflow.ornl.gov was created as an ORNL-wide, demonstration deployment of DataFlow. This instance is running the CADES OpenStack cloud (15) on a virtual machine and has the destination of the data landing in the CADES `$HOME` directory (2) of the User. This also allows the DataFlow team to now maintain a "production-like" instance of DataFlow to get feedback on new features and developments.

4.2 New Staging Environment

To accommodate new features that are being developed by the DataFlow team, a staging environment is currently being set up at https://dataflow-staging.ornl.gov. This allows for the team to have a place for manual integration testing and quality assurance before rolling new features and improvements over to the production stage running at https://dataflow.ornl.gov, along with other deployed instances such as the CNMS DataFlow gateway.

This is still under active development to capture all the configuration management installation in code.

4.3 Documentation of a DataFlow Deployment

As both the new demonstration service and the staging environment were being set up, it was realized that even a manual deployment of DataFlow was not captured yet in documentation. A markdown document was created and added to the code repository for DataFlow to document each installation step of each service along with template configuration files. An original version of the document was created after the initial deployment of the demonstration service. However, the document is being actively revised and corrected as we now move to setting up the new staging environment. The document is under version control so collaborative updates and history of changes are being captured.

This deployment document is currently found in the code repository at:
https://code.ornl.gov/cades-gateway/cades-gateway/-/blob/dev/DEPLOY.md

4.4 Infrastructure Repositories

Over the last year, DataFlow has had lots of changes to its infrastructure. This includes the following:

- Development environment infrastructure was set up in CADES OpenStack cloud
- Development environment was used to test containerization so was configured with only container build and run-time dependencies (and not DataFlow dependencies)
- Testing destination GCS endpoint infrastructure was set up in CADES OpenStack cloud
- Development environment was switched to the DataFlow demonstration service and now has a "bare metal" deployment (non-containerized)
- Test destination endpoint was no longer being used so infrastructure was removed

- Staging environment infrastructure was set up in CADES OpenStack cloud

With each of these changes, errors can occur and there can be a desire to resurrect or rollback to a previous state of the DataFlow infrastructure. With software development, this is easy with version control since the history is preserved. Yet, with infrastructure, this is a bit more challenging for provisioning cloud compute assets from code and basically impossible with physical servers (for infrastructure provisioning, not configuration management changes). As seen above, aside from the off-network instrument deployments, there are lots of cloud compute deployments for the DataFlow project.

To capture the infrastructure needed for DataFlow on cloud compute resources, we have implemented infrastructure-as-code (IaC) (16). One of the leading technologies for this is the tool from HashiCorp called Terraform (17). Terraform allows for writing declarative programming modules for API interfaces and, for cloud computing APIs, to provision resources with that API.

With infrastructure-as-code, the infrastructure is captured as a "recipe" for the current development and operations teams as the "source of truth" for what infrastructure is currently running and provides version control for changes that are created. From the IaC, the software engineering tools or running continuous integration (CI) can be applied even to infrastructure code: linting of the code, testing to ensure it provisions infrastructure properly, and automated pipelines to use the IaC to deploy infrastructure.

HashiCorp's Terraform was used to write the IaC for DataFlow. The setup is currently not in a state where the entire DataFlow team can collaborate. Yet, as this capability matures with the team, it can be migrated to be such that any member of the team can manage the infrastructure state from IaC repositories that sit next to the software repositories.

The DataFlow IaC code repository is found at:
https://code.ornl.gov/cades-gateway/infrastructure/deployment

5. Other Notable Outcomes

Outside of the software development and operations of DataFlow, communications for DataFlow were prepared. These included a poster presentation and an oral presentation on DataFlow at ORNL events.

A poster on DataFlow was presented during the ORNL Software and Data Expo 2021 (18) in May. The presentation can be found in the RESolution publication system at:
https://resolution.ornl.gov/pub/preview/157959

An oral presentation and demonstration of DataFlow was given to the ORNL Data Assets Council on August 11th, 2021. An overview was provided along with a live demonstration of https://dataflow.ornl.gov and then an interactive poll was performed to get immediate feedback on the DataFlow service. The feedback was used in project management to prioritize features to support in the future.

6. Conclusion

The details included in the report cover the software engineering efforts and architecture of the project for the fiscal year 2021 developments. The current developments in this report cover the progress on adding new features to the core DataFlow service, initial work on containerization of DataFlow and its different services, GCS endpoint operations, new infrastructure for the project, and overall improvements to the DataFlow project.

Finally, we have noted future developments and planned features where necessary in the report. We plan to submit a follow-on report next year to also capture these feature developments, other developments of the project not captured in this report over the next year, and reasoning and decisions for the choices made.

7. REFERENCES

References

[1] ORNL, “CADES Website,” 2021. [Online]. Available: "https://cades.ornl.gov/"

[2] ——, “CADES SHPC Condos Documentation Website,” 2021. [Online]. Available: "https://docs.cades.ornl.gov/#condos/overview/"

[3] Rails, “Ruby on Rails Website,” 2021. [Online]. Available: "https://rubyonrails.org"

[4] Phusion Passenger, “Phusion Passenger Website,” 2021. [Online]. Available: "https://www.phusionpassenger.com/"

[5] NGINX, “NGINX Website,” 2021. [Online]. Available: "https://nginx.com"

[6] ——, “NGINX’s nginx-upload-module Website,” 2021. [Online]. Available: "https://www.nginx.com/resources/wiki/modules/upload/"

[7] Globus, “Globus Connect Server Documentation Website,” 2021. [Online]. Available: "https://www.globus.org/globus-connect-server"

[8] PostgreSQL, “PostgreSQL Website,” 2021. [Online]. Available: "https://www.postgresql.org/"

[9] Apache, “Solr Website,” 2021. [Online]. Available: "https://solr.apache.org/"

[10] Globus, “Globus Connect Server version 4 Documentation Website,” 2021. [Online]. Available: "https://www.globus.org/globus-connect-server/v4/"

[11] ORNL, “Center for Nanophase Materials Sciences Website,” 2021. [Online]. Available: "https://www.ornl.gov/facility/cnms"

[12] Globus, “Message in Feb. 2021 with Jason Alt of Globus on Globus Google Groups about GCSv5 containers,” 2021. [Online]. Available: "https://groups.google.com/a/globus.org/g/discuss/c/k3SYIX_NP0A/m/lcXfHxE3AgAJ"

[13] NCSA, “MyProxy Website,” 2019. [Online]. Available: "http://grid.ncsa.illinois.edu/myproxy/"

[14] OAuth, “OAuth Website,” 2021. [Online]. Available: "https://oauth.net/"

[15] ORNL, “CADES Cloud Documentation Website,” 2021. [Online]. Available: "https://docs.cades.ornl.gov/#openstack/about/cloud-overview/"

[16] C. Riley, “Version your Infrastructure,” 2015. [Online]. Available: "https://devops.com/version-your-infrastructure/"

[17] HashiCorp, “Terraform Website,” 2021. [Online]. Available: "https://www.terraform.io/"

[18] ORNL, “Software and Data Expo 2021,” 2021. [Online]. Available: "https://softwaredataexpo.ornl.gov/"