ORNL/TM-166131

# Research Software Engineering Efforts for DataFed: FY2021 Developments

Marshall McDonnell
Addi Malviya-Thakur
Dale Stansberry
Olga Kuchar

**September 2021**

**ORNL/TM-166131**

# Research Software Engineering Efforts for DataFed: FY2021 Developments

**Computer Science and Mathematics Division**

Marshall McDonnell

Addi Malviya-Thakur

**National Center for Computational Sciences**

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 . . . . . v
ABBREVIATIONS . . . . . vii
ABSTRACT . . . . . 1
1. Introduction . . . . . 1
2. Software Overview and Architecture . . . . . 1
3. Progress on Software Developments . . . . . 2
    3.1 Continuing Containerization . . . . . 2
    3.2 Documenting DataFed Data Repository Installation . . . . . 3
    3.3 Globus Connect Server Version 5 Testing . . . . . 4
4. Progress on Operations . . . . . 6
    4.1 Infrastructure Repositories . . . . . 6
5. Conclusion . . . . . 7
6. REFERENCES . . . . . 8

## LIST OF FIGURES

1 High-level software architecture of the DataFed project. . . . . . . . . . . . . . . . . . . . . . . . 2

## ABBREVIATIONS

| | |
|---|---|
| CADES | Compute and Data Environment for Science |
| CI | Continuous Integration |
| GCS | Globus Connect Server |
| GCSv4 | Globus Connect Server version 4 |
| GCSv5 | Globus Connect Server version 5 |
| IAC | Infrastructure-as-Code |
| ORNL | Oak Ridge National Laboratory |
| VM | Virtual Machine |

## ABSTRACT

DataFed is a scientific data management system for big data providing simple and uniform data access, organization, discovery, and sharing within and across scientific facilities - with the goals of enhancing productivity and scientific reproducibility. We hope to aid in communicating development contributions to DataFed over the last year and any planning we can provide for the future developments in the next year.

## 1. Introduction

DataFed (1; 2; 3; 4; 5) is a federated and scalable scientific data management and collaboration system that addresses the critical need for holistic and FAIR-principled (6; 7) "big data" handling within, and across, scientific domains and facilities with the goal of enhancing the productivity and reproducibility of data-oriented scientific research. DataFed supports the early life-cycle stages of "working" scientific data and serves as a tool to ease the burden associated with capturing, organizing, and sharing potentially large volumes of heterogeneous scientific data. DataFed provides an environment in which scientific data can be precisely controlled and refined in preparation for eventual data publishing. DataFed relies on Globus (8) for high-performance, reliable, and secure data transport between (and within) facilities.

This report describes the software engineering efforts and developments for the fiscal year 2021. The current developments in this report cover the progress on continued efforts to provide containerization of DataFed components, additional documentation of setting up DataFed data repositories, standing up Globus Connect Server (GCS) version 5 endpoints (9) for DataFed data repositories to explore integration issues, infrastructure additions related to development, and overall improvements to the DataFed project.

## 2. Software Overview and Architecture

In Fig. 1, a high-level software architecture is presented for DataFed. DataFed's central services include the "core" service which is essentially the "brains" of DataFed. The core service manages the metadata associated with managed raw data and also implements access control, orchestration, and concurrency controls for data movements across the DataFed network. The core service, however, is not directly involved in the transfer of raw data - this function is delegated to Globus services, (9) or more specifically, to the GridFTP servers (10; 11) (managed by Globus) (12) located at DataFed data repositories and other facilities.

The raw data storage resources within a DataFed data repository can be any form of physical storage hardware, so long as the interface to this storage is supported by the Globus Connect Server endpoint used. Currently this includes POSIX file systems and S3 object stores for Globus Connect Server version 4 (GCSv4) and for Globus Connect Server version 5 (GCSv5), also includes various object store types and archival storage systems. The inherit reliability of the physical storage of a repository is determined by the host facility and could range from inexpensive magnetic disks to high-speed solid state drives or even archival-quality geographically distribute storage systems. Local administrators control repository policies and determine which DataFed users can utilize a repository by granting (or revoking) repository allocations. These local administrative policies and actions have no impact on DataFed repositories at other facilities.

The web services within the DataFed central services primarily support a web portal that allows users to easily organize and share data from a web browser; however, the web services also play a critical role in authenticating DataFed users through Globus's federated identity system (which is based on OAuth2). (13) New DataFed users must register through the DataFed data portal and grant certain permission to DataFed through Globus's authorization system. These permissions relate to user identification and enabling automatic data transfers on behalf of DataFed users.

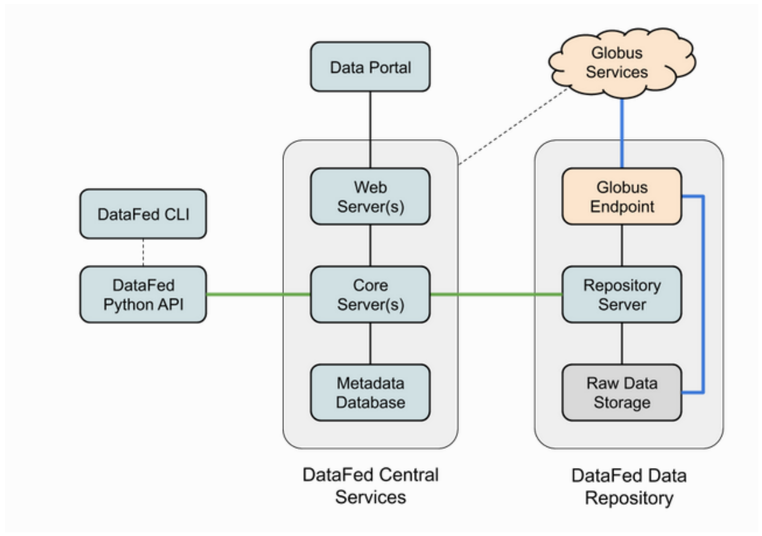

**Figure 1. High-level software architecture of the DataFed project.**

## 3. Progress on Software Developments

Below are details about some of the the progress on software engineering developments for DataFed.

### 3.1 Continuing Containerization

Previously, there were efforts put into containerizing (14) the components of DataFed in the last fiscal year. Docker (15) was used to define the Dockerfiles, or the container image definitions, and as the run-time tool to spin up the containers. These were added in the `dockerfiles` directory of the DataFed GitHub repository: (16)

https://github.com/ORNL/DataFed/tree/master/dockerfiles

The original goal for this effort was to help developers setup local development environments by spinning up containerized components of DataFed and also to help with future deployments if DataFed was switched to a containerized deployment. The latter has become a reality now that DataFed has begun to target the OLCF Slate service (17) as a deployment platform.

For the original DataFed containerization setup, the container images were split into two separate definitions. First, a "kickstart" image is defined that has all the necessary libraries and dependencies installed but no DataFed installation. This provides a read-to-use development container that DataFed can be installed into. Second, there is the DataFed image that defines the installation of all DataFed components into one container. The intention was to modify this container image definition for each individual component. The container image definitions were meant to be run sequentially, where the kickstart image was created and then used as the base for the DataFed component container images. The reasoning behind this was that the kickstart image has a long build time. Starting from the kickstart image as a base allows the DataFed component image creation to have a much smaller lead time to build. With container image layers, one could say that keeping both the kickstart and DataFed builds in one container image definition would work since the layers would be cached after they have been executed such that re-builds would be fast for just the DataFed code that changes. Yet, since DataFed has multiple components that would all use the same base installation of dependencies, there would be a large amount of replication and builds of individual DataFed components would all be re-building the same base when built separately. This goes against the "Don't Repeat Yourself" principle (18) when designing software, in that changes to the base would need to be carried out over multiple definitions instead of only one container image definition.

Progress this year included:

- Fix to the ZeroMQ dependency configuration in build system. The location for source and header file locations was mis-configured and not cross-platform. This was a subtle bug found at runtime.
- Updates to the directory structure inside the container in the container image definition. This is required since DataFed data repositories have specific locations for "collections" used to store files. The two required collections were for "Users" and "Projects".
- Added documentation to the `dockerfiles` for setting up a DataFed data repository. This details some of the required steps for an administrator of a data repository such as server configurations, generation of keys required by the DataFed core service, and specific commands that need to be run.
- Added the JSON and JSON schema dependencies to the container image definitions. This was more than a simple package manager installation for the container image definition since these dependencies must be configured and built from source.
- Additional documentation about the GridFTP authorization call-out output for a DataFed data repository setup. Globus "sits on top of" GridFTP and requires it for all transfers. DataFed interacts at this GridFTP level for authorization and thus has non-trivial setup requirements to integrate DataFed's own authorization library with GridFTP authorization call-outs to this library.

As seen, most work is related to documentation and standing up a DataFed data repository. This was the focus mainly due to the experimentation of setting up a GCS version 5 (GCSv5) endpoint and testing integration with DataFed as a data repository, as opposed to the current GCS version 4 (GCSv4) endpoints. More is discussed in Section 3.3 that discusses these efforts on setting up the GCSv5 endpoints for DataFed.

## 3.2 Documenting DataFed Data Repository Installation

From Section 3.1, two of the items listed in the the software developments list for containerization are related to documentation of data repositories. The main reason is there was a focus on setting up and running a DataFed data repository container with a pre-existing GCS endpoint running on the host machine (not running the GCS from a container). This is directly related to working on the integration of the GCSv5 endpoint for a DataFed data repository in Section 3.3.

Mainly, the following general information was captured in documentation setting up a DataFed data repository:

- The need to setup a port for communication between the DataFed data repository and DataFed core service. This requires both a port from the machine is made available and, if running a CADES OpenStack cloud instance for the data repository, opening up this port for the security group. The default port is 9000.
- The command via Docker to execute that spins up a container where commands can be run to stand up a DataFed data repository.
- How to generate keys required for the DataFed data repository to be registered in the DataFed core service.
- How to obtain the public DataFed core service keys that are required for the data repository.
- Directory structure and permissions required for the data repository.
- Information about the data repository that need to be manually sent to the DataFed core service administration to complete registration of the new DataFed data repository.
- Initial work to describe what is needed to integrate the DataFed data repository with the running GridFTP server on the host machine for the GCS endpoint. This is still a work-in-progress in the documentation.
- After it is registered, how to spin up the data repository.

The documentation of a data repository is still a work in progress, and is initially focused on capturing what is required to start setting up a DataFed data repository and is stored in the version control software (16). Going forward, this documentation will be moved to a data-repository-specific directory instead of with the containerization directory. This transition is motivated by the fact that the data repository can be created on bare metal, from a virtual image, and in a container using this documentation. Also, this information will be either replicated or directly used in the DataFed documentation page (19) as appropriate. At the time of writing, the DataFed data repository has started its documentation page at https://ornl.github.io/DataFed/admin/general.html#data-repository. Using this documentation, work towards automating the process of configuring a data repository should be achievable, given modifications to the underlying code (i.e. allow a data repository to be registered programmatically instead of via a manual step to send info to DataFed core admins, etc.).

## 3.3 Globus Connect Server Version 5 Testing

DataFed is currently able to use the GCSv4 (20) endpoints for the underlying Globus data transfers. The GCSv4 endpoints are considered "legacy" by Globus, but the paradigm for interacting with the API has changed significantly as appropriately denoted by the change in major version of the GCS software. When Globus removes these legacy GCSv4 endpoints, DataFed will no longer have a functioning data transfer layer and will not operate until the migration to GCSv5 has completed. Progress on this migration has begun, but is far from complete. The paradigm of interacting with Globus endpoints is very different between GCSv5 and GCSv4 endpoints. Both data transfer is still possible between both endpoint types as of today. Yet, an administrator of a GCSv5 endpoint (which is anyone standing up a DataFed data repository) will experience a completely different installation, setup, and maintenance for the endpoint compared to GCSv4 and also experience a "learning curve" on the new architecture. Examples of these differences include not having to stand up or point to a pre-existing authentication service such as MyProxy or CILogin but instead handle this directly with Globus, understand a completely different layered architecture and terminology for GCSv5 endpoints, have different ports that are open through firewalls due to the change in authentication and architecture, understand that GCSv5 only exposes a collection for an endpoint similar to a GCSv4 endpoint but that the server and roles are no longer available to configure, that some GCSv5 features are now only available via subscriptions, and not all of the features available for GCSv4 are available in GCSv5.

One task for the fiscal year was to begin investigating setting up GCSv5 endpoints as DataFed data repositories and trying to integrate with DataFed to see what changes, if any, would need to be made. To accomplish this, the following work was carried out:

- Research into the new architecture of GCSv5 endpoints along with reading the documentation about installs. (9)
- New network ports were allocated under the DataFed Resource Utilization Allocation (RUC) for the CADES OpenStack cloud. This RUC, titled `RUC_DATAFED`, was acquired in FY20 for, initially, DataFed but also is used for the DataFlow project at this time of writing.
- Working with the CADES OpenStack cloud team, these network ports were created with the following names, datafed-globus-endpoint-test01.ornl.gov and datafed-globus-endpoint-test01.ornl.gov
- Input firewall exception rules for the network ports for GCSv5 requirements into ORNL SAFer system. (21) Mainly, ingress traffic from "Internet" to "OPENRESEARCH-SYSTEMS" (for CADES OpenStack cloud) for TCP ports 50000-51000 (for data transfer) and 443 (for Globus API).
- Setup infrastructure definitions via Terraform for creating the instances that use these network ports. Read more in Section 4.1
- Registered the instances as GCSv5 endpoints in Globus at https://developers.globus.org/
- Installed GCSv5 onto the instances manually. This was manual since it was found out that the process requires manual input from the User installing the GCSv5 software. This is a big problem going forward for automation and containerizing the endpoint installs for DataFed data repositories.
- Tested just via Globus transfers to ensure the endpoints worked.
- Started to create DataFed data repositories out of these GCSv5 endpoints via containers (see section 3.1)
- Registered these DataFed data repositories running GCSv5 with the development environment DataFed core service (https://sdms.ornl.gov)
- Began to explore the GridFTP authorization integration with DataFed data repository library. Was not able to do data transfers from the data repository endpoints.
- Switched to a "bare metal" install of DataFed data repository instead of via a container to make the configuration less complex and simplify debugging efforst

- Left off at working directly with the DataFed data repository library, https://github.com/ORNL/DataFed/blob/master/repository/gridftp/authz/libauthz.c, to add logging and print-statement debugging since the issue exists at this level for why the transfers are unable to work.

Overall, the investigation of this effort is not complete but much progress has been made. Further exploration of the integration with GCSv5 and the GridFTP authorization call-out library is required to make more progress on this front. Once a working prototype test data repository is finished, documentation should be added for the setup of a DataFed data repository with a GCSv5 endpoint vs. a GCSv4 endpoint. Also, efforts from Section 3.1 should be re-visited. After a working GCSv5 endpoint is integrated with DataFed as a system install, follow-on efforts could revitalize containerization of the DataFed components and how to integrate them with a system-installed GCSv5 endpoint. However it has been indicated by Globus directly that container support is being actively worked on for GCSv5. (22) This has a direct impact on setting up DataFed data repositories. The containerization of the GCSv5 endpoint for a DataFed data repository should be pioneered by Globus first and then later adopted by DataFed for standing up a completely containerized DataFed data repository.

## 4. Progress on Operations

### 4.1 Infrastructure Repositories

With adding new GCSv5 (9) endpoints to test standing them up as DataFed data repositories, new infrastructure was needed. The Compute and Data Environment for Science (CADES) OpenStack cloud (23; 24) was used to stand up two GCSv5 test endpoints.

To capture the infrastructure needed for these new GCSv5 endpoints on the CADES cloud compute resources, infrastructure-as-code (IaC) was used. (25) IaC is to capture infrastructure in code, just like software, such as cloud server instances, security groups, and networking. One of the leading technologies for this is the tool from HashiCorp called Terraform. (26) Terraform is meant to be a tool used to create API interfaces and develop declarative programming modules 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 but 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 the new GCSv5 endpoint requirements. Mainly, the security groups, virtual machine (VM) instances, and configuration management scripting was captured via IaC. The setup is currently not in a state where the entire DataFed 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.

One large takeaway is that the GCSv5, unlike the GCSv4 (20) endpoints, require manual input from a User during provisioning. This is a very large drawback since it is a barrier to automation. This is true for system-installs and containers of GCSv5, as discussed in Section 3.3.

## 5. Conclusion

We have described the software engineering efforts and developments for the DataFed project for fiscal year 2021. The main goals for these efforts have been to iteratively improve DataFed and research new capabilities going forward. Details have been provided on progress on continued efforts to provide containerization of DataFed components, additional documentation of setting up DataFed data repositories, standing up GCSv5 endpoints (9) to test out, infrastructure additions related to development, and overall improvements to the DataFed project. We hope to aid in communicating findings from our work, challenges we have overcome, and how we will continue our development of DataFed in the future.

## 6. REFERENCES

### References

[1] D. Stansberry, S. Somnath, J. Breet, G. Shutt, and M. Shankar, “Datafed: Towards reproducible research via federated data management,” in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 1312–1317.

[2] D. Stansberry, 2020. [Online]. Available: "https://ornl.github.io/DataFed/index.html"

[3] OLCF, “Datafed - a federated scientific data management system,” 2020. [Online]. Available: "https://www.olcf.ornl.gov/olcf-resources/rd-project/datafed-a-federated-scientific-data-management-system/"

[4] A. Schneibel, “Data matters: Scientific data management tool improves labeling, information sharing for olcf researchers,” 2020. [Online]. Available: "https://www.olcf.ornl.gov/2020/11/18/data-matters-scientific-data-management-tool-improves-labeling-information-sharing-for-olcf-researchers/"

[5] D. Stansberry, “Datafed website,” 2021. [Online]. Available: "https://datafed.ornl.gov"

[6] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne *et al.*, “The fair guiding principles for scientific data management and stewardship,” *Scientific data*, vol. 3, 2016.

[7] GO-FAIR Initiative, “Go-fair website,” 2021. [Online]. Available: "https://www.go-fair.org/fair-principles/"

[8] Globus, “Globus website,” 2021. [Online]. Available: "https://www.globus.org"

[9] ——, “Globus connect server documentation website,” 2021. [Online]. Available: "https://www.globus.org/globus-connect-server"

[10] W. Allcock, “Gridftp: Protocol extensions to ftp for the grid,” 2003. [Online]. Available: "https://www.ogf.org/documents/GFD.20.pdf"

[11] I. Mandrichenko, “Gridftp protocol improvements,” 2003. [Online]. Available: "https://www.ogf.org/documents/GFD.21.pdf"

[12] W. Allcock, J. Bresnahan, R. Kettimuthu, and M. Link, “The globus striped gridftp framework and server,” in *SC ’05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, 2005, pp. 54–54.

[13] OAuth, “Oauth website,” 2021. [Online]. Available: "https://oauth.net/"

[14] Docker, “Use containers to build, share and run your applications,” 2021. [Online]. Available: "https://www.docker.com/resources/what-container"

[15] ——, “Docker website,” 2021. [Online]. Available: "https://www.docker.com/"

[16] D. Stansberry, “Datafed github,” 2021. [Online]. Available: "https://github.com/ORNL/DataFed"

[17] OLCF, “Olcf slate documentation website,” 2021. [Online]. Available: "https://docs.olcf.ornl.gov/services_and_applications/slate/index.html"

[18] Wikipeida, “Dry wikipedia page,” 2021. [Online]. Available: "https://en.wikipedia.org/wiki/Don’t_repeat_yourself"

[19] D. Stansberry, “Datafed documenation website,” 2021. [Online]. Available: "https://ornl.github.io/DataFed/index.html"

[20] Globus, “Globus connect server version 4 documentation website,” 2021. [Online]. Available: "https://www.globus.org/globus-connect-server/v4/"

[21] ORNL, “Ornl’s firewall rule exception safer website,” 2021. [Online]. Available: "https://safer.ornl.gov/"

[22] 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"

[23] ORNL, “Cades website,” 2021. [Online]. Available: "https://cades.ornl.gov/"

[24] ——, “Cades cloud documentation website,” 2021. [Online]. Available: "https://docs.cades.ornl.gov/#openstack/about/cloud-overview/"

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

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