Greyfish: An Out-of-the-Box, Reusable, Portable Cloud Storage Service

Carlos Redondo
carlos.red@utexas.edu
The University of Texas at Austin
Austin, Texas

Ritu Arora
rauta@tacc.utexas.edu
Texas Advanced Computing Center, The University of Texas at Austin
Austin, Texas

ABSTRACT

A scalable storage system is an integral requirement for supporting large-scale cloud computing jobs. The raw space on storage systems is made usable with the help of a software layer which is typically called a filesystem (e.g., Google's Cloud Filestore). In this paper, we present the design and implementation of an open-source and free cloud-based filesystem named as "Greyfish" that can be installed on the Virtual Machines (VMs) hosted on different cloud computing systems, such as Jetstream and Chameleon. Greyfish helps in: (1) storing files and directories for different user-accounts in a shared space on the cloud, (2) managing file-access permissions, and (3) purging files when needed. It is currently being used in the implementation of the Gateway-In-A-Box (GIB) project. A simplified version of Greyfish, known as Reef, is already in production in the BOINC@TACC project. Science gateway developers will find Greyfish useful for creating local filesystems that can be mounted in containers. By doing so, they can independently do quick installations of self-contained software solutions in development and test environments while mounting the filesystems on large-scale storage platforms in the production environments only.

KEYWORDS

cloud storage, containerization, file storage, filesystem

ACM Reference format:
Carlos Redondo and Ritu Arora. 2019. Greyfish: An Out-of-the-Box, Reusable, Portable Cloud Storage Service. In *Proceedings of Practice and Experience in Advanced Research Computing, Chicago, IL, USA, July 28-August 1, 2019 (PEARC '19),* 6 pages.
https://doi.org/10.1145/3332186.3333055

1 INTRODUCTION

Cloud computing systems are capable of providing scalable storage to their users. However, a software layer is required for managing the raw storage space (e.g., Google's Cloud Filestore [1]). In this paper, we will present the design and implementation of an open-source and free cloud-based file-storage service named as "Greyfish" that can work on different cloud computing platforms. The Greyfish file-storage service helps in storing files and directories, managing access permissions, and purging data when needed. It can be mounted as a volume [2] within a Docker [3] container and used to provide data-persistence service for interactive, multi-user, web applications that are built using containers. The data stored in Greyfish can be accessed from within a web application through Application Programming Interfaces (APIs). All Greyfish APIs are made to operate using a Web Server Gateway Interface (WSGI) and support multiple threads for multiple users.

The Greyfish service is already being used in the Gateway-In-a-Box (GIB) [4] project. GIB provides a basic out-of-the-box solution for creating small-scale science gateways that are equipped with browser-based terminal access. A simplied version of Greyfish, named Reef, is being used in the BOINC@TACC project [5]. The process to install and reuse Greyfish is simple - it takes about two minutes to complete the installation process, including base image download and building time. Furthermore, a Grafana [6] interface can be connected to Greyfish to visually monitor all the calls made to it from the web applications.

In this paper, we provide an overview of the Greyfish system design and its functionality in Section 2. The file storage model adopted in this project is described in Section 3, and an overview of the Greyfish installation process is presented in Section 4. An overview of each supported API is provided in section 5. We report the performance of file upload and download in Section 6. The current users and the possible use cases of Greyfish are briefly mentioned in Sections 7 and 8. The related work and future work are described in Sections 9 and 10, respectively.

2 DESIGN OVERVIEW

Greyfish [7] is built as a container-based file-storage (or filesystem) service for cloud computing platforms. It is composed of the following containers:

- **Main Container**: Contains the user files and the necessary APIs. Each new user is assigned a personal directory on the shared space. Nested sub-directories are also allowed. All user directories are a part of the docker-volume. This allows to restart the service and avoid data loss in case of upgrades, thereby, allowing Greyfish to be built in servers already hosting other web applications. It is built using the python3.6 alpine Docker image due to its small size.
  All user communication for file upload and/or download is done via the python APIs. These APIs are not the primary container processes, and as such, they can be stopped for updates without killing the main container. This process can

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07…$15.00
https://doi.org/10.1145/3332186.3333055

be repeated as many times as needed. All APIs are written in python3 and using gunicorn [8] as a Web Server Gateway Interface (WSGI) server that allows multiple thread calls. Four threads are used by default. The API will record the login status of a user, whether it is successful or not, and will log all failed login attempts.
Greyfish supports administrative purge APIs, designed to eliminate all files older than an arbitrary number of seconds.

- **Redis Container**: Redis [9] is an in-memory database used only for temporary tokens and caches. Temporary tokens can be granted for single actions for any user and/or system not entrusted with the main Greyfish credentials. They can be created by any system with Redis access, complete Greyfish access is therefore not required.
  A Temporary token is always assigned to one user, and is immediately deleted after first use. Tokens are therefore implemented as keys, and can be set as permanent (only being deleted after used) or only existing for a certain time-window. Note that Greyfish itself does not provide any specific APIs to create these tokens. Instead, this Redis container, that is open by default to outside connections, can be accessed with a password for automatically creating tokens for users.
- **InfluxDB Container**: InfluxDB [10] is a time-series database used for logs. All Greyfish actions are recorded and can be accessed at a later point in time. Each successful database transaction records the action, caller ID, caller IP, and specific information depending upon the action itself. All failed logins are registered as well, such as, client IP, reason for failure, and the account name. All actions are associated with a time-stamp, which is assigned when the action is completed.

The entire Greyfish ecosystem can be built from source using Docker Compose [11], with its only prerequisites being the availability of Docker and Docker Compose tools.

Greyfish user data and the InfluxDB logs are stored in Docker volumes, so that they persist even if a container stops, or is under maintenance.

Nonetheless, it is also possible to simply run Greyfish without temporary tokens and/or recording logs, using it only as a cloud storage functionality - this reduces the service to only the main container.

Although not required, a Grafana instance can also be connected to the InfluxDB database for data visualization. A complete overview of the Greyfish architecture is shown in Figure 1.

Greyfish supports multiple users, each being allowed a nested directory structure where each directory can contain files and/or other directories. This structure is supported directly by the APIs. Using the APIs, a file can be uploaded directly into a selected directory location. The directory locations are automatically created if they do not already exist at the time of file upload. Similarly, a user may download a file from any location within his main directory. The directory downloads are recursive and help in downloading all subdirectories within a directory that is selected for download.

The Greyfish APIs can then be called from any system having access to either the main Greyfish key or a single-use token.

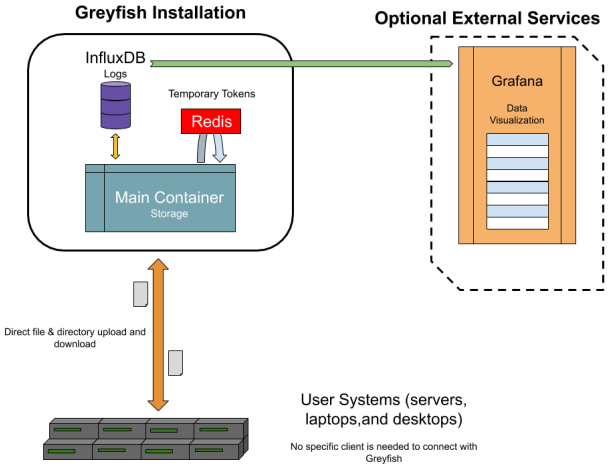

Figure 1: Overview of the Greyfish architecture

When a new user is added, Greyfish will automatically assign a new directory. All user files will be located within it. All uploaded files have their usernames 'sanitized' before storing them within the system, in order to avoid hidden commands within the file-names.

All InfluxDB calls within Greyfish itself are done via the official Python client. Greyfish does not execute any queries directly, and relies instead on the appropriate write and read client operations in order to avoid injections. Note that, if added, a Grafana instance will require more precautions for security when executing queries through it.

Furthermore, Greyfish also allows the purging of all the files older than a given time in seconds. All empty directories will be deleted as well after the purge is complete. For any of these purges, however, the main user directory will not be deleted. Additionally, deleting a user directory, as well as all its files, is also possible.

Greyfish does not provide a web-interface to access user data nor does it have an administrative one. Therefore, all API and system updates must be executed within the container over an SSH connection. However, Greyfish allows to scan all the user content within a directory (please see the *File Storage Model* in Section 3 for more information). This could allow an external web-interface to obtain the data from Greyfish. Nonetheless, for this to be possible, it would require access to the main Greyfish key or to the Redis container in order to assign itself tokens.

3 FILE STORAGE MODEL

As mentioned above, each Greyfish user is assigned a personal directory upon signing up for the application that uses Greyfish. Each personal directory is then recorded as *DIR_USERNAME*. That is, for the username "joe", the personal directory will be "DIR_joe". This signup can be done by any system with access to the main Greyfish key or with a temporary token, provided to a user already signed up. As a result, Greyfish signup should be done automatically from another system, and the main key should not be shared with the users directly.

All the inter-process communication for file uploads and downloads is done using the Python APIs. The path for upload/download is encoded directly into the URL, using "++" to specify "/" in the path, starting at the user's personal directory. Currently, there are no limitations on either the number of files and directories or the overall space occupied by a particular user by default.

Both text and binary file storage is possible. It is also possible to upload a directory in the form of a compressed tar file (*.tar.gz*, *.tgz* file endings), which is then uncompressed on the server, and the compressed tar file is deleted. All uploaded directories retain their original structure in the tar file.

As mentioned above, Greyfish supports a nested directory structure. Therefore, a file or directory can be uploaded directly to its target subdirectory, which is the path that is created recursively if it does not exist already. Greyfish does not support version control or multiple versions of the same file/directory. Therefore, if a path collision occurs, the previous version will be overwritten. Greyfish also does not optimize the internal data storage more than the default Alpine container already does.

Similar to the upload process, files can be downloaded from any subdirectory path. Directories can also be downloaded as compressed tar (*.tar.gz* file ending) files.

Furthermore, it is possible to download all data for a certain user at once in the form of a compressed tar file. It is also possible to replace all current user data in the same way. This system is ideal for transferring large datasets from other services for a certain user or project.

Greyfish APIs also support a scan functionality, which returns a JSON string containing all of the users' files and subdirectories in a nested format.

Finally, if a user account is deleted, the entire directory associated with the user's account will also be deleted. Greyfish does not pose any constraints on reusing the deleted user-ids when creating a new account.

All directories on Greyfish are stored within a docker volume as well. This volume is mapped to a host directory. Therefore, users' data can be accessed directly from the main server. Docker also allows other Docker containers to access the aforementioned volume if needed. However, all file changes and accesses that are made directly, either through the host machine or other containers, without using the Greyfish APIs are not recorded in the logs.

4 INSTALLATION

As mentioned above, a Greyfish system can be easily setup on any system with Git, Docker and Docker Compose installed. Then, the complete download and installation process - including the Docker build - can be completed in less than two minutes. However, this process can be longer for systems with low RAM and/or a poor network connection, due to the time-taken in Docker image downloads.

All the necessary passwords for Redis, InfluxDB, and Greyfish itself are setup as environmental variables in the startup scripts. If none are provided, default values are assigned.

By default, the Redis and InfluxDB containers are setup as password-protected, but with their ports being publicly available. This is done so that other systems can communicate with them. For example, an external Grafana instance will require that the InfluxDB ports are available.

A snippet of the steps for installing a complete Greyfish server are provided in Listing 1 to demonstrate that the setup process is easy to follow.

Table 1: Measuring Upload and Download Performance

File size (MB)	Upload time (s)	Download time (s)
1	0.327	0.078
50	2.573	0.082
100	4.704	0.093
150	6.995	0.093
200	8.459	0.093
250	10.697	0.096
300	12.605	0.100
350	14.594	0.102
400	17.825	0.106
450	18.611	0.109
500	20.828	0.111

5 SUPPORTED APIS AND FUNCTIONS

Greyfish is composed of the following APIs:

- **Upload and Download**: This is the main API. It supports the download and upload of files and directories. It creates the necessary paths if not present already.
- **Bulk Download**: This API supports downloading all user files and directories as a *.tar.gz* file.
- **Bulk Upload**: Uploads new contents to a user directory, deleting all its current contents in the process.
- **New User Creation and Deletion**: This API helps in creating new user accounts and deleting existing accounts on the filesystem.
- **Administrative features**: Supports getting a list of all users and file purges.

As mentioned above, all the API calls and actions are automatically logged in the InfluxDB database.

6 PERFORMANCE

Greyfish provides a set of Python files to test download and upload speeds, making use of the python module requests. These tests are measured from a client perspective, rather than the server.

A complete test, checking for both upload and download speeds, was done on a Greyfish system mounted on a Jetstream [12] [13] VM with 30 GB of RAM and 60 GB disk, with a similar virtual machine as the client. All files used for it were created using random strings of a fixed length in MBs. The results are graphically presented in Figure 2. Table 1 includes some of the data-points from this performance test to improve the readability.

The graphs shown in Figure 2 indicate that the file-upload time for a Greyfish system is mostly linear with respect to the file size. Nonetheless, the download time is much smaller as compared to the

Listing 1: Complete setup of a Greyfish server

```
git clone https://github.com/noderod/greyfish
cd greyfish
# Change the influxdb log credentials
vi credentials.yml
# Set the appropriate passwords and base URL (without / and http(s)://
# Define the number of threads using "greyfish_threads", default is set to 4
REDIS_AUTH="examplepass" URL_BASE=example.com greyfish_key="examplegrey" docker-compose up -d

# Activate the necessary APIs
# Enter container
docker exec -it greyfish_greyfish_1 bash
cd /grey
# Start the needed databases and assign permission (APIs will not be started)
/grey/setup.sh
# Activate (change the number of threads if needed, standard is 4)
./API_Daemon.sh -up
```

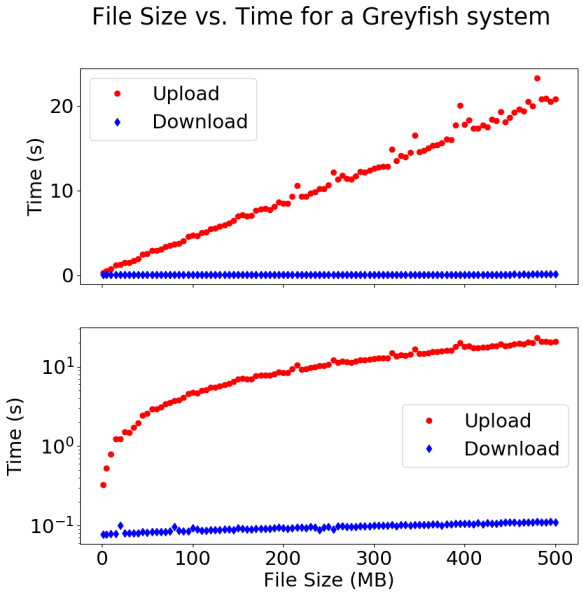

Figure 2: Linear and Y-logarithmic plots for upload and download time for a Greyfish service.

upload time and is almost constant for different file sizes considered in this experiment. Of course, with much larger file sizes, there will be a significant variation in download time.

7 CURRENT USERS

The GIB framework uses Greyfish for creating a shared, multi-user filesystem. The BOINC@TACC project is using a basic version of Greyfish - called Reef - which does not provide logs and subdirectories. Reef saves the files submitted by the researchers using the job-specifications. These files can then be accessed by volunteer devices executing jobs. Similarly, all output files are stored in Reef and are accessible via hyperlinks after the job is completed.

8 POSSIBLE USE CASES

Greyfish has been developed specifically to act as a cloud filesystem for GIB and other similar science gateways. Like any other cloud storage service, it can be used for storing any type of data. It can also be setup as an out-of-the-box, standalone storage application for development and testing environments, where a complete control over the filesystem is desired.

Additionally, Greyfish can be used for data backups - both binary (such as, database backups), and textual files can be backed up. For storing confidential or restricted data, Greyfish can be located within a private network, while retaining all its features.

9 RELATED WORK

There are multiple cloud storage services available, such as AWS S3 [14], Dropbox [15], and Google drive [16]. However, most of these services rely on either a web interface (in the case of Google drive) or a GUI (Dropbox). With the exception of AWS S3 and Google's Cloud Filestore, these services are mostly designed to meet the storage needs of the end-users who can access their data directly.

AWS S3 and Cloud Filestore are designed for server access. However, they can only be set-up on specific cloud computing platforms (AWS and Google Cloud respectively). Access to these services is also limited as they often require specific clients.

Therefore, it is not possible to setup any of the above services for creating science gateways or filesystems on cloud computing systems like Jetstream and Chameleon. Furthermore, all the above services are expensive. However, one advantage that some of these services have is the feature of file encryption, which is not available in Greyfish.

Table 2 shows a side-by-side comparison of Greyfish with other available file-storage services in the cloud. All the services were

exhaustively researched at the time of writing this paper and can change in the future.

10 FUTURE WORK

Greyfish and its simplified version Reef are currently part of the Gateway-In-A-Box and BOINC@TACC projects. As a result, they will continue to evolve depending upon the storage needs of these projects.

Greyfish will provide the option of using percent-encoding instead of the current, custom "++" separator for selecting paths to files and directories. For example, currently, the path DIR/dir2/file.txt becomes "DIR_joe++dir2++ file.txt". However, when percent-encoding will be used, it will become DIR_joe%2Fdir2%2Ffile.txt.

This encoding will allow users to provide files and directories using characters that cannot be used in URLs (such as "?", ">", etc.) as well as improve Unicode support for systems that do not support it in the command-line mode. Although it is already possible, specific configuration options will be added for users who do not need an attached volume. More options will be provided for temporary tokens, such as specifying restricted services for users holding a particular token. The tokens will also be implemented as Redis lists to both allow multiple uses per token and to avoid a possible race-condition (check, delete) if a malevolent user wishes to use a token multiple times in rapid succession.

Future Greyfish versions will guarantee scalability in supporting a large number of files and users. Distributed versions of Greyfish that split the filesystem data across multiple containers will also be supported. Checksums and possibly data encryption for some files may also be supported in the future.

Although a web-interface may not be a part of Greyfish, future APIs would provide a flexible rule-set so that external servers using Greyfish for storage can provide users with a direct view of their files without granting access to the entire system.

11 CONCLUSION

Greyfish provides a simple, out-of-the-box cloud storage system that can be installed in the cloud and can be made accessible to the applications through public APIs and HTTP requests. Therefore, it helps in providing alternatives to commercially available file-storage services.

Any science gateway project that requires out-of-the-box, persistent cloud-based file-storage service can leverage Greyfish and reduce their time-to-development. Greyfish can be especially useful for development and testing environments in which the developers want to have full-control of mounting/unmounting filesystems.

12 ACKNOWLEDGEMENT

All current Greyfish servers used for testing are setup using the Jetstream and Chameleon [17] systems. We are grateful to XSEDE for providing the allocation required for implementing this project. This project is generously supported through the National Science Foundation (NSF) award #1664022.

REFERENCES

[1] Cloud Filestore documentation | Cloud Filestore Documentation | Google Cloud. Retrieved on 2019-04-15 from https://cloud.google.com/filestore/docs/.
[2] Use Volumes | Docker Documentation. Retrieved on 2019-04-15 from https://docs.docker.com/storage/volumes/.
[3] What is a Container? | Docker. Retrieved on 2019-04-15 from https://www.docker.com/resources/what-container.
[4] Gateway-In-A-Box. Retrieved on 2019-04-15 from https://github.com/ritua2/gib.
[5] Ritu Arora, Carlos Redondo, and Gerald Joshua. Scalable Software Infrastructure for Integrating Supercomputing with Volunteer Computing and Cloud Computing. In Majumdar A. and Arora R., editors, *Software Challenges to Exascale Computing. SCEC 2018. Communications in Computer and Information Science*, volume 964. Springer, Singapore, 2019.
[6] Using InfluxDB in Grafana | Grafana Documentation. Retrieved on 2019-04-15 from https://docs.grafana.org/features/datasources/influxdb/.
[7] Greyfish, Portable Cloud Storage. Retrieved on 2019-04-15 from https://github.com/noderod/greyfish.
[8] Gunicorn - WSGI server – Gunicorn 19.9.0 documentation. Retrieved on 2019-04-15 from https://docs.gunicorn.org/en/stable/.
[9] Redis. Retrieved on 2019-04-15 from https://redis.io/documentation.
[10] InfluxDB 1.7 documentation | InfluxData Documentation. Retrieved on 2019-04-15 from https://docs.influxdata.com/influxdb/v1.7/.
[11] Overview of Docker Compose | Docker Documentation. Retrieved on 2019-04-15 from https://docs.docker.com/compose/overview/.
[12] C.A. Stewart, T.M. Cockerill, I. Foster, D. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, S. Tuecke, G. Turner, M. Vaughn, and N.I. Gaffney. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure, 2015. ACM: 2792774. p. 1-8. http://dx.doi.org/10.1145/2792745.2792774.
[13] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkins-Diehr. XSEDE: Accelerating Scientific Discovery. *Computing in Science Engineering*, 16(5):62–74, 2014.
[14] What is Amazon S3? - Amazon Simple Storage Service. Retrieved on 2019-04-15 from https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html.
[15] Dropbox. Retrieved on 2019-04-15 from https://www.dropbox.com.
[16] Using Google Drive-New Features, Benefits & Advantages of Google Cloud Storage. Retrieved on 2019-04-15 from https://www.google.com/drive/using-drive/.
[17] Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. Chameleon: a Scalable Production Testbed for Computer Science Research. In Jeffrey Vetter, editor, *Contemporary High Performance Computing: From Petascale toward Exascale*, volume 3 of *Chapman Hall/CRC Computational Science*, chapter 5. CRC Press, Boca Raton, FL, 1 edition, 2018.

Table 2: Comparison between Greyfish and other cloud storage services

	Greyfish	**Amazon S3**	**Dropbox**	**Google Cloud Filestore**	**Google Drive**
Propietary	No	Yes	Yes	Yes	Yes
Main emphasis	Scientific applications	None	Personal	None	Personal, team sharing
Terminal access	Yes	Yes	Yes	Yes	Yes
Specific client required	Yes	Yes	Yes	Yes	Yes
Accessible through a web interface	No	Yes	Yes	No	Yes
Available for custom setup	Yes	No	No	No	No
Pricing	Free	0.023 $/GB (< 50 GB)	Yes (16.58 $/mo, Professional)	Yes (0.000274 $/GB/hr/VM)	Yes (12 $/user/mo, Business, < 1 TB per user)