# D4: Implementation

## Introduction

Httpfs addresses the problem of the lack of a simple, fast, and secure network filesystem. This problem affects developers and IT administrators, the impact of which is that NFS, originally developed in the 1980s, is overused with no viable alternative. For developers and IT administrators who need an easy-to-set-up remote storage solution, Httpfs is a network filesystem that is easy-to-use, like NFS, but is also fast, secure, and cross-platform. Httpfs makes it easy to access your data in the cloud.

[Github Repo](https://github.com/httpfs/httpfs)

[Trello Board](https://trello.com/b/cY9hPQYZ/httpfs)

## Implemented Requirements

[Trello Board](https://trello.com/b/cY9hPQYZ/httpfs) - **As an administrator, I want to use a custom port for an HttpFs server**: Worked on by whole group, committed by Evin - **As a user, I want to get filesystem stats**: Worked on by whole group, committed by Evin - **As a user, I want to open a file and write to it**: Worked on by whole group, committed by Evin - **As a user, I want to delete a directory**: Worked on by whole group, committed by Evin

## Adopted Technologies

HttpFs- Network filesystem over Http. Since we want to create a HTTpFs server, this is critical.

Python- The programming language we use for HttpFs.

JSON- we use JSON requests in order to do filesystem actions from the client to the server, and vice versa.

Linux- This is the operating system that we intend to use in order to access our server. Windows will also be usable assuming we get that far.

## Learning/Training

How to Access Server- One of the first things we did was to make sure everyone could access the server from their device.

UML Diagrams- during our meetings, we used UML diagrams as a way of outlining what specific classes in our project should do.

Understanding JSON Requests- since the server/client communication uses JSON requests, we have to make sure that we fully understand this concept.

## Deployment

To run the client and server in production, Docker is used. Both the client and server Docker images extend the [Miniconda](https://docs.conda.io/en/latest/miniconda.html) image in order to control their Python environment. Both images install the HttpFs Python environment and the HttpFs Python package.

By default, the HttpFs server Docker image starts the server on port 8080 and serves the directory /srv/httpfs inside the container.

By default, the [client dockerfile](https://github.com/httpfs/httpfs/blob/master/docker/client.dockerfile) attempts to connect to a server called httpfs-server on port 8080 and mounts its directory at /mnt/httpfs.

By default, the [server dockerfile](https://github.com/httpfs/httpfs/blob/master/docker/server.dockerfile) serves the directory /srv/httpfs on port 8080.

**AWS Production Server**: http://httpfsserver-env.eba-tamwr4qv.us-east-1.elasticbeanstalk.com

To connect to the production server, download [docker-compose.yml](https://github.com/httpfs/httpfs/blob/master/docker-compose.yml) file from the HttpFs repository and edit the SERVER\_HOST and SERVER\_PORT environment variables in the httpfs-client section:

...  
   
 httpfs-client:  
 container\_name: httpfs-client  
 image: httpfs/httpfs-client  
 privileged: true  
 environment:  
 SERVER\_HOST: httpfsserver-env.eba-tamwr4qv.us-east-1.elasticbeanstalk.com  
 SERVER\_PORT: 80  
 cap\_add:  
 - SYS\_ADMIN  
 volumes:  
 - /mnt/httpfs/client:/mnt/httpfs:shared  
 devices:  
 - /dev/fuse:/dev/fuse

Then run docker-compose up httpfs-client to start the client. The directory /mnt/httpfs/client will be created and populated with the contents of the HttpFs filesystem on the AWS server. You can read and write to this directory as if it is a directory on your local computer. Currently the production build of HttpFs only supports Linux clients.

## Licensing

### GNU General Public License v3.0

Our team decided to use this strong copyleft license in the hope to encourage any entity to pull from and build upon **Httpfs**. The GNU GPL 3.0 license allows for commercial use, modification, distribution, patent use, and private use of our **httpfs** product

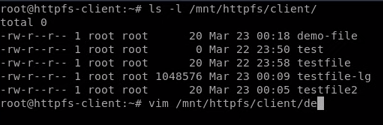
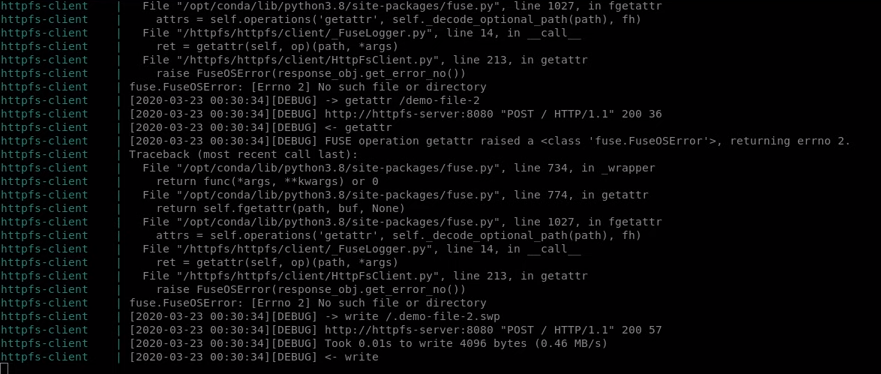
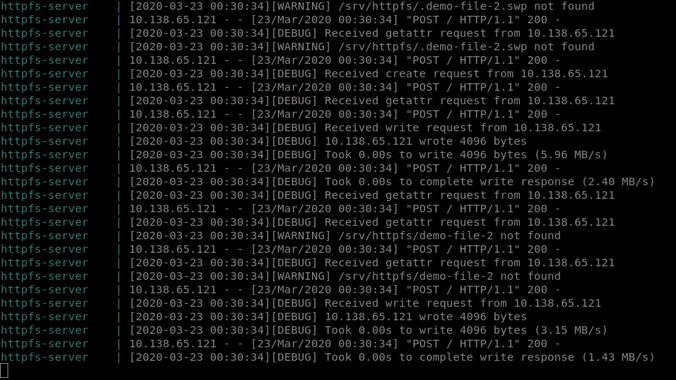
for more information on GNU GPL 3.0, please reference this [gplv3 quick guide.](https://www.gnu.org/licenses/quick-guide-gplv3.html)

## README Files

The main project README, with instructions for Docker deployment in production is located [here](https://github.com/httpfs/httpfs/blob/master/README.md).

[Contributing](https://github.com/httpfs/httpfs/blob/master/CONTRIBUTING.md) [Code of Conduct](https://github.com/httpfs/httpfs/blob/master/CODE_OF_CONDUCT.md) [License](https://github.com/httpfs/httpfs/blob/master/LICENSE)

## Look & feel

Our approach to the look and feel of our product was to implement our user interaction through a command line interface. Using a CLI favors high productivity especially when file interacing actions are taken. In later iterations we hope to develop a simple GUI in which a user can interact with the directory file tree visaully. #### Screenshots: {#screenshots } The current client CLI:   The current server CLI: 

## Lessons Learned

During the developement of our first release of **Httpfs** we ran into a few roadblocks. The first major one was with the configuration of our remote test space on DigitalOcean. The issue was that DigitcalOcean did not assign ip addresses to each machine on the server in the typical way a linux system would. to work around this we had to find and configure the ip address in order to have client and server communicate while both being our hosting server.

The second issue that was ran into that will still need more attention as we step into further development of the product is the write speeds when writing large cluters of small packets. After last testing we reported .02KB/s when writing 1KB clusters. This write speed is well below our standard and we hope to up those speeds as we progress with **Httpfs**.

## Demo

The first HttpFs demo is located [here](https://youtu.be/kGtppGISIBQ). Make sure to watch in 1080 HD in order to see the terminal text.