Multi Cloud File System

Team – 10

Nevedha, Shreyas, Siddharth

Overview

- ❑Highlights of the File System implemented.
- ❑System Components
- ❑Workflow
- ❑Network Topology
- ❑System Architecture
- ❑Code Repo and Structure
- ❑Demo

Highlights of the File System

- Achieve consistent file storage. [uniform across all the clusters]
- Provide single write access using Distributed Mutex.
- Files stored in HDFS (with replications)
- Study and evaluate file storage across multiple clusters.

System Components

Master:

- Maintains meta data of where files are stored
- Handles clients request to read and write files.
- Puts and gets the files from HDFS (deployed on Slave nodes)
- Broadcasts the changes to all other clusters

Slaves:

- The actual place where data files are stored.
- All the slaves in the cluster form a HDFS storage.
- Fault tolerant and replication

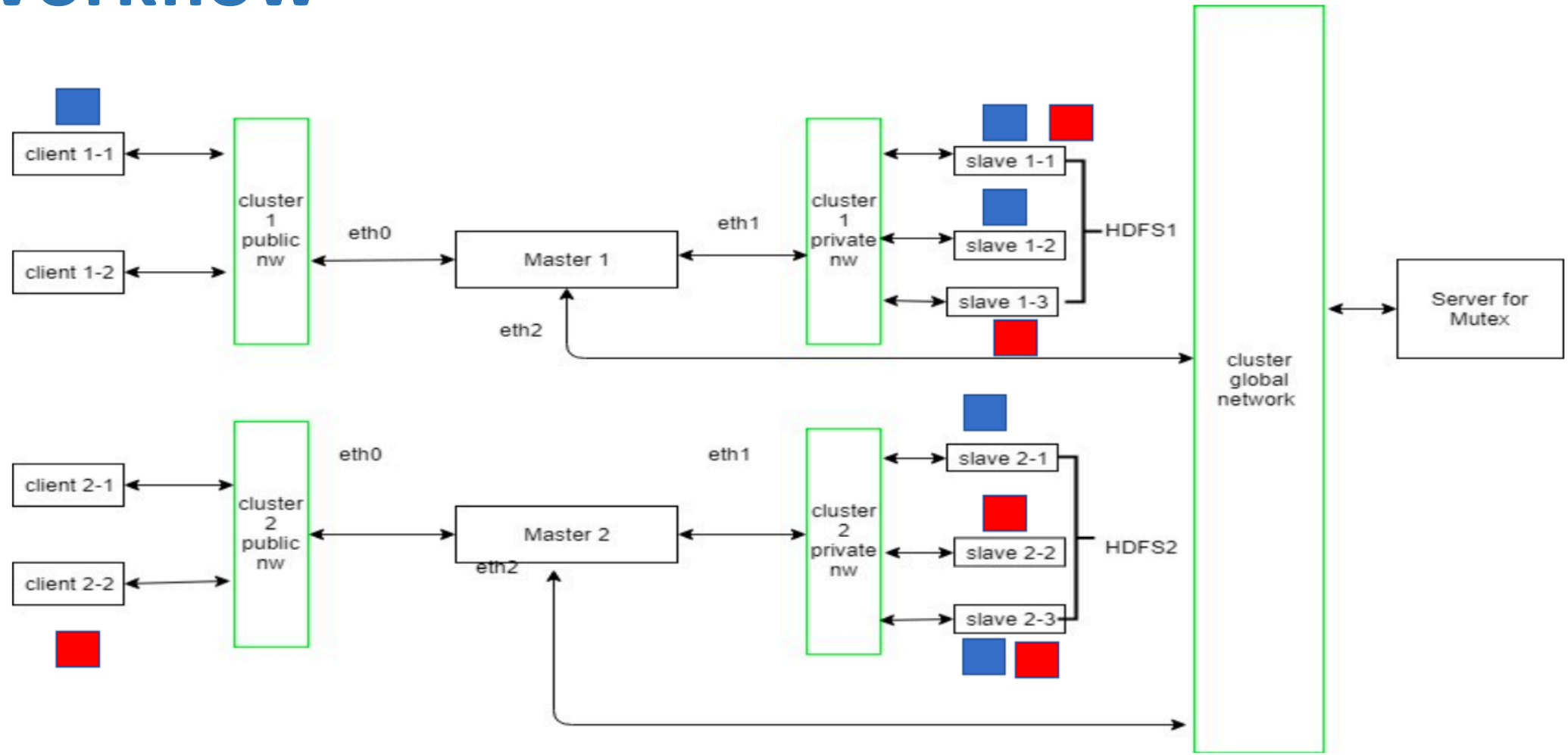
Client:

- Read/Write files

Mutex Server:

- Helps to achieve distributed mutex.
- Maintains a record of which file is being written (holds lock)
- Handles requests from masters to access a file in write mode.
- Fault tolerant (As we have also maintained a backup Mutex Server)

Workflow


client 1-1
client 1-2
cluster 1 public nw
eth0
Master 1
eth1
cluster 1 private nw
slave 1-1
slave 1-2
slave 1-3
HDFS1
eth2
client 2-1
client 2-2
cluster 2 public nw
eth0
Master 2
eth1
eth2
cluster 2 private nw
slave 2-1
slave 2-2
slave 2-3
HDFS2
cluster global network
Server for Mutex

Network Topology

Public Networks:

Formed between Clients and Masters

Private Networks:

This is where Hadoop File System is deployed

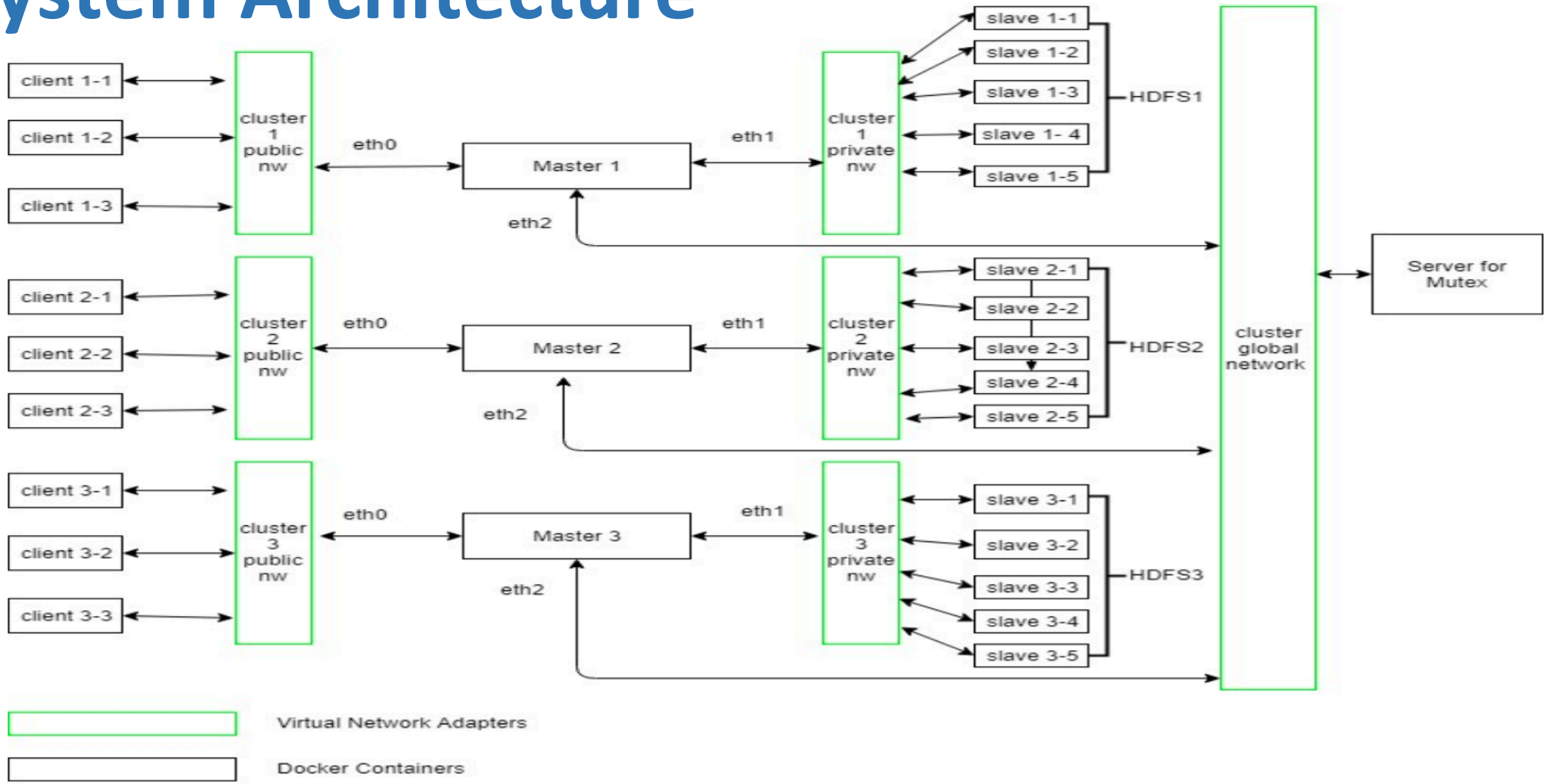
Slave nodes aren’t visible to outside world

Global Network:

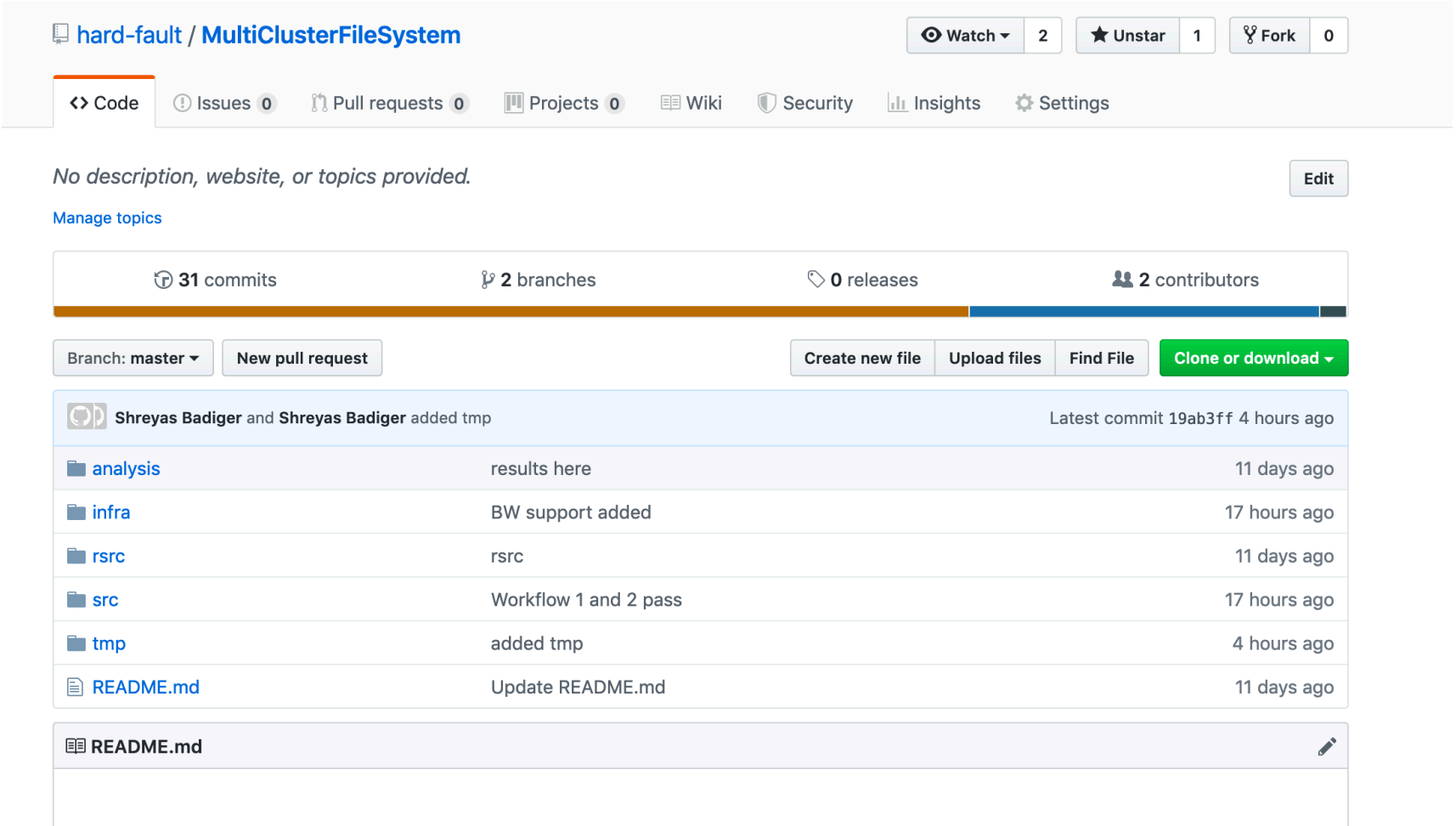
Masters from different clusters communicate over this network

Also to achieve distributed Mutex.

System Architecture



Code Repo [Github]



Code structure

```
MultiClusterFileSystem/
├── README.md
├── src                          (Client and Master scripts for fileSystem)
│   ├── client
│   ├── master
│   └── mutex_server
└── simulation                   (Docker containers for simulation and testing)
    ├── config                   (Descriptive model for container deployment, json file)
    │   └── config.json
    ├── docker                   (Docker file to build custom docker images)
    │   └── DockerFile
    ├── setup
    │   ├── setup.py             (Container deployment script)
    │   └── workflows.py         (Testing scripts)
    └── output
        └── ip.json              (Created upon deployment, IPs of containers used by client-servers)
```

Demo

- ✓ Code walk through
- ✓Deploy Containers
- ✓ Explain workflows
- ✓ Run workflows
- ✓Tear down the setup. (Easy to reproduce ☺)