README.md

Distributed Systems – Project 2

Students:
Lev Svalov - l.svalov@innopolis.university,
Dmitry Podpryatov - d.podpryatov@innopolis.university

Group: **DS-02**

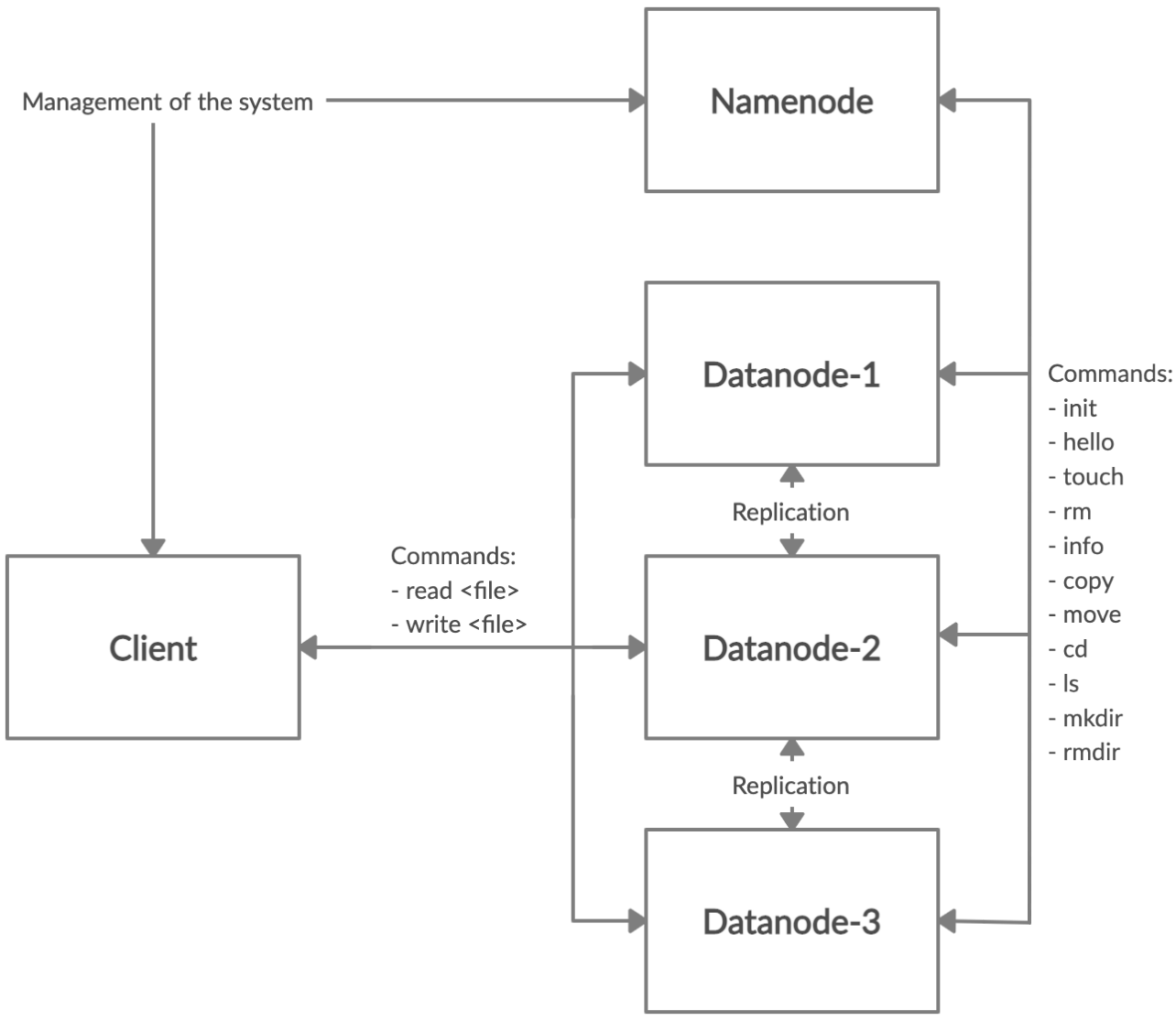
Links:

- GitHub repository
- DockerHub repository

Distributed File System

Architecture

The file system has the following structure:



Description of Communication Protocols

We used `Flask REST API` for all communication between nodes.

The structure of request is as follows:

1. Client sends a GET request of format `http://<namenode IP>/<command>` with specified parameters if needed.
2. Namenode accepts a request, and depending on the command, either redirects a request to chosen datanodes or responses with a datanode or a list of datanodes for the client to contact to upload to or to download a file from.
3. Datanode accepts a request, performs an action, and returns a response.

How to Launch

Clone repository to machines with private closed network

```
git clone https://github.com/DmitriyPodpryatov/ds-project-2.git
```

Install docker and docker-compose on machines

Storage Servers aka Datanodes

On server machines open folder with datanodes' files

```
cd ds-project-2/dfs/datanode
```

Run this to start datanodes

```
sudo docker-compose up -d
```

To access DFS on container, run

```
sudo docker exec -it <container name> bash
cd /dfs
```

Naming Server aka Namenode

On namenode machine open folder with namenode files

```
cd ds-project-2/dfs/namenode
```

Run this to start namenode

```
sudo docker-compose up -d
```

Client

On client machine open folder with client python script

```
cd ds-project-2/client
```

How to Use

Set up an alias for python script

```
alias dfs='python3 client.py'
OR
alias dfs='/home/<user>/ds-project-2/client/python3 client.py'
```

Run commands by

```
dfs <args>
```

List of available commands:

```
dfs hello - get hello from namenode and active datanodes
dfs help - list of all commands
dfs init - initialize DFS and retun available space
dfs touch FILE - create empty FILE
dfs read FILE - download FILE
dfs write FILE DEST_DIR - upload FILE into DEST_DIR
dfs rm FILE - remove FILE
dfs info FILE - show info about FILE
dfs copy SOURCE DEST - copy SOURCE into DEST
dfs move FILE DEST_DIR - move FILE into DEST_DIR
dfs cd DIR - open DIR
dfs ls DIR - list of files in DIR
dfs mkdir DIR - create DIR
dfs rmdir DIR - remove DIR
```

Note:

- Use `/` for the root folder
- No `.` or `..` are allowed
- No trailing `/` are allowed

Contribution

As you may check from history of commits in our repository, we have decided not to explicitly split the task and work separately, but take some different subtasks, do them and after some time, will test and resolve faced problems together. So, it is not easy for us to provide the particular distribution of responsibilities, since both of us contributes to each other task, but here the approximate list for both us:
Dmitry:

- Docker part
- The first implementation of commands - touch, rm, rmdir, ls, cd, mkdir, copy
- A lot of bug fixes, testing, debugging and thinking

Lev:

- FileSystem class implementation
- The first implementation of commands - hello, init, read, write, info, move
- A lot of bug fixes, testing, debugging and thinking