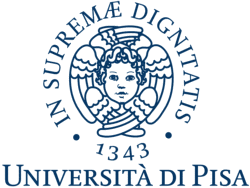
a.a. 2020-2021



CLOUD COMPUTING

CEPH-BASED FILE MANAGER

Presented by:   
Dini federica  
PANICHI NICCOLò  
BICCHIERINI IACOPO  
BIANCHI LORENZO

## general introduction

The aim of the project is to deploy a distributed system to storage files.

* The development is made over three modules of ***ceph-mon***, that we have installed and configured during the laboratory hours of the course.
* We exploit the library ***librados*** to communicate with these modules.
* Each module is inserted in a **juju** container.
* A **client** program is executed to allow the user to access the system and make operations. Every time the user performs an action, an HTTP request is created and sent to a dispatcher:

| **METHOD** | **PATH** | **ACTION** |
| --- | --- | --- |
| **GET** | /files | Retrieve the list of files |
| **POST** | /files/<filename> | Upload a new file |
| **GET** | /files/<filename> | Download a file |
| **DELETE** | /files/<filename> | Delete a file |
| **GET** | /statistics | Retrieve information about the cluster |

* The **dispatcher** is a program inside a **Docker** container, and exposes a REST interface to receive the requests and forward them to one module of *ceph-mon*.
* The choice of the module is done considering the current **workload** of each node, so that the request will be forwarded to the node that has the lower amount of work to perform.
* Each module exposes a REST interface to receive the HTTP request from the dispatcher. This is the **backend** of the application, where a program runs to directly manage the file system and to answer to the dispatcher.

## GENERAL ARCHITECTURE

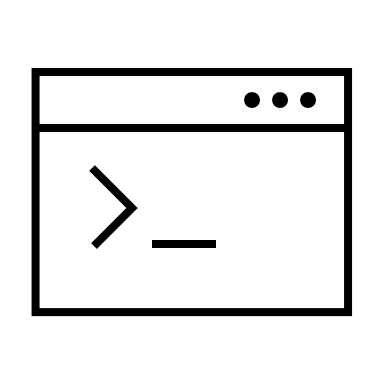
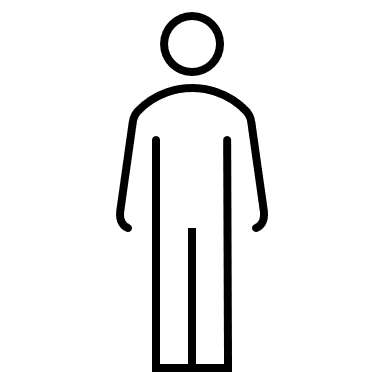
backend.py

backend.py

**HTTP**

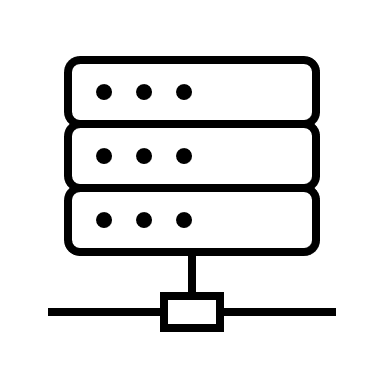
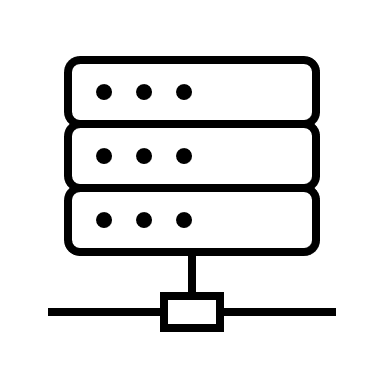
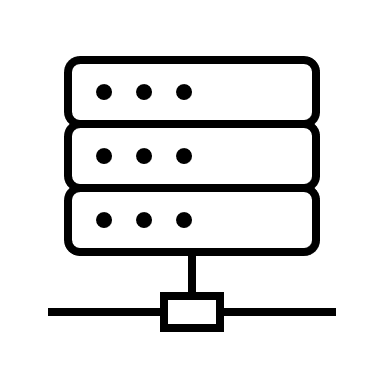
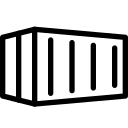
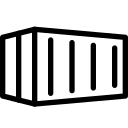
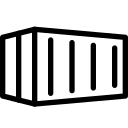
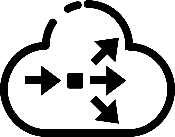
dispatcher.py

172.16.3.190



client.py

172.16.3.247



**LIBRADOS**

backend.py

**HTTP**

172.16.3.197

172.16.3.180

172.16.3.226

CEPH-MON

CEPH-MON

CEPH-MON

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Datanode1** | **Datanode2** | **Datanode3** |
| **IP** | 172.16.3.226 | 172.16.3.180 | 172.16.3.197 |
| **Ceph-MON** | 1/lxd/0 | 2/lxd/0 | 3/lxd/0 |
| **Juju container IP** | 252.3.226.236 | 252.3.180.89 | 252.3.197.73 |
| **Juju container name** | juju-f254bd-1-lxd-0 | juju-f254bd-2-lxd-0 | juju-f254bd-3-lxd-0 |

## deployment

### **Backend**

Inside one of the juju containers create the pool ‘**data**’:

|  |
| --- |
| ceph osd pool create data 8 8 replicated |

For each datanode:

1) Forward the incoming requests from the machine to the container with a new IPTABLES rule:

|  |
| --- |
| iptables -t nat -A PREROUTING -p tcp -i eth0 –drop 8080 -j DNAT –to-destination <juju container IP>:8080 |

2) Enter into the juju container:

|  |
| --- |
| <juju container name> /bin/bash |

3) Create the file **backend.py**:

|  |
| --- |
| vi backend.py |

4) Install ‘rados’ and ‘Flask’:

|  |
| --- |
| sudo apt-get install python3-rados  pip3 install Flask |

5) Create the file **ceph.conf**:

|  |
| --- |
| vi ceph.conf |
| [global] mon host = <juju container IP> |

6) Run the python code:

|  |
| --- |
| lxc exec <juju container name> /bin/bash  python3 backend.py |

### **Dispatcher** (namenode 172.16.3.190)

1) Create the folder ‘**dispatcher**’ and enter it:

|  |
| --- |
| mkdir dispatcher  cd dispatcher |

2) Create the file **dispatcher.py:**

|  |
| --- |
| vi dispatcher.py |

3) Create ‘**Dockerfile**’ file:

|  |
| --- |
| vi Dockerfile |
| FROM python:3 WORKDIR /app  COPY . .  RUN pip3 install Flask  EXPOSE 8080 CMD [ "python3", "dispatcher.py"] |

4) Build and run the container:

|  |
| --- |
| docker build -t dispatcher .  docker run -p 8080:8080 -d dispatcher |

**Client** (controller 172.16.3.247)

1) Create the file **client.py**:

|  |
| --- |
| vi client.py |

2) Run the client:

|  |
| --- |
| python3 client.py |