# Manual

To run the code of the programming assignment #2, one will need to run only one executable per node. Two helping scripts have been written to simplify deployment of the code on EC2 instances. In this document, we will start by describing how to run the main program and we will quickly explain how to deploy the assignment on EC2.

The code to run a node can be found in the `dht` folder and the code related to EC2 is stored in the `ec2` folder.

# 1. Main executable

To run the client and server on one node, a user only need to run the following command `python dht.py config.json` where `config.json` is a configuration file in the following format:

```
{
  "0": {"ip": "173.156.0.1", "port": 5000},
  "1": {"ip": "173.156.0.2", "port": 5000},
  ...
  "n": {"ip": 173.156.0.15, "port": 5000}
}
```

In words, `config.json` contains a map that associates a peer id with its ip and its listening port. To simplify this launch step, we created a script `run.sh` that run the command described above using a configuration file that must be stored in the `dht` folder. This script is stored at the root of the programming assignment and it allows a user to run the code using the simple command `./run.sh`.

Once the program is running, the user can interact with it and give it command through a command line interface. The following actions are possible:

- `exit` terminates the connections, and exit the program.
- `put <k> <v>` adds an entry to the distributed hash table (DHT) with key `<k>` and value `<v>`.
- `get <k>` retrieves and returns the value associated with `<k>` in the DHT.
- `del <k>` removes the entry associated with `<k>` in the DHT.
- `benchmark <action> <first-key> <count>` run a benchmark for the operation `<action>` which can be `put`, `get`, or `del`. It runs `<count>` operations with keys ranging from `<first-key>` to `<first-key> + <count> - 1`. For example, the command `benchmark put 100000 100,000` runs 1,000 `put` operations with keys ranging from 100,000 to 199,999. It returns to the user the results of the operations and the time spent to run the operations. More details on how this command works can be found in the report.

# 2. Additional executables

## 2.1. EC2 helpers

In addition to the main program, two scripts have been developed to help deploying the assignment on EC2 instances. They both assume that instances running on EC2 are nodes used for the assignment. To run these scripts, the user will need to install the `boto` Python package.

- The `deploy_config.py` script takes in parameters an AWS credentials file and the EC2 SSH key. It lists the running

nodes on EC2, automatically creates a configuration file and copy this configuration files to all the nodes in the `dht` folder. By running locally this command, we configure all the nodes for the programming assignment.
- The `connect.py` script takes the generated configuration file, a node id, and the EC2 SSH key as arguments. It connects the user in SSH to the corresponded EC2 instance. This simplifies the connection to the nodes by letting the user use a node id to connect to it instead of having to get the IP address every time.

## 2.2. Create key value pairs for benchmark

In order to benchmark our system, we need to create key-value pairs that will be loaded by every node at startup and used when running the benchmark. To do so, we developed the script `dht/dht_gen_data.py` that create a file in the following format:

```
key_1 val_1
key_2 val_2
...
key_n-1 val_n-1
key_n val_n
```

Each line is a new key-value pair and the key is separated from the value by a space. This makes it easy to load when starting a node. The benchmark data has to be called `keyval.data` and be stored in the `dht` folder for it to be loaded by a node at startup.