## **Install**

For all the environments of FL Server (Aggregator), FL Client (Agent), and Database Server, please create conda environment and activate it.

# macOS

conda env create -n federatedenv -f ./setups/federatedenv.yaml

# Linux

conda env create -n federatedenv -f ./setups/federatedenv\_linux.yaml

Be sure to do conda activate federatedenv when you run the codes.

Note: The environment has Python 3.7.4. There is some known issues of ipfshttpclient with Python 3.7.2 and older.

## **Usage**

### **Running Database and Aggregator**

Here is how to configure the FL server side modules of database and aggregator.

1. Edit the configuration files in the setups folder. The configuration details are explained [here](https://github.com/tie-set/simple-fl/blob/master/setups).
2. Run the following 2 modules as separated processes in the order of pseudo\_db -> server\_th.

python -m fl\_main.pseudodb.pseudo\_db

python -m fl\_main.aggregator.server\_th

### [**Minimal Example**](https://github.com/tie-set/simple-fl/blob/master/examples/minimal)

This sample does not have actual training. This could be used as a template for user implementation of ML Engine.

#### **Sample Execution**

1. Edit the configuration files (config\_agent.json) in the setups folder. The configuration details are explained [here](https://github.com/tie-set/simple-fl/blob/master/setups).
2. Make sure the Database and Aggregator servers are running already. Then, run the minimal example as follows.

python -m examples.minimal.minimal\_MLEngine

#### **Simulation**

FL systems can be run multiple agents for simulation within the same machine by specifying the port numbers for agents.

##### **Agent side**

python -m examples.minimal.minimal\_MLEngine [simulation\_flag] [gm\_recv\_port] [agent\_name]

* simulation\_flag: 1 if it's simulation
* gm\_recv\_port: Port number waiting for global models from the aggregator. This will be communicated to the aggregator via a participate message.
* agent\_name: Name of the local agent and directory name storing the state and model files. This needs to be unique for every agent.

For example:

# First agent

python -m examples.minimal.minimal\_MLEngine 1 50001 a1

# Second agent

python -m examples.minimal.minimal\_MLEngine 1 50002 a2

1. Edit the configuration files in json format in the setups folder. In particular, the agg\_threshold can be 1 in this case.
2. After running the database and aggregator servers, run the minimal\_MLEngine

—-

# **setups**

This directory stores configuration files for (1) environment set-up and (2) aggregator/agent/DB/model set-up

## **federatedenv.yaml**

This yaml file contains all necessary python packages to run this simple FL framework. You can create a virtual Anaconda environment with this script:

conda env create -n federatedenv -f ./setups/federatedenv.yaml

Note: The environment has Python 3.7.4. There is some known issues of ipfshttpclient with Python 3.7.2 and older.

## **config json files**

These json files are read by aggregators, agents, and DB handlers to configure their initial setups.

### **config\_db.json**

* db\_ip: An DB IP address
  + e.g. localhost
* db\_socket: A socket number used between DB and an aggregator.
  + e.g. 9017
* db\_name: Name of the SQLite database.
  + e.g. sample\_data
* db\_data\_path: Path to the SQLite database.
  + e.g. ./db
* db\_model\_path: Path to the directory to save all ML models
  + e.g. ./db/models

### **config\_aggregator.json**

* aggr\_ip: An aggregator IP address
  + e.g. localhost
* db\_ip: An DB IP address
  + e.g. localhost
* reg\_socket: A socket number used by agents to join an aggregator for the first time.
  + e.g. 8765
* exch\_socket: A socket number used to upload local models to an aggregator from an agent. Agents will get to know this socket from the communications with an aggregator.
  + e.g. 7890
* recv\_socket: A socket number used to send back semi global models to an agent from an aggregator. Agents will get to know this socket from the communications with an aggregator.
  + e.g. 4321
* db\_socket: A socket number used between DB and an aggregator.
  + e.g. 9017
* round\_interval: Period of time after which an agent check if there are enough number of models to start an aggregation step. (Unit: seconds)
  + e.g. 5
* aggregation\_threshold: Percentage of the number of collected local models required to start an aggregation step
  + e.g. 1.0, 0.8
* polling: A flag for using a polling method or not. If 1, use the polling method, otherwise use a push method.
  + e.g. 1

### **config\_agent.json**

* aggr\_ip: An aggregator IP address
  + e.g. localhost
* reg\_socket: A socket number used by agents to join an aggregator for the first time.
  + e.g. 8765
* model\_path: A path to a local director in the agent machine to save local models and some state info.
  + e.g. "./data/agents"
* local\_model\_file\_name: A file name to save local models in the agent machine.
  + e.g. lms.binaryfile
* global\_model\_file\_name: A file name to save local models in the agent machine.
  + e.g. gms.binaryfile
* state\_file\_name: A file name to store the agent state in the agent machine.
  + e.g. state
* init\_weights\_flag: 1 if the weights are initialized with certain values, 0 otherwise where weights are initialized with zeros.
  + e.g. 1
* polling: A flag for using a polling method or not. If 1, use the polling method, otherwise use a push method.
  + e.g. 1