Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Source: https://www.rmit.edu.au/study-with-us/biomedical-sciences

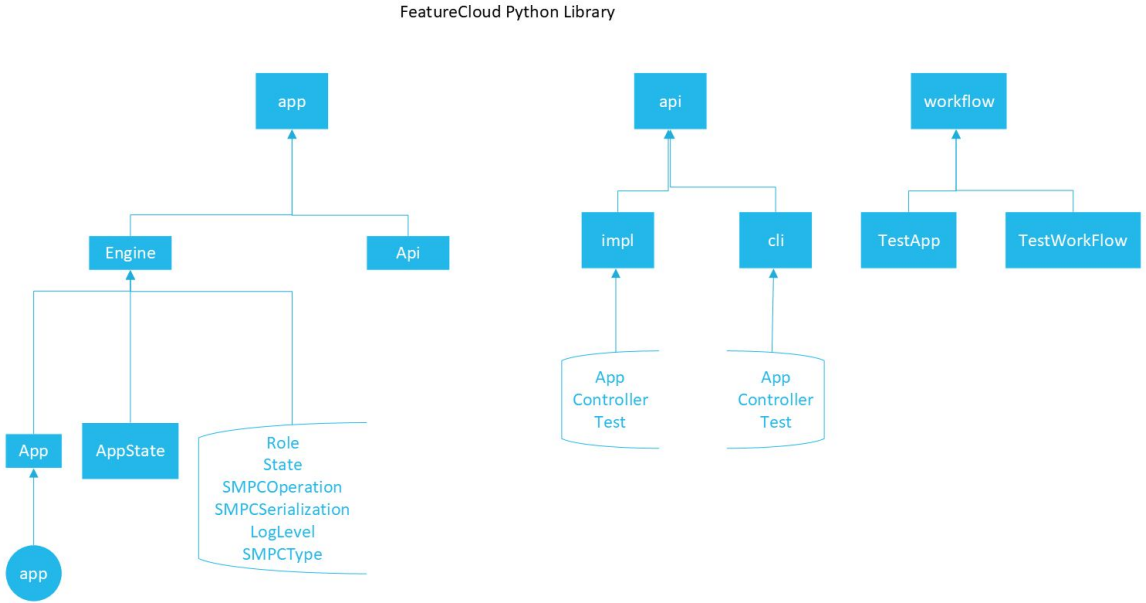
ISMB 2022 - Tutorial Session

Developing Federated Applications

ISMB 2022, Federated Learning in Biomedicine (Tutorial)

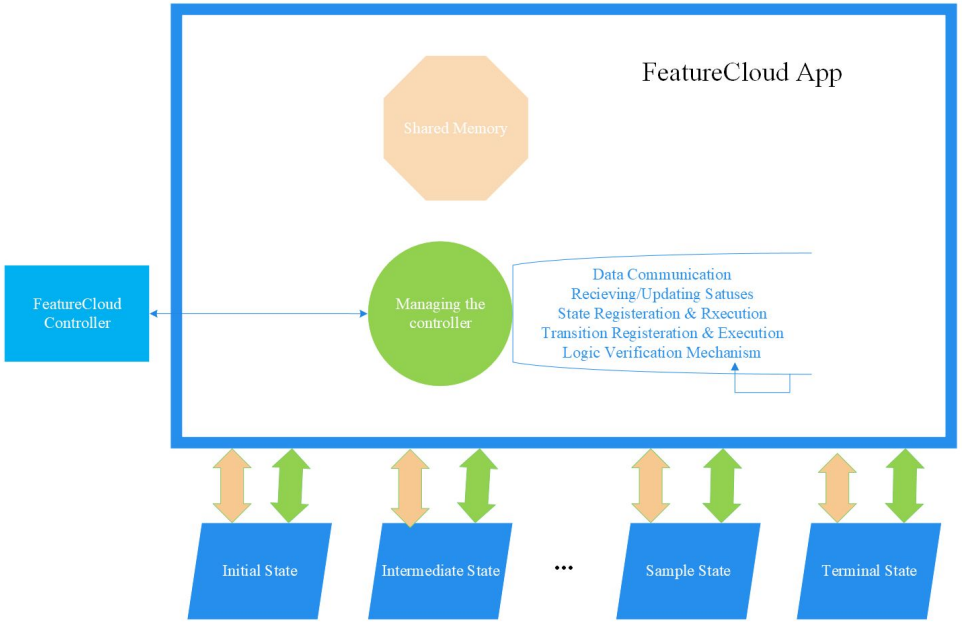
Developing Federated Applications

FeatureCloud pip package: Overview



Developing Federated Applications

FC App, State, and Controller interactions



Developing Federated Applications

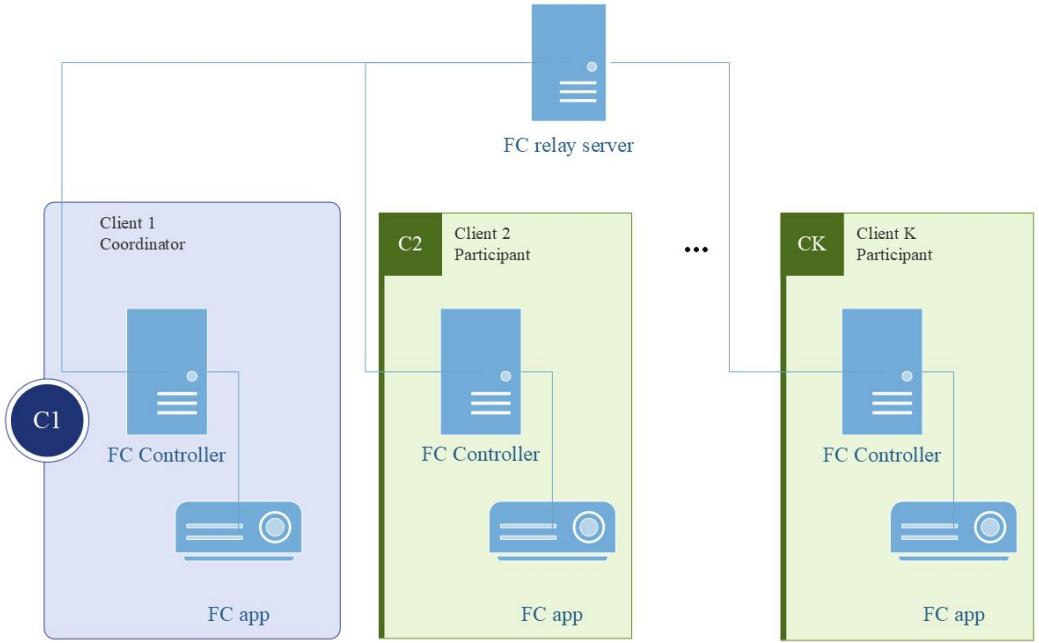
FeatureCloud: Status Attributes

- **status_available**
  - Signalling the controller to execute the communication
- **status_finished**
  - signalling the controller the end of app execution.
- **status_message**
  - messaging the end-user in front-end.
- **status_progress**
  - Informing the end-user of the app progress
- **status_state**
  - Informing the end-user of the state of the app
- **Status_destination**
  - Informing the controller about destination client that you want to communicate to
- **status_smpc**
  - SMPC parameters

Developing Federated Applications

FeatureCloud: Roles

- Tuples
  - COORDINATOR
  - PARTICIPANT
  - BOTH
- Different roles, different access level
  - Data access: Aggregate data
  - State and transition permission



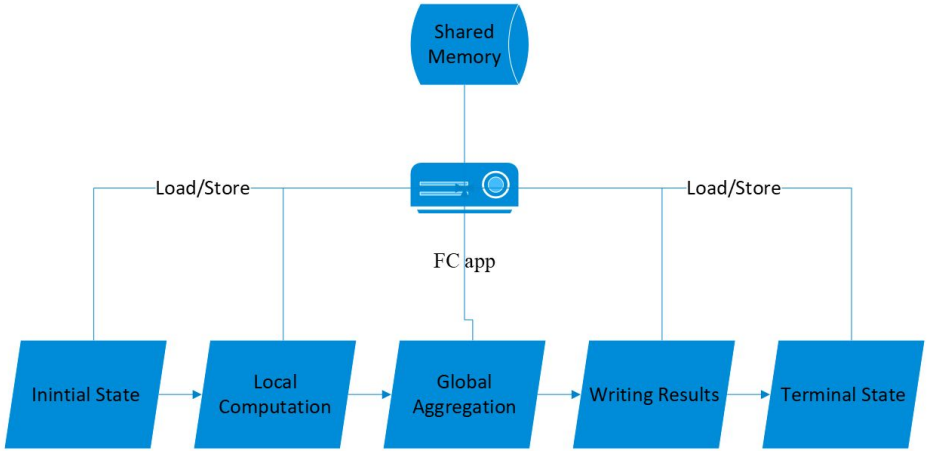
FeatureCloud: Reports

- **Operational States**
  - `RUNNING`: running
  - `ERROR`: error
  - `ACTION`: action_required
- **Log levels**
  - `DEBUG`: debug-related logs.
  - `ERROR`: message the error in the front-end
  - `FATAL`: fatal events that the app may encounter during the execution and cannot recover from.

Developing Federated Applications

FC: Shared Memory

- **Local data sharing among states**
- **App.internal**
  - **Shared memory in the app instance.**
    - **States are separate instantiation of AppState class to handle local and/or global computations.**
    - **No Intrinsic shared memory.**
    - **AppState store and load methods.**
  - **Python Dictionary**



FeatureCloud App: Summary

- **Managing App execution:**
  - Instantiate states
  - Register states
  - Verify the logic
  - Execute states
  - Transit between states
  - Log the app execution
  - Supports shared memory for states
- **Fully transparent:**
  - No instantiation by developers
  - No registration by developers
  - Avoidable direct call by developers

FeatureCloud State: Overview

- **Abstract class**
- **Abstract methods**
  - Registering transitions
  - Executing local computations

- **Communication methods**
  - Aggregating clients data
  - Gathering clients data
  - Waiting to receive data
  - Communicating Data to others
  - Communicating data to the coordinator
  - Broadcasting data

- **Registering a specific transition for state**
- **Updating local app status: update**
- **Configuring SMPC Module**
  - Secure Multi-Party Computation
    - Exponent
    - Shards
    - Operations
    - Serialization

FeatureCloud State: Checking roles and IDs

- Roles and IDs are available once states are registered (in run() method)
- Role of the local app and ID of all clients will be set by the Controller
- Checking the role of the app instance:
    - is_coordinator()
- Checking the clients' ID
    - clients()
- Checking the role of the app instance:
    - id()

FeatureCloud State: Defining a custom state

- Extending the AppState class
- Assissigning roles to states
- defining permitted transitions and permitted roles to take them
- Each state should have a unique name
  - used for naming transitions
- roles, developers should set participant and coordinator

Practical issues

- Unique Naming for states and transitions
- Sharing data across clients
  - Serialization (SMPC: JSON vs other: Pickle)

FeatureCloud State: Extending the AppState class

- **Extend AppState class to define states**
  - implement two abstract methods: register and run
- **Registering transitions**
  - register all possible transitions
    - self.register_transition(target, role, name)
    - just a declaration of transitions
    - eligibility of the transition will be checked.

`register(self)`

- **Computations**
  - All the operations
  - Role base set of operations to handle.
  - Call communication methods
  - Logging
  - Updating operational states
  - report progress
  - Etc.

`run(self)`

FeatureCloud State: Communication methods

Sending Data:

- **Communicating Data to other clients:**
  - `Send_data_to_participant`
    - `data`
    - `destination`
  - Communicate any serializable data using Pickle

- **Communicating data to the coordinator:**
  - `send_data_to_coordinator`
    - `data`
    - `send_to_self=True`
    - `use_smpc=False`

- **Broadcasting data:**
  - `broadcast_data`
    - `data`
    - `send_to_self=True`
  - Only for the coordinator

FeatureCloud State: Communication methods

Receiving data:

- **Gathering clients data**
  - Only coordinator
  - For all clients' data
  - Without Aggregation

```
gather_data(self, is_json=False)
```

- **Aggregating clients data**
  - automatically handles SMPC
  - the same data structure and shape as the one was sent out
  - Structural and data consistency
  - SMPC usage:
    - Using SMPC: looks like waiting for just one client.
    - Without SMPC: waits for all clients

```
aggregate_data(self, operation:
SMPCOperation, use_smpc=False)
```

FeatureCloud: Creating an app

```
$ featurecloud app new <APP_NAME> <template_name>
```

- APP_NAME:
  - a desired app name
  - if not provided, the name of the containing directory will be used
- template_name:
  - one of the provided templates by FC GitHub repositories can be used
  - Default app-blank
    - Sort of hello-world template
    - Only includes initial state

Developing Federated Applications

FeatureCloud: Creating an app

```
$ featurecloud app new -app-name
fc_test -template-name app-blank
```

```
fc test
├── Dockerfile
├── LICENSE
├── main.py
├── README.md
├── requirements.txt
├── states.py
└── server config
    ├── docker-entrypoint.sh
    ├── nginx
    └── supervisord.conf
```

```
from bottle import Bottle
from FeatureCloud.app.api.http ctrl import api server
from FeatureCloud.app.api.http web import web_server
from FeatureCloud.app.engine.app import app
import tutorials.communicate.app

server = Bottle()


if   name   == ' _main__':
   app.register()
   server.mount('/api', api server)
   server.mount('/web', web server)
   server.run(host='localhost', port=5000)
```

FeatureCloud: Simple initial state

- First state in any FC app
- register method:
  - introducing transition to terminal state
    - last state in FC app
    - works as a flag to show the end of app execution
- run method
  - executes all the local/global computation
  - Data communication and I/O
  - Transitions
  - Reporting

```
@app_state(name='initial')
class InitialState(AppState):
    def register(self):
        self.register_transition('terminal')

    def run(self):
        self.log('Hello World!')
        return 'terminal'
```

FeatureCloud: App deployment

```
$ featurecloud app build <app-name>
```

- app-name:
  - developed app name
- The app's docker image will be created based on Dockerfile, requirement, and the app implementation

```
$ featurecloud app publish -name fc_test -tag latest
```

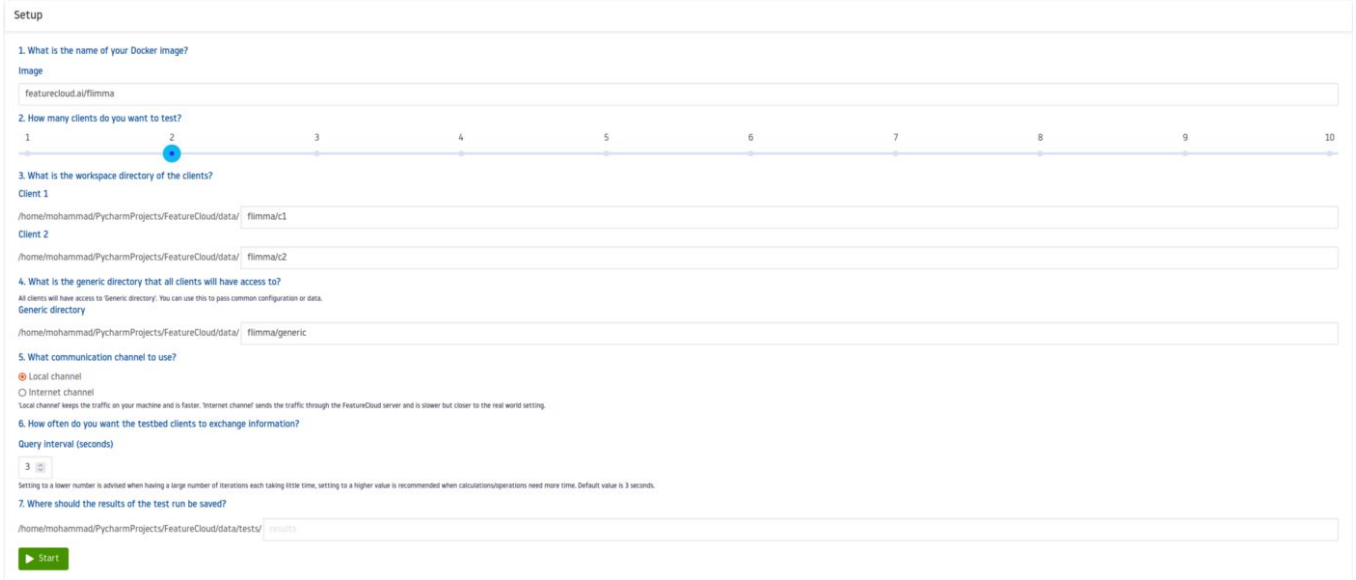
- publishing apps in AI-store
- pushing apps image into the FC docker repository
  - All app names should start with featurecloud.ai/
  - fc_test is not valid
  - either build the app with another name or tag it

FeatureCloud: App execution

- testbed:
  - Standalone app
  - providing input data and config file(if required)
  - Suitable for app developers to test their platform
  - Supported by both front-end and the pip package's CLI
- workflow
  - Linear execution of a workflow of apps
  - providing first app's data
  - providing a config file for all apps
  - passing any app's results as input for the following app
- test workflow
  - Non-linear workflow
  - Supported in the pip package's CLI

Developing Federated Applications

FeatureCloud: Testbed on the front-end



FeatureCloud: Testbed on CLI

Running an app as a testrun in FC testbed

-

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

ISMB 2022 - Tutorial - Federated Learning in Biomedicine

Thank you!