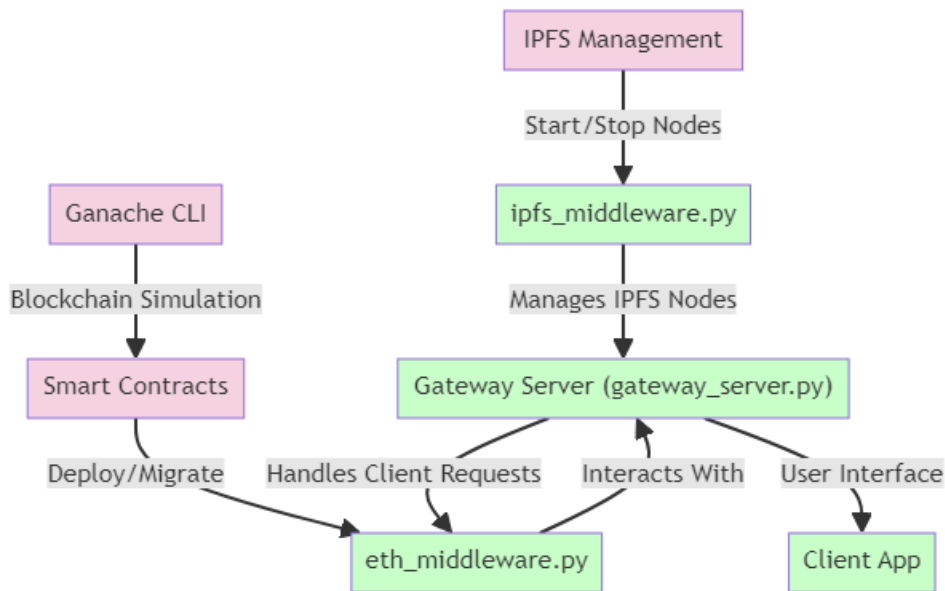


# Smart Contract-Based Data Sharing



This project involves a decentralized application (dApp) framework that integrates Ethereum blockchain technologies and IPFS (InterPlanetary File System) for data management and smart contract interactions. The main components include:

**Smart Contracts:** Managed by DataToken.sol, these contracts handle the logic for token management and interactions on the Ethereum blockchain.

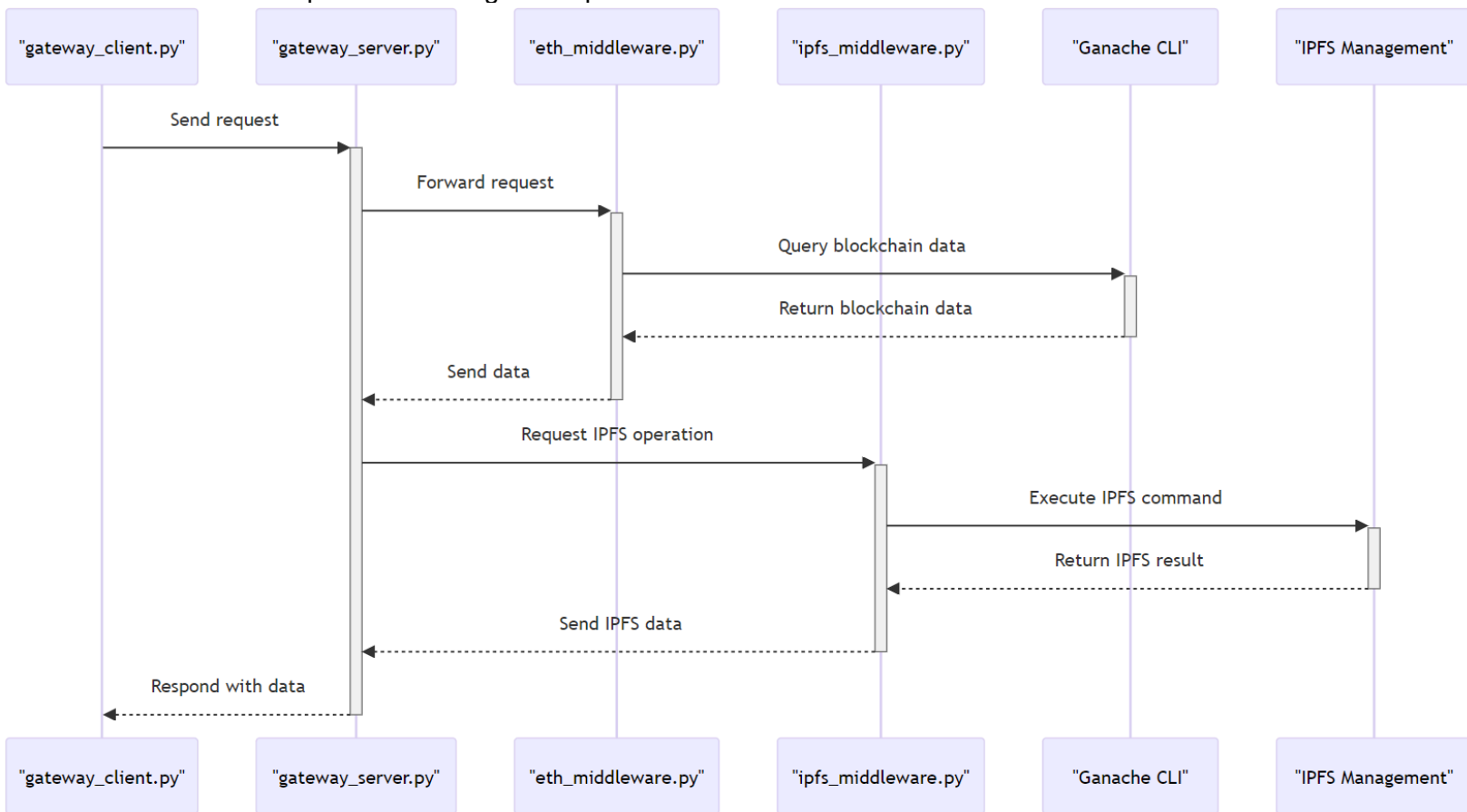
**Ganache CLI:** Used as a local blockchain simulator to test and deploy smart contracts without needing an actual Ethereum network.

**Middleware Servers:**

- **eth\_middleware.py:** Interfaces with the Ethereum blockchain to facilitate interactions with deployed smart contracts.
- **ipfs\_middleware.py:** Manages IPFS nodes to store, retrieve, and manage decentralized files.

Gateway Server (gateway\_server.py): Acts as the main interface for client applications, directing API calls to the appropriate middleware server based on the request.

Client Application (gateway\_client.py): Interacts with the gateway server to perform blockchain and IPFS operations through a simplified interface.



Shell Scripts:

- manage\_ipfs.sh: A utility script to start and stop IPFS nodes.
- start\_project.sh: Automates the process of starting the local blockchain, compiling and migrating smart contracts, and launching middleware and gateway servers.

## Demo Video

<https://www.youtube.com/watch?v=HjShlcFITQU>

## Prerequisites

Before starting, ensure you have the following installed:

- Node.js - JavaScript runtime required for Truffle and Ganache CLI.
- Truffle - A development framework for Ethereum.
- Ganache CLI - A personal Ethereum blockchain for testing.
- Python - Required for the middleware layer; ensure `venv` is installed.
- Docker - Required to run the IPFS nodes
- Git - Version control system for cloning the project repository.

## Initialize Python Virtual Environment and Install requirements.txt

Create and activate a Python virtual environment in the project root:

```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

## Initialize Node Project

Initialize a Node.js project and install the necessary packages:

```
npm init -y
npm install web3 ganache-cli truffle
```

# 1. Run the Setup Scripts

Execute the `manage_ipfs.sh` script to automatically set up the docker instances for the IPFS nodes.

```
(myenv) root@instructor-virtual-machine:/GitHub/Blockchain-Projects/SmartContractDataSharing# bash manage_ipfs.sh
1) Start IPFS Nodes
2) Stop IPFS Nodes
3) Quit
Please enter your choice: 1
d619bb0c61d7c53f0c84a402e9872eeeada6dc4fd7185a50ffe22e4150a4a7c5
fd1706d8b266e0ca64d2a0c7647dda1b8af87c910be93b51dc239d9b64e370f3
Started 2 IPFS nodes.
Please enter your choice:
```

Execute the `start_project.sh` script to automatically set up the project. This script initializes a local Ethereum blockchain using Ganache CLI, compiles and deploys the smart contract using Truffle, and starts both the middleware and gateway components.

`bash start_project.sh`

```
(myenv) root@instructor-virtual-machine:/GitHub/Blockchain-Projects/SmartContractDataSharing# bash start_project.sh
Ganache CLI started with PID 1214482
Compiling and migrating contracts...

Compiling your contracts...
=====
> Compiling ./contracts/DataToken.sol
> Artifacts written to /GitHub/Blockchain-Projects/SmartContractDataSharing/build/contracts
> Compiled successfully using:
  - solc: 0.8.10+commit.fc410830.Emscripten.clang

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Starting migrations...
=====
> Network name:   'development'
> Network id:    5777
> Block gas limit: 10000000 (0x989680)

1_deploy_token_manager.js
=====

Replacing 'DataToken'
-----
> transaction hash: 0x8349b84eabe7ec27d45c23fed51d59c0c7a8b7d684d037219dac45c6c20a3d7
> Blocks: 0
> contract address: 0x3285f926E64c7730C98879F3807F321E2447c2e9
> block number: 1
> block timestamp: 1713024611
> account: 0x5965A83e90126bF0268b75f10e17187B3325b81e
> balance: 99.97512834
> gas used: 1243583 (0x12f9bf)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.02487166 ETH

> Saving artifacts
-----
> Total cost: 0.02487166 ETH

Summary
=====
> Total deployments: 1
> Final cost: 0.02487166 ETH

DataToken contract deployed at 0x3285f926E64c7730C98879F3807F321E2447c2e9
```

Please ensure that the script has executable permissions. If not, grant them using:

```
chmod +x start_project.sh.
```

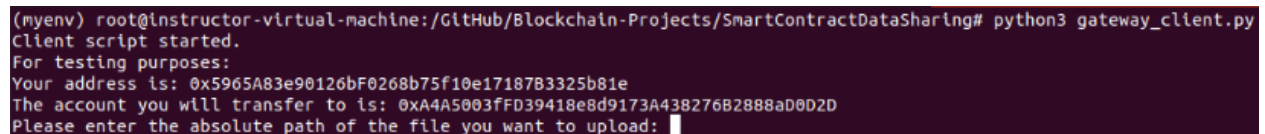
## 2. Test the Setup

After the setup scripts are both complete successfully, you can test the smart contract and API endpoints using the provided `gateway_client.py` script.

```
python3 gateway_client.py
```

This script will interact with the smart contract through the gateway and middleware utilized by both the Ganache CLI network and the Docker IPFS nodes.

You will be prompted to enter an absolute file path of the file that you wish to upload and register to the blockchain:



```
(myenv) root@instructor-virtual-machine:/Git/Hub/Blockchain-Projects/SmartContractDataSharing# python3 gateway_client.py
Client script started.
For testing purposes:
Your address is: 0x5965A83e90126bF0268b75f10e17187B3325b81e
The account you will transfer to is: 0xA4A5003fFD39418e8d9173A438276B2888aD0D2D
Please enter the absolute path of the file you want to upload: █
```

Once you have given an absolute file path that you wish to upload, the script will utilize all of the coded functions from both the Smart Contract and the IPFS nodes.

After the upload is completed the script will:

1. Register the file to the Smart Contract's blockchain
2. Transfer the ownership of the file to another test account
3. That new owner initiates a burning request for the data and the information is purged
4. The tracker function is queried to return all individuals that have owned the file
5. The file is downloaded from via IPFS

## Additional Information

- The project also includes a `requirements.txt` file, ensuring that you install the exact versions of Python dependencies that the project relies on.

For more detailed information on the project components or configurations, please refer to the respective documentation or source code comments within the project repository.

## Notes

*How does your system design ensure data integrity during the data-sharing process?*

- Blockchain Technology: Uses the Ethereum blockchain to create an immutable ledger, ensuring that all data transactions cannot be changed retroactively.
- IPFS for Storage: Uses IPFS to store data in a decentralized manner, with each file's unique hash acting as a tamper-evident feature.

*How does your system design ensure transparency and controllable data transfer?*

- Smart Contract Governance: Transparent smart contracts manage all data transactions and can be audited by anyone on the network.
- Decentralized Control: Uses decentralized platforms such as Ethereum and IPFS to give users complete control over their data without interference from central authorities.

*How does your system design protect sensitive information during data sharing?*

- Encryption Practices: Encrypts sensitive data before storing it on IPFS, ensuring that it is secure from unauthorized access.
- Access Control with Smart Contracts: Uses smart contracts to define and enforce access control policies, ensuring that only authorized users can access or operate on sensitive data.

## File Tree

```
SmartContractDataSharing\  
├── gateway_client.py  
├── server_gateway.py  
├── eth_middleware.py  
├── ipfs_middleware.py  
├── start_project.sh  
├── manage_ipfs.sh  
├── requirements.txt  
├── contracts\  
│   └── DataToken.sol  
├── migrations\  
│   └── 1_deploy_token_manager.js  
├── build\  
│   └── contracts\  
│       └── DataToken.json  
└── truffle-config.js
```