

# Innovation Lab - Report

Content:

- [Building blockchain which keep the details of various agricultural transaction](#)
  - [Modules Imported](#)
  - [Class Blockchain](#)
  - [Mining the blockchain](#)
  - [Decentralising the blockchain](#)
- [Covid-19 Data Collection dapp](#)
  - [Writing smart contracts](#)
  - [Building web interface to connect with blockchain](#)
  - [Building web interface for normal user](#)
  - [Hosting blockchain locally on Ropsten](#)
  - [Hosting blockchain on ipfs server](#)
  - [Accessing website Using Pinata as a pinning service for ipfs](#)

## Building blockchain which keep the details of various agricultural transaction

IMPLEMENTATION:-

### 1. MODULES IMPORTED

- a. Datetime

- b. Hashlib
- c. Json
- d. flask—Flask, jsonify, requests
- e. Requests
- f. Uuid—uuid4
- g. Urllib.parse--urlparse
- h. crypto.publickey—RSA

## 2. CLASS BLOCKCHAIN

- a. **Self- chain**(emptylist) , farmer\_details(emptylist), nodes(set), create\_block(function to create the genesis block)
- b. **Create\_block**- function, parameters-(self, proof, previous\_hash)  
Description- It creates a dictionary block which contains index(length of chain+1),timestamp( by using the module datetime), Proof( passes as parameter),previous\_hash(passed as parameter), Farmer\_details(from self) and append this to the chain.
- c. **get\_previous\_block**- function, parameter(self)  
Description-It returns the last block of the chain.
- d. **Proof\_of\_work**- function,parameter(self, previous\_proof)  
Description- It runs a loop and checks if the hash of new proof^2- previous proof^2 contains 4 leading zeroes. if yes,then it returns the new proof otherwise increment the new proof by 1 and iterates again.
- e. **Hash**- function,parameters(self,block)  
Description- It returns the hash of the block using sha256
- f. **Is\_chain\_valid**- function, parameter(self,chain)

Description- It iterates a loop from 0 to chain length and checks if hash of the block is the same as returned by the hash function, then it checks if hash of the proof of current block<sup>2</sup>-proof of previous block<sup>2</sup> contains 4 leading zeroes or not. if no, then the chain is not valid.

- g. **Add\_farmerdetails**- function, parameters (self, name, crop\_name, quantity, rate)

Description- It creates the private key using the RSA.generate(1024), then creates the public key, hash of transaction (it is the hash of the sum of hashes of the name, crop\_name, quantity, rate), data (it is the hash of the transaction in the int form), signature (it is created by raising the data to the power of privatekey.d%privatekey.n). Then it appends a dictionary containing all this information in the hash format to the chain farmer\_details and returns the index of the new block.

- h. **Add\_node**- function, parameters ( self, address)

Description- It takes the url using the urlparse of the address and then adds this to the set nodes in the self.

- i. **Replace\_chain**- function, parameter(self)

Description- It accesses all the nodes in the set nodes and then iterates a loop to get their chain length using get\_chain (to be described) and replaces the current chain with the longest chain of all the nodes.

### 3. MINING THE BLOCKCHAIN

- a. **App**- It uses the flask to create a web app.

- b. **node\_address**- It creates an address for the node using uuid4 and removing –

- c. **mine\_block**- `@app.route('/mine_block', methods = ['GET'])`  
Description- It access the previous block by calling the function `get_previous_block()`, then access the previous proof by `previous_block['proof']`, then it creates a new proof by using the function `proof_of_work('previous_proof')`, then it finds the hash of the previous block by using the function `blockchain.hash(previous_block)`, then calls the function `create_block( proof,previous_hash)`, then finds the hash of this block. It creates a response containing all the details of the new block, jsonify it and returns it.
- d. **print\_chain**- `@app.route('/print_chain', methods=['GET'])`  
Description- It creates an empty list `chain_till_now`, then iterates over all the blocks in the blockchain and find it's hash then check if the list `farmer_details` is empty or not, if it is empty then it appends a dictionary containing the current block's `index,timestamp,proof,previous_hash, current_hash, farmer_details`. If the `farmer_details` list is not empty then it first finds the length of the list `farmer_details` then it iterates over the length of the list `farmer_details` and appends the hash of transaction contained within the dictionary of the list `farmer_details`. Then it creates the hash of this appended hash. This is the merged hash. Then it creates a dictionary containing merged `hash,index,timestamp,proof,previous_hash,farmer_details` and current hash. Then, it appends this dictionary to the list `chain_till now`. It then creates the response containing the chain till now and length of the blockchain, jsonifies it and returns it.
- e. **get\_chain**- `@app.route('/get_chain', methods = ['GET'])`  
Description- It creates the response containing the `blockchain.chain` and its length, jsonifies it and returns it.

- f. **is\_valid**- @app.route('/is\_valid', methods = ['GET'])  
Description- It calls the function is\_chain\_valid and returns a string as response based on whether the chain is valid or not.
- g. **add\_farmer\_details**- @app.route('/add\_farmerdetails', methods = ['POST'])  
Description- It takes the input in Jason format and checks if all the keys in the farmer keys(name\_of\_farmer,crop\_name,quantity\_inkg, rate\_perkg) are available in the json file. If no, It returns that some elements are missing otherwise it calls the function add\_farmer\_details by passing the farmer details in the json file as parameter and returns the index of the block in which these details will be added.

#### 4. DECENTRALIZING THE BLOCKCHAIN

- a. **Connect\_node**- @app.route('/connect\_node', methods = ['POST'])  
Description- It takes a Jason file as a request and first checks if it contains any node or not. If it contains the nodes then it calls the function blockchain.add\_node .Then it returns the list of blockchain.nodes as response.
- b. **replace\_chain**- @app.route('/replace\_chain', methods = ['GET'])  
Description- It calls the function blockcain.replace\_chain. If the chain is replaced it returns the response with a message that the nodes have the different chains so the chain has been replaced by the longest chain along with the blockchain.chain. Otherwise it returns the response with a message all good the chain is the longest one with the blockchain.chain .then it jsonify the response and returns it.

- c. **App.run** (host = '0.0.0.0', port = 5001)- Different ports are provided for the different nodes.
- d. **nodes.json**- It is a json file which contains the format in which the post request for connect\_node has to be entered.
- e. **farmer.json**- It is a json file which contains the format in which the post request for the add\_farmerdetails has to be entered.

## COVID-19 DApp

### Writing smart contracts



### Data structure

Indian structure contains all the details which are needed to be stored for a state.

```
struct Indian{
    uint id;
    string state_name;
    uint Covid19_activecases;
    uint Covid19_effectedcases;
    uint Covid19_deaths;
    uint Covid19_recovered;
}
```

An array named Indian\_states of structure Indian.  
Nextid is used to store the size of the array.

```
Indian[] public Indian_states;  
uint public nextid;
```

## Functions

There are five functions:-

1. New\_state: It enters a new state in the blockchain. Provide state name, effected cases, recovered, active cases, deaths.

Implementation:-

```
function New_state(string memory state_name,uint activecases,  
    uint effectedcases,  
    uint Deaths,  
    uint Recovered) public  
{  
    uint Covid19_activecases=activecases;  
    uint Covid19_recovered=Recovered;  
    uint Covid19_effectedcases=effectedcases;  
    uint Covid19_deaths=Deaths;  
    Indian_states.push(Indian(nextid,state_name,Covid19_activecases,Covid19_effectedcases,Covid19_deaths,Covid19_recovered));  
    nextid++;  
}
```

2. Search\_statename: It is a call function. Enter the state name and it will display all information of the state. If the entered state name is wrong, it will throw an error.

Implementation:-

```
function Search_statename (string memory state_name) view public  
returns(string memory, uint,string memory,string memory,string  
memory,uint,string memory,uint,string memory,uint,string  
memory,uint)  
{  
    uint i=find_id(state_name);  
    return("Id",Indian_states[i].id,  
        "State name",Indian_states[i].state_name,  
        "Activecases", Indian_states[i].Covid19_activecases,  
        "Effected cases", Indian_states[i].Covid19_effectedcases,  
        "Deaths", Indian_states[i].Covid19_deaths,
```

```

        "Recovered cases", Indian_states[i].Covid19_recovered
    );
}

```

3. **Update\_state**: It updates the information of an already existing state. Enter the state name, active cases, recovered, deaths, effected cases.

Implementation:

```

function Update_state( string memory state_name, uint activecases,
uint effectedcases, uint Deaths,uint Recovered) public
{
    uint i=find_id(state_name);
    Indian_states[i].state_name=state_name;
    Indian_states[i].Covid19_activecases=activecases;
    Indian_states[i].Covid19_effectedcases=effectedcases;
    Indian_states[i].Covid19_deaths=Deaths;
    Indian_states[i].Covid19_recovered=Recovered;
}

```

4. **Delete\_state**: It deletes the details of any state. Enter the state name which you want to delete.

Implementation:

```

function Delete_state(string memory state_name)public{
    uint i=find_id(state_name);
    delete Indian_states[i];
}

```

5. **Find\_id**: It checks whether the state name provided as an argument is present in the blockchain or not.

Implementation:

```

function find_id(string memory state_name)view internal
returns(uint){
    for(uint i=0;i<Indian_states.length;i++){
        if(keccak256(abi.encodePacked(Indian_states[i].state_name))
            == keccak256(abi.encodePacked(state_name)))
        {
            return i;
        }
    }
}

```



```
}  
    revert('This state does not exist in blockchain!');  
}
```

## Building web interface to connect with blockchain

Technologies Used:-

1. Truffle: Used for creating a development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM).
2. Web3: web3.js is used to interact with remote ethereum node using HTTP, IPC or WebSocket.
3. Webpack: It is used to bundle all js files into one.

Further details about hosting this locally, or using ipfs is documented [here](#).

## Building web interface for normal user

Get, Post, Put and Delete apis are using PHP. MySQL is used as a database.


It is hosted using website

<https://app.infinityfree.net/>

And is live at domain

<http://covid-database.freecluster.eu/user/>

Structure of database

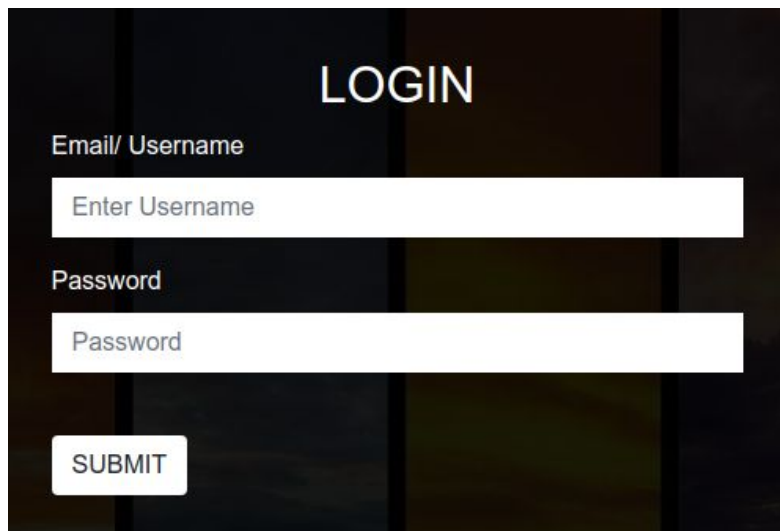
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	<b>id</b> 	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	<b>state_name</b>	varchar(250)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	3	<b>active</b>	int(11)			No	None		
<input type="checkbox"/>	4	<b>aff</b>	int(11)			No	None		
<input type="checkbox"/>	5	<b>death</b>	int(11)			No	None		
<input type="checkbox"/>	6	<b>rec</b>	int(11)			No	None		
<input type="checkbox"/>	7	<b>isresponded</b>	int(11)			No	None		

Special users can login to check the update/add requests, and add to blockchain after validating.

Demo Credentials for special user:

Username: pranaykgupta

Password: qwerty



A screenshot of a login interface. At the top, the word "LOGIN" is displayed in large, white, uppercase letters. Below it, there are two input fields. The first is labeled "Email/ Username" and contains the placeholder text "Enter Username". The second is labeled "Password" and contains the placeholder text "Password". At the bottom left of the form, there is a white button with the word "SUBMIT" in black, uppercase letters. The background of the form is dark and slightly blurred.

## Hosting blockchain locally on Ropsten ( Ethereum Public Network)

To connect the blockchain to Ethereum Public Test Network , we have used infura <https://infura.io/>. It is an Ethereum and ipfs API . After Creating an account on infura , we select Ropsten Test Network . It will give you a

Project ID and Project Secrets . Then, the blockchain is connected to the Ropsten test network using the Project Id.

The screenshot shows a web interface for 'covid19\_blockchain'. The top navigation bar includes a logo, the project name, an 'UPGRADE' button, and a user profile icon. A left sidebar contains links for 'REQUESTS', 'SETTINGS' (active), 'ETHEREUM', 'FILECOIN', 'DOCS', and 'COMMUNITY'. The main content area is titled 'PROJECT DETAILS' and contains a form with a 'NAME' field set to 'covid19\_blockchain' and a 'SAVE CHANGES' button. Below this is a 'KEYS' section with a table of project identifiers and endpoints.

PROJECT ID	PROJECT SECRET
c6ba759c54c24d72b386e5b2944bf78f	7a93be18f6594373b8831261a64bc638

Below the table, the 'ENDPOINTS' dropdown is set to 'Ropsten', showing the following URLs:

- https://ropsten.infura.io/v3/c6ba759c54c24d72b386e5b2944bf78f
- wss://ropsten.infura.io/ws/v3/c6ba759c54c24d72b386e5b2944bf78f

Currently, we have 2 web-pages to access our covid-19 daap. In first web page , only search option is available. This is to ensure thar normal users can only search the blockchain database but cannot directly make transactions in it. In the second webpage, all transactions options are available. This webpage can only be accessed through authenticated users. For this purpose a Login option has been created.

## FIRST WEB PAGE

## How to use?

1. Check whether the state/city/country name is already stored in blockchain or not. If available, check whether the covid-19 information of that place is updated or not.
2. If place name is not present in the blockchain, you can provide the covid-19 details of that place in enter new state/city/country.
3. If place name is already available in database and the covid-19 details is not up to date then you can provide us the updated details in update state/city/country.
4. Your information will be checked and if found correct, then will be inserted in the blockchain.

NOTE:- The state/city/country name spelling should be according to [https://en.wikipedia.org/wiki/Lists\\_of\\_cities](https://en.wikipedia.org/wiki/Lists_of_cities).

## Enter the name of state/city/country to be searched

State/city/country Name

SUBMIT

## SECOND WEB PAGE

### Covid19\_blockchain

#### ENTER new state/city/country

State Name

Active cases

Affected cases

Death

Recovered

Submit

#### Enter the name of state/city/country to be searched

State Name

Submit

## STEPS TO RUN THE Daap

truffle Ethereum - Covid19\_blockchain

Run `npm install truffle-hdwallet-provider`

For first run the following commands

1. Run `truffle compile`
2. Run `truffle develop`, then a list of 10 accounts will come ,take any account and copy it.Also remember the private key corresponding to that account.  
Paste the account on <https://faucet.ropsten.be> and click on send ether. This will give us free ether.
3. Run `truffle migrate --reset --network ropsten`
4. Run `npm install`
5. Run `npm start` // This will run second webpage at <http://localhost:8080/>
6. Run `cd st`
7. Run `npm install`
8. Run `npm run build`
9. Run `npm start` // This will run first webpage
10. On metamash select Ropsten test network
11. Import account, enter the private key from step 3.

In this local hosting Method, you can access the daap at <http://localhost:8081/>

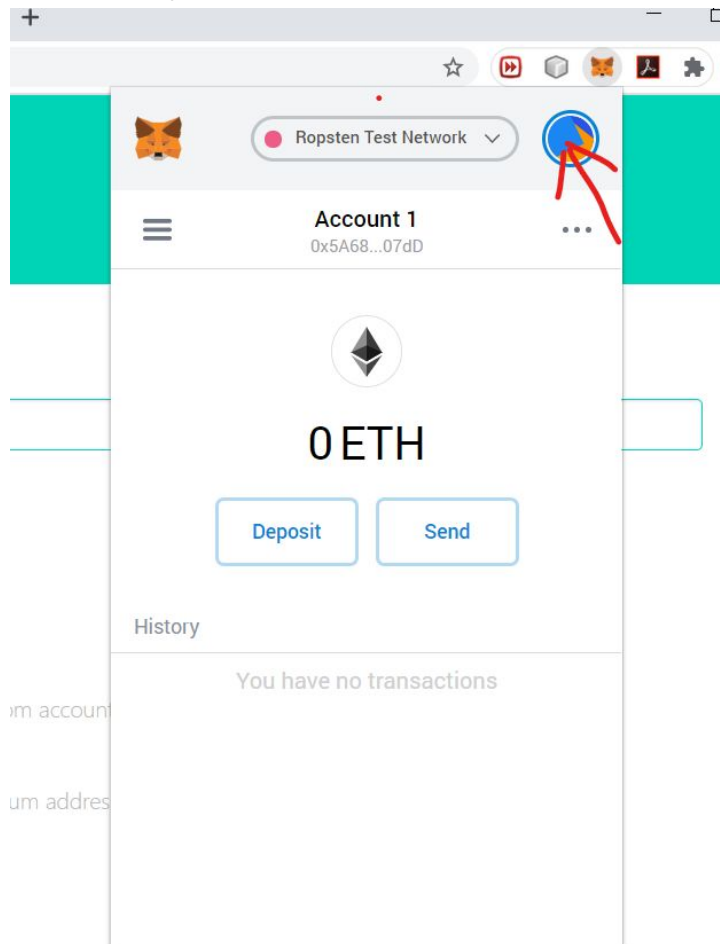


# METAMASK

Metamash extension from the below link in google chrome:

<https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en>

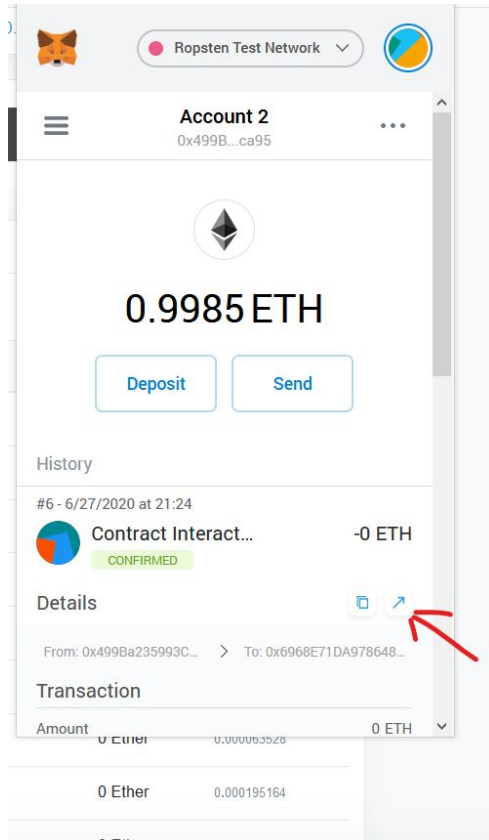
In metamash select Ropsten test network, then import a account, you need to enter private key to import an account. For time being I am providing a private key.



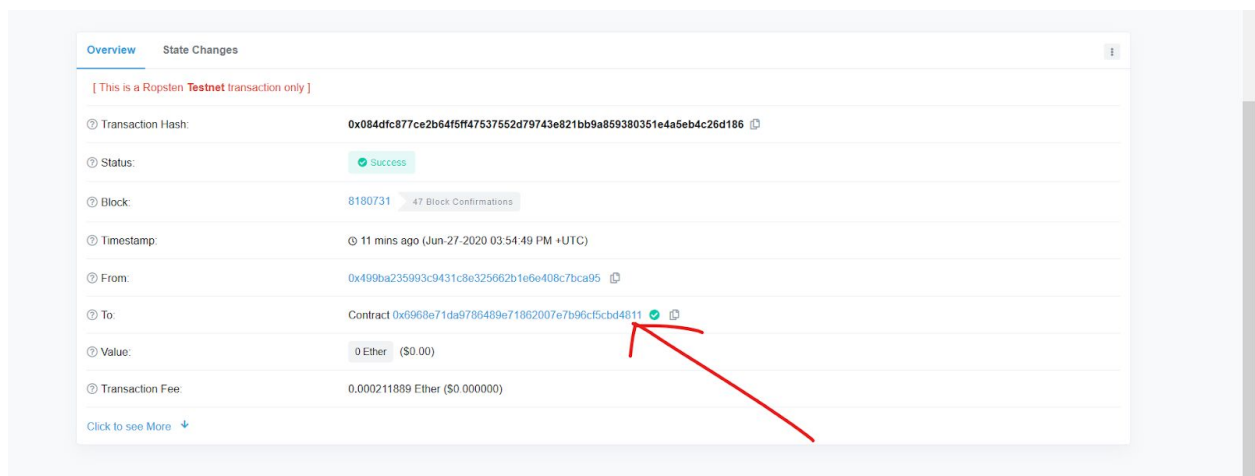
Click on the mark, you will get import account option. Enter the below private key, you will get a ropsten account with 1 ether.

90bb16ac4a4e891ff62bda26d1153a0c2b513757b49b15bd0c5e2f001903c453

After making a transaction, you can go on metamash and view the transaction on etherscam.



Click on the arrow to view on etherscam. You will see your current transaction.



To check all other transaction, You can click on the arrow

OverviewState Changes

[ This is a Ropsten Testnet transaction only ]

Transaction Hash:

0x084dfc877ce2b64f5ff47537552d79743e821bb9a859380351e4a5eb4c26d186

Status:

Success

Block:

818073147 Block Confirmations

Timestamp:

11 mins ago (Jun-27-2020 03:54:49 PM +UTC)

From:

0x409ba235993c9431c8e325662b1e6e408c7bca95

To:

Contract 0x0968e71da9786489e71862007e7b96c5cbd4811

Value:

0 Ether (\$0.00)

Transaction Fee:

0.000211889 Ether (\$0.000000)

Click to see More

## SIGNING A TRANSACTION

ANY TRANSACTION TO THE BLOCKCHAIN IS SIGNED FIRST TO AVOID ANY UNWANTED DATA MANIPULATION.

Signature Request

Account:

Account 11

Balance:

2.707354 ETH

Your signature is being requested

Signing this message can have dangerous side effects. Only sign messages from sites you fully trust with your entire account. This dangerous method will be removed in a future version. [Learn more](#)

Message:  
0x4e4247424e3232313131313131

CancelSign



## Hosting Daap using ipfs

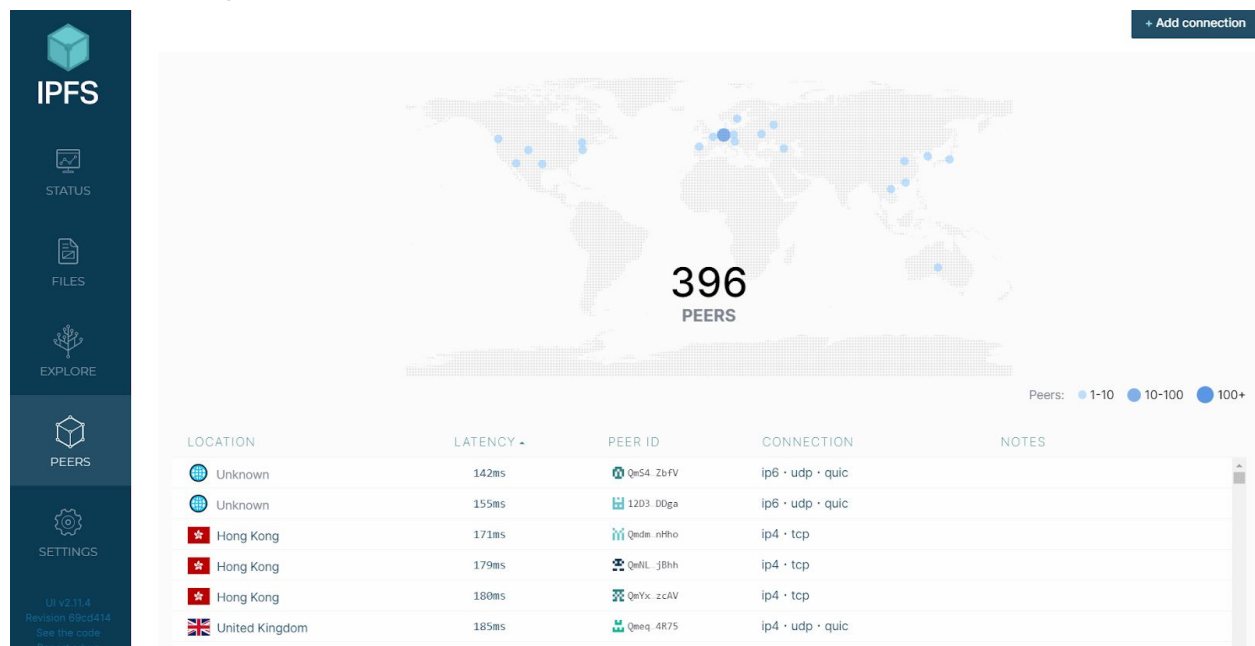
The **InterPlanetary File System (IPFS)** is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS allows users to not only receive but host content, in a similar manner to BitTorrent. As opposed to a centrally located server, IPFS is built around a decentralized system of user-operators who hold a portion of the overall data, creating a resilient system of file storage and sharing. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node who has it using a distributed hash table (DHT).

To install IPFS

<https://github.com/ipfs-shipyard/ipfs-desktop>

It has a link for ipfs desktop setup as well as command prompt installation command.

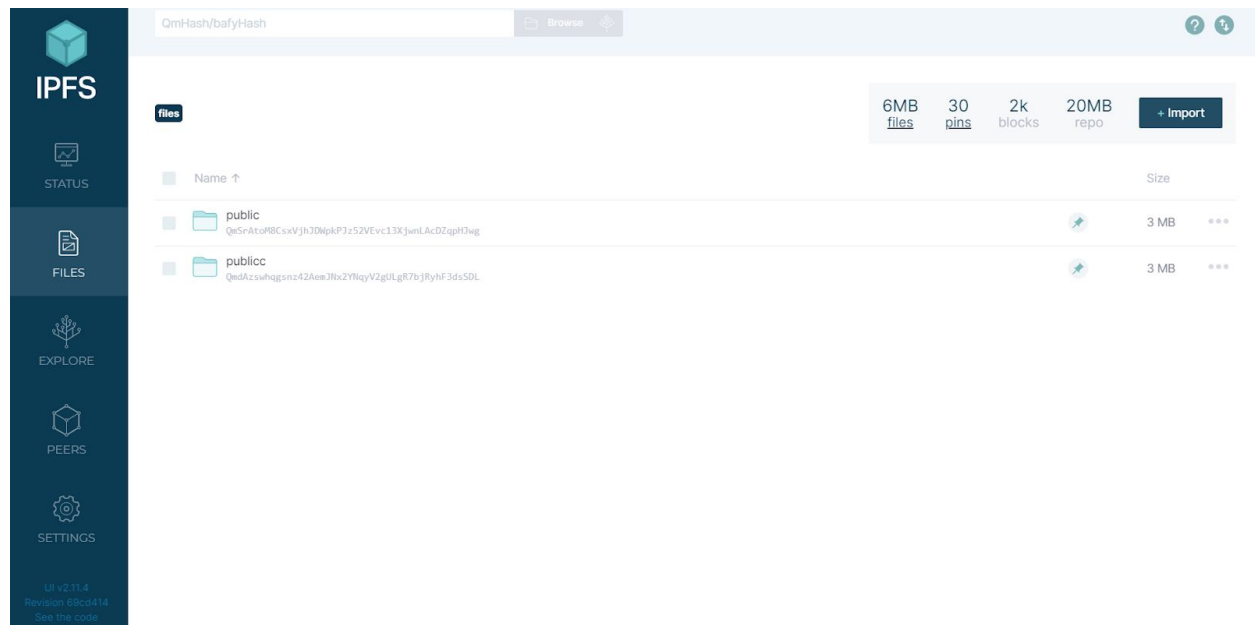
After installing ipfs, open it.



You can see a list of peers that are currently connected to you world wide. If you have installed using command prompt then type these in command prompt

ipfs daemon // To start ipfs as a node  
ipfs swarm peers // To show peers connected to you

Then, import both the public folders ( one in the public daap folder, other inside the st directory).



If you are using command prompt ipfs , Type

Ipfs add -r Folder name

You will see a hash for your folder. To access the website, type

<https://gateway.ipfs.io/ipfs/> Paster hash for Public folder here.

Since, ipfs creates hash for the public folder , so any changes made inside the contents of the folder will change the hash of the public folder. So, the website will also change as the website address has hash of the Public folder.

To solve this issue, we publish our file to ipns (Inter Planetary Naming System ). The hash for ipns remains same . Therefore, the website address also doesnot change on changing the contents of the files. A *name* in IPNS is the hash of a public key. It is associated with a record containing

information about the hash it links to that is signed by the corresponding private key. New records can be signed and published at any time.

To publish on ipns type in command prompt  
Ipfs name publish // Paste public hash here \\

You will get a new hash.

```
PS F:\daap public zip> ipfs add -r public
2.77 MiB / 2.77 MiB [=====] 99.87%
added QmQacSmrjei4aitn1t3jKSh9YjCApY5RvqeqAh9J3pLKME public/bundle.js
2.77 MiB / 2.77 MiB [=====] 100.00%
added QmWoJ77JwyYdmt25ugPTYH5q7Zp9UcYBR4zzTNnLF44Mg9 public/index.html
2.77 MiB / 2.77 MiB [=====] 100.00%
added QmSrAtoM8CsxVjhJDWpkPJz52VEvc13XjwnLAcDZqpHJwg public
2.77 MiB / 2.77 MiB [=====] 100.00%
PS F:\daap public zip> ipfs name publish QmSrAtoM8CsxVjhJDWpkPJz52VEvc13XjwnLAcDZqpHJwg
Published to k2k4r810kzbbesrv172c1me8b0mn7tx210bsvpqnylir0ncuosmakm4: /ipfs/QmSrAtoM8CsxVjhJDWpkPJz52VEvc13XjwnLAcDZqpHJwg
PS F:\daap public zip>
```

You can access the website using  
gateway.ipfs.io/ipns/ Paste the new hash here

## Accessing website Using Pinata as a pinning service for ipfs

The Interplanetary File System, or IPFS, is a distributed storage network made up of “nodes” or computers all over the world where people and apps are storing and sharing data. When an IPFS node retrieves data from the network it keeps a local cache of that data for future usage, taking up space on that particular IPFS node. IPFS nodes frequently clear this cache out in order to make room for new content.

But if we want to make sure that certain content will never be deleted. The act of saving data on an IPFS node is often referred to as “pinning”. It’s also the reason **Pinata** is named Pinata!

When we “pin” data on an IPFS node, you are telling that node that the data is important and it should be saved. Pinning prevents important data from being deleted from your node when the clearing process happens. However, you can only control and pin data on your node(s). You can not force other nodes on the IPFS network to pin your content for you. So, to guarantee your content stays pinned, you have to run your own IPFS nodes.

An IPFS pinning service is an IPFS node or collection of IPFS nodes dedicated to pinning other peoples’ and apps’ content on IPFS. Below, we can see four advantages of why using an IPFS pinning service can help you leverage the IPFS network with or without running your own IPFS nodes.

- 1.> Speed
- 2.>Uptime
- 3.>Space
- 4.>Redundancy

You can check more from this link.

[https://medium.com/pinata/what-is-an-ipfs-pinning-service-f6ed4cd7e475#:~:text=When%20you%20%E2%80%9Cpin%E2%80%9D%20data%20on,on%20your%20node\(s\).](https://medium.com/pinata/what-is-an-ipfs-pinning-service-f6ed4cd7e475#:~:text=When%20you%20%E2%80%9Cpin%E2%80%9D%20data%20on,on%20your%20node(s).)

So, we have used Pinata to keep our website online without keeping the ipfs node online.

<https://pinata.cloud/>

Create an account and then login with the same.

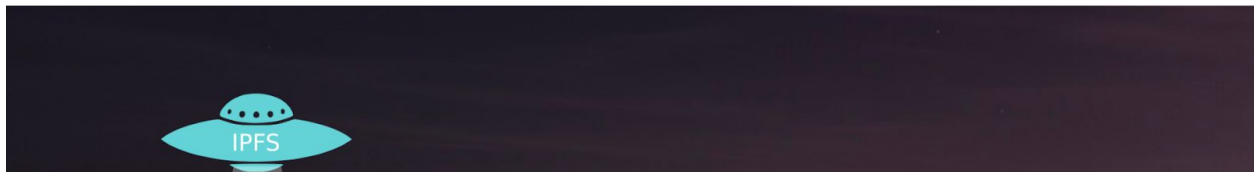
Then , go to Pinata upload and upload your files.



## PINATA UPLOAD

[Upload File](#) [Upload Directory](#) [Pin By Hash](#)

☐ Preserve filename (wrap with directory) ?



Then, go to Pin explorer . You can see the fash for your uploaded files.

## PIN EXPLORER

Hash Contains

Name Contains

Pin Status

Pin Date Range

Unpin Date Range

Total Pins: 2  
Total Size: 5.9 MB

Name	IPFS Hash	Size	Date Pinned	Date Unpinned		
publicc	QmdAzswhqgsnz42AemJNx2YNqyV2gULgR7bjRyhF3dsSDL	3.0 MB	2021-01-16T15:12:56.766Z	Currently Pinned	<input checked="" type="checkbox"/>	<input data-bbox="1385 1150 1409 1182" type="button" value="..."/>
public	QmSrAtoM8CsxVjhJDWpkPjz52VEvc13XjwnLacDZqpHJwg	2.9 MB	2021-01-16T15:12:00.670Z	Currently Pinned	<input checked="" type="checkbox"/>	<input data-bbox="1385 1192 1409 1224" type="button" value="..."/>

Click on the hash , it will open the website.

Currently, the Daap is accessible at

<https://gateway.pinata.cloud/ipfs/QmdAzswhqgsnz42AemJNx2YNqyV2gULgR7bjRyhF3dsSDL/>

**Note--:** Since, Pinata is a pinning service for ipfs, Therefore if changes are made to the public folder, the hash will change. Hence, the above link will also get changed.

The project can be downloaded at

<https://github.com/Puspeshsingh7/Agricultural-Block-chain>

[https://github.com/Puspeshsingh7/Covid\\_19\\_Local\\_daap](https://github.com/Puspeshsingh7/Covid_19_Local_daap)

[https://github.com/Puspeshsingh7/Covid19\\_Public\\_Testnet\\_daap](https://github.com/Puspeshsingh7/Covid19_Public_Testnet_daap)