**Innovation Lab - Report**

Content:

* [Building blockchain which keep the details of various agricultural transaction](#_8w96o8fj2i01)
  + [Modules Imported](#_9j4zkd1oxxwi)
  + [Class Blockchain](#_jlacgfpxuk5c)
  + [Mining the blockchain](#_zevmyyyoz3g6)
  + [Decentralising the blockchain](#_4d7rpg2znq6x)
* [Covid-19 Data Collection dapp](#_ppb3y6ihielm)
  + [Writing smart contracts](#_1lsk16m75wl)
  + [Building web interface to connect with blockchain](#_jo5mnt6hdnu4)
  + [Building web interface for normal user](#_kuvxjy4y1t1o)
  + [Hosting blockchain locally on Ropsten](#_csvjwsw2gvpm)
  + [Hosting blockchain on ipfs server](#_6g1m7bh37tmm)
  + [Accessing website Using Pinata as a pinning service for ipfs](#_6ch13k5b7jf9)

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

IMPLEMENTATION:-

## **MODULES IMPORTED**

* 1. Datetime
  2. Hashlib
  3. Json
  4. flask—Flask, jsonify, requests
  5. Requests
  6. Uuid—uuid4
  7. Urllib.parse--urlparse
  8. crypto.publickey—RSA

## **CLASS BLOCKCHAIN**

* 1. **Self- chain**(emptylist) , farmer\_details(emptylist), nodes(set), create\_block(function to create the genesis block)
  2. **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.

* 1. **get\_previous\_block**- function, parameter(self)

Description-It returns the last block of the chain.

* 1. **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.

* 1. **Hash**- function,parameters(self,block)

Description- It returns the hash of the block using sha256

* 1. **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^2-proof of previous block^2 contains 4 leading zeroes or not. if no, then the chain is not valid.

* 1. **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 append a dictionary containing all these information in the hash format to the chain farmer\_details and returns the index of the new block.

* 1. **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.

* 1. **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.

## **MINING THE BLOCKCHAIN**

* 1. **App**- It uses the flask to create a web app.
  2. **node\_address**- It creates an address for the node using uuid4 and removing –
  3. **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.

* 1. **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,jasonifies it and returns it.

* 1. **get\_chain**- @app.route('/get\_chain', methods = ['GET'])

Description- It creates the response containing the blockchain.chain and its length,jasonifies it and returns it.

* 1. **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.
  2. **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.

## **DECENTRALIZING THE BLOCKCHAIN**

* 1. **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.

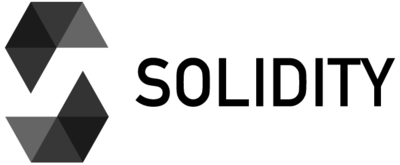
* 1. **replace\_chain**- @app.route('/replace\_chain', methods = ['GET'])

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.

* 1. **App.run** (host = '0.0.0.0', port = 5001)- Different ports are provided for the different nodes.
  2. **nodes.json**- It is a json file which contains the format in which the post request for connect\_node has to be entered.
  3. **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++;

}

1. 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

);

}

1. 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;

}

1. 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];

}

1. 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:-

Truffle: Used for creating a development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM).

Web3: web3.js is used to interact with remote ethereum node using HTTP, IPC or WebSocket.

Webpack: It is used to bundle all js files into one.

## **Building web interface for normal user**

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

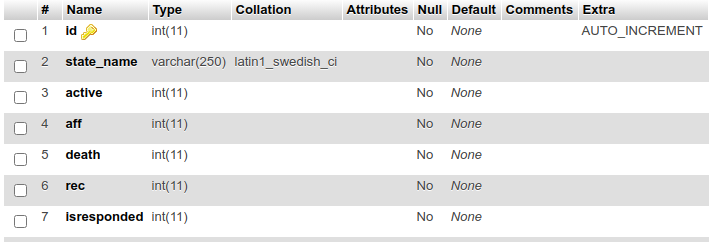
It is hosted using website

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

And is live at domain

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

Structure of database

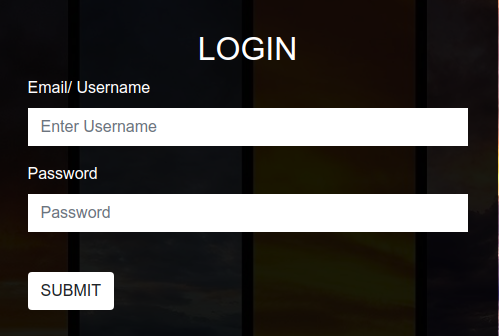


Special users can login to check the update/add requests, and add to blockchain after validating.

Demo Credentials for special user:

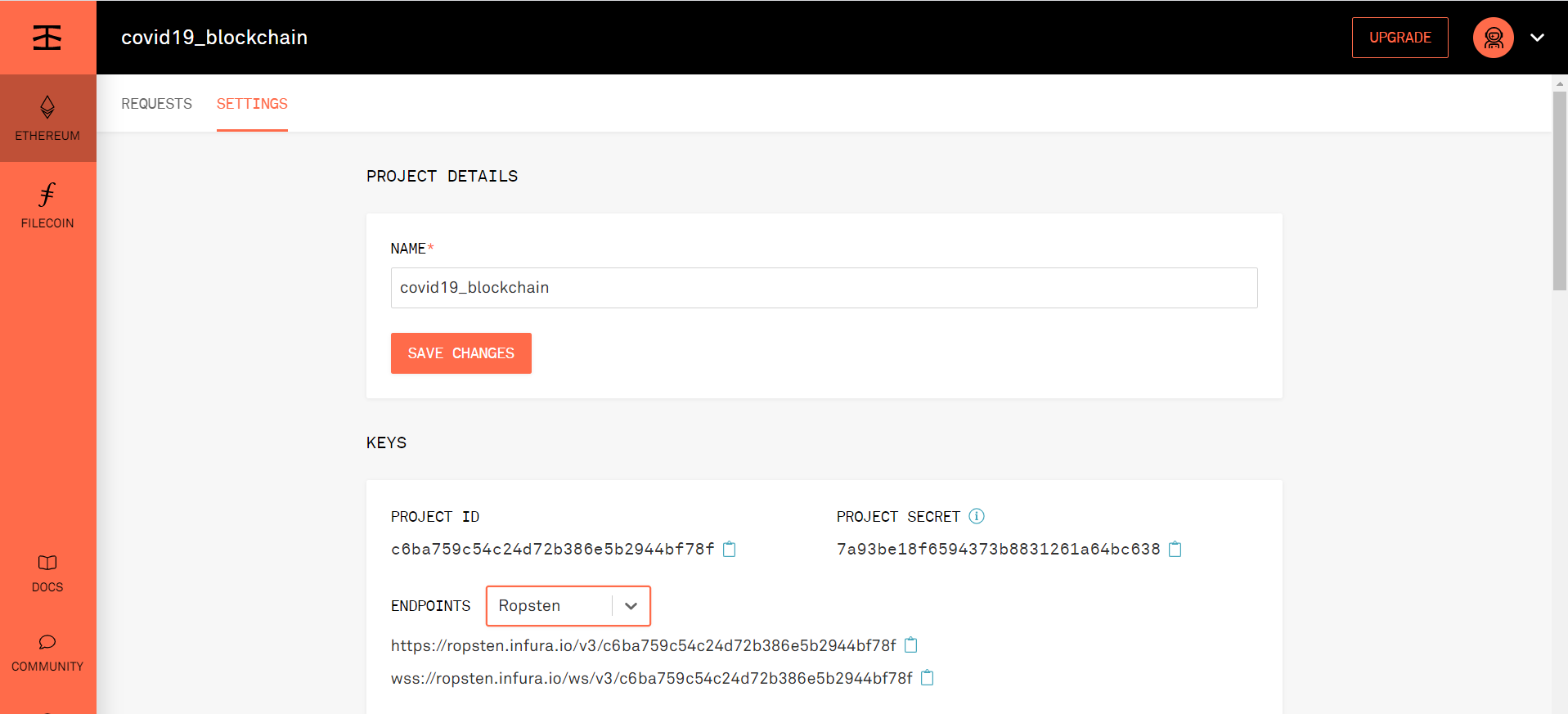
Username: pranaykgupta

Password: qwerty



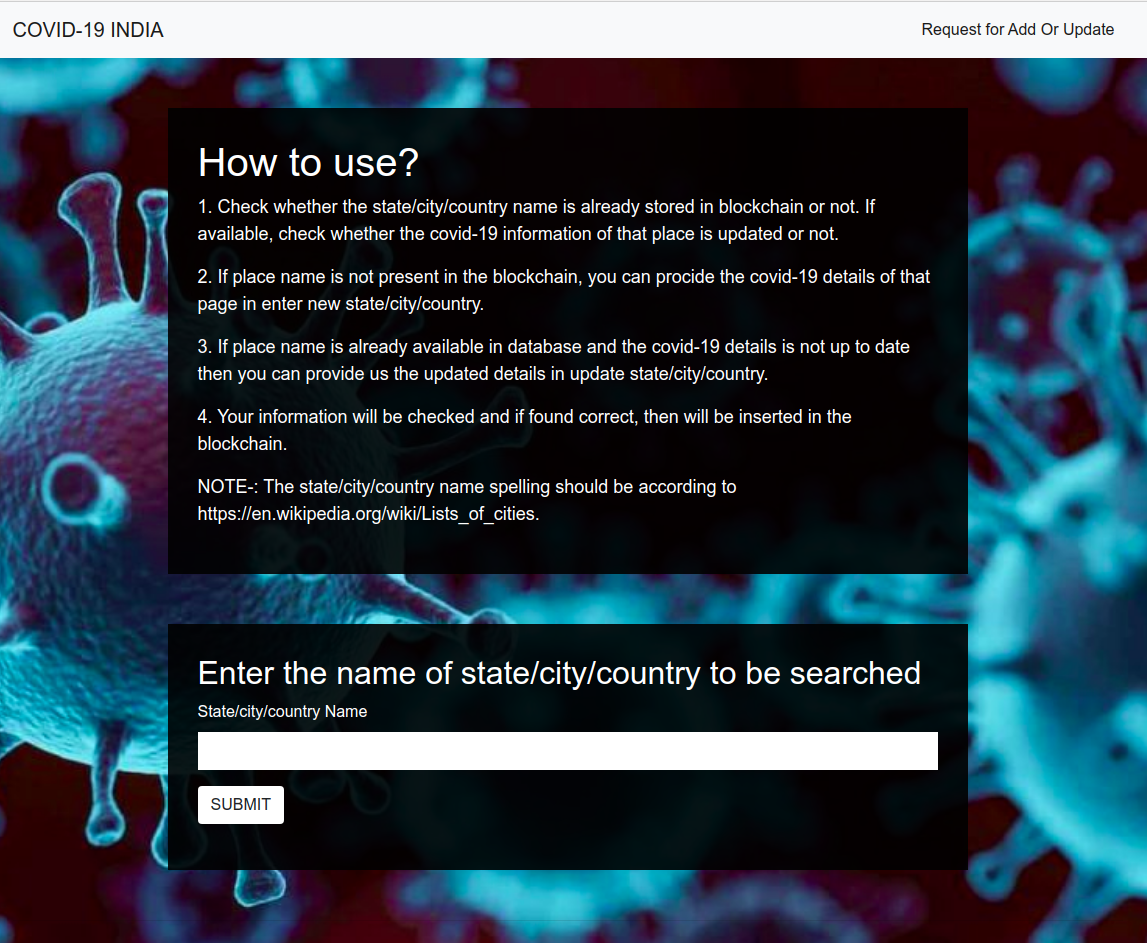
## **Hosting blockchain locally on Ropsten ( Ethereum Public Network)**

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.



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**



**SECOND WEB PAGE**

****

**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.

1. Run `truffle migrate --reset --network ropsten`
2. Run `npm install`
3. Run `npm start` // This will run second webpage at

http://localhost:8080/

1. Run `cd st`
2. Run `npm install`
3. Run `npm run build`
4. Run `npm start` // This will run first webpage
5. On metamash select Ropsten test network
6. Import account, enter the private key from step 3.

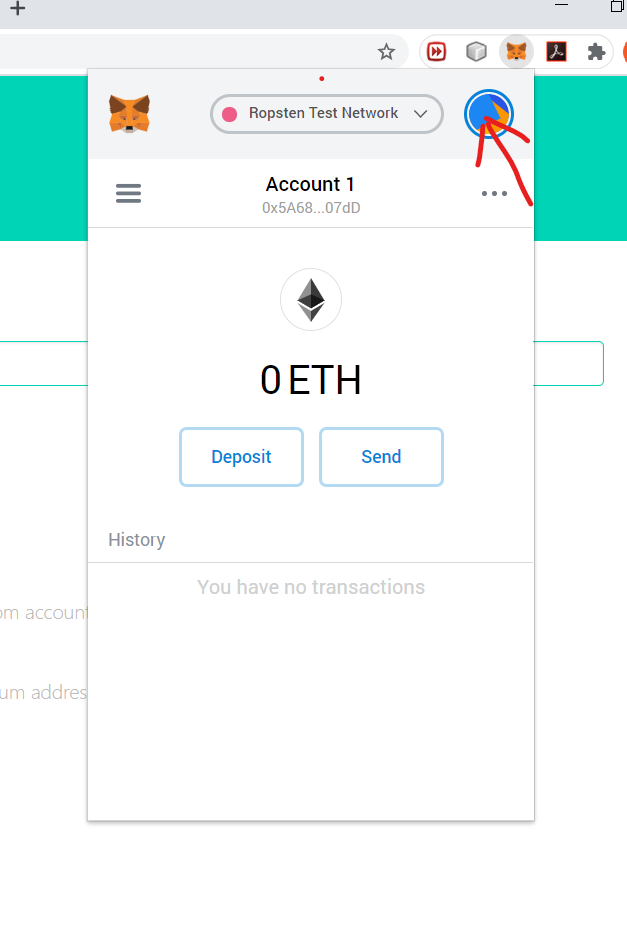
In this local hosting Method, you can access the daap at http://localhost:8081/



Metamash extension from the below link in google chrome:

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

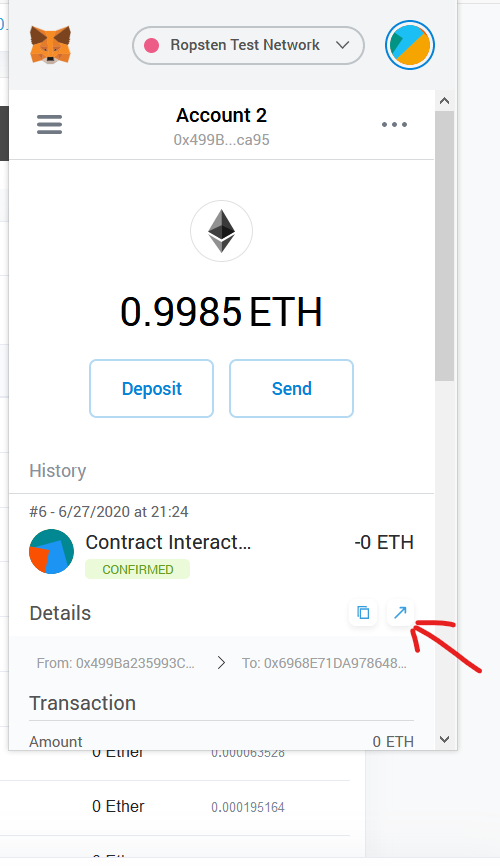
In metamash select Ropsten test network, then import a account, you need to enter private key to import an account. For time being I am providing a private key.



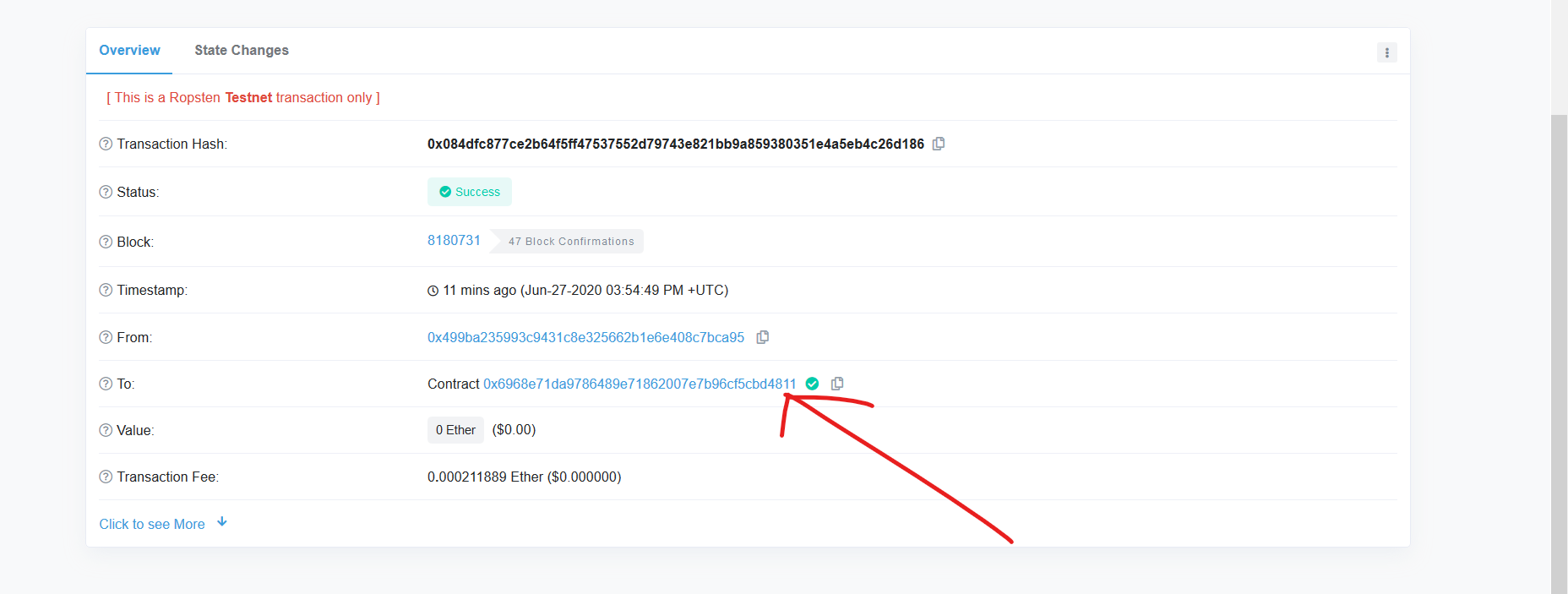
Click on the mark, you will get import account option. Enter the below private key, you will get a ropsten account with 1 ether.

90bb16ac4a4e891ff62bda26d1153a0c2b513757b49b15bd0c5e2f001903c453

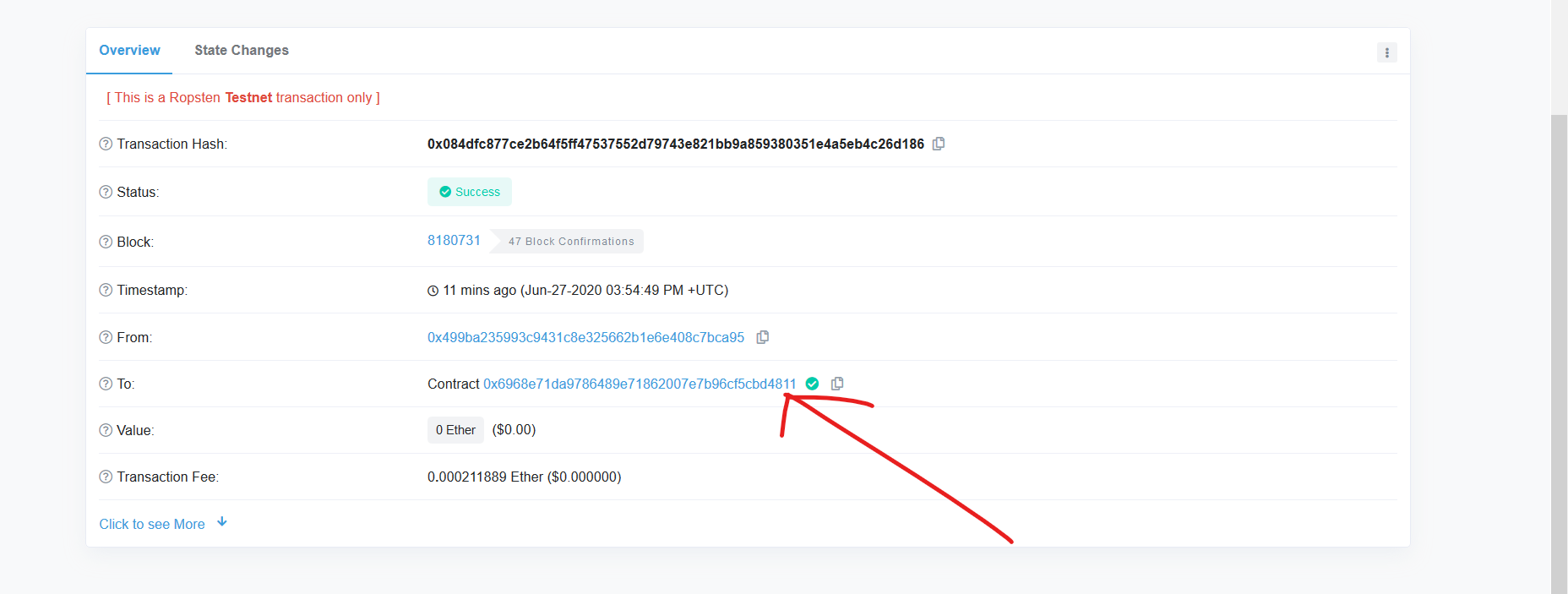
After making a transaction, you can go on metamash and view the transaction on etherscam.



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

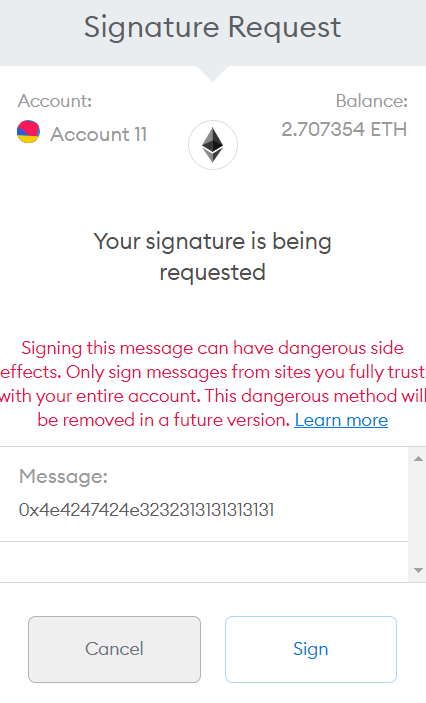


To check all other transaction, You can click on the arrow



**SIGNING A TRANSACTION**

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



## **Hosting Daap using ipfs**

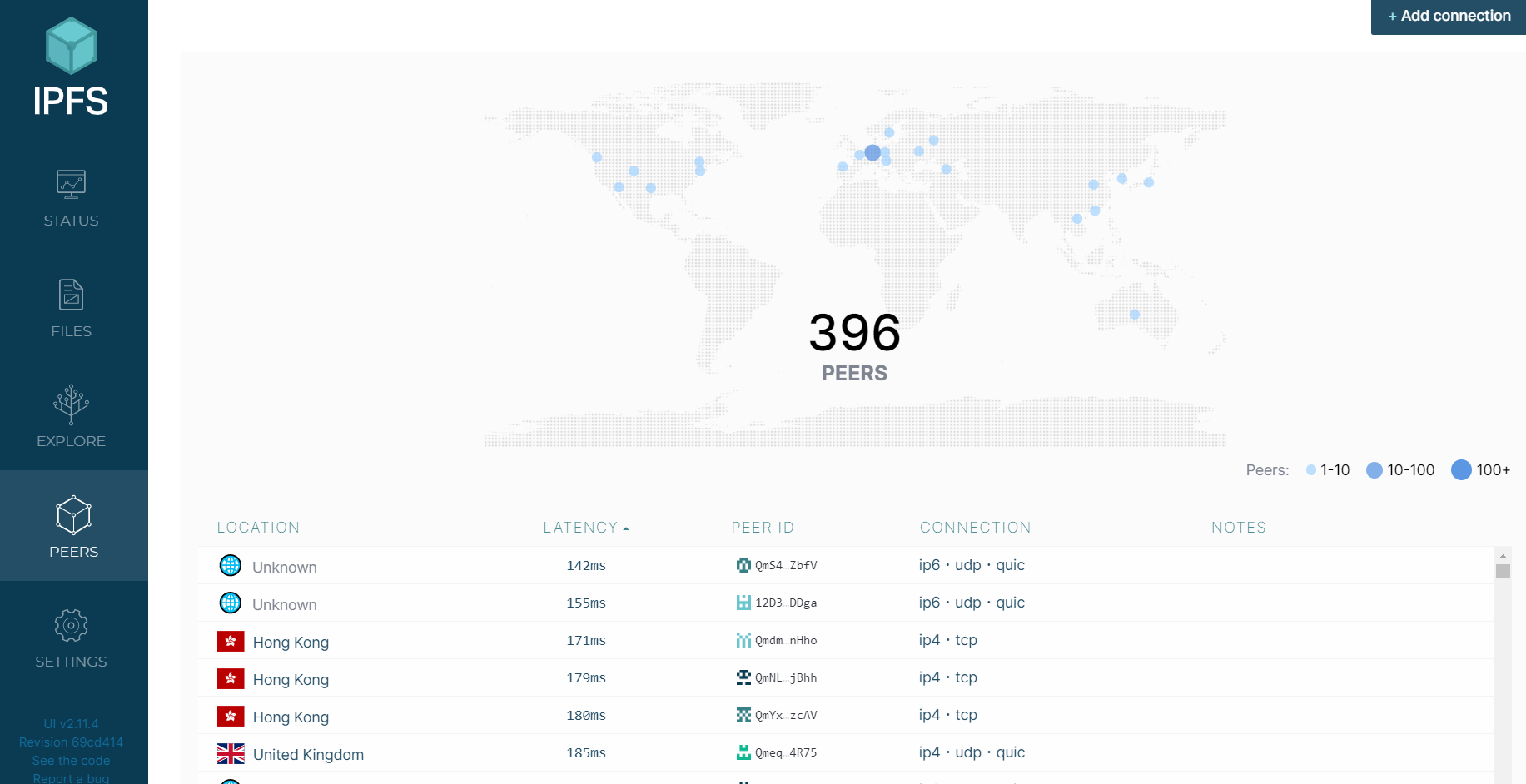
The **InterPlanetary File System** (**IPFS**) is a [protocol](https://en.wikipedia.org/wiki/Communications_protocol) and [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer) network for storing and sharing data in a [distributed file system](https://en.wikipedia.org/wiki/Distributed_file_system). IPFS allows users to not only receive but host content, in a similar manner to [BitTorrent](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/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.

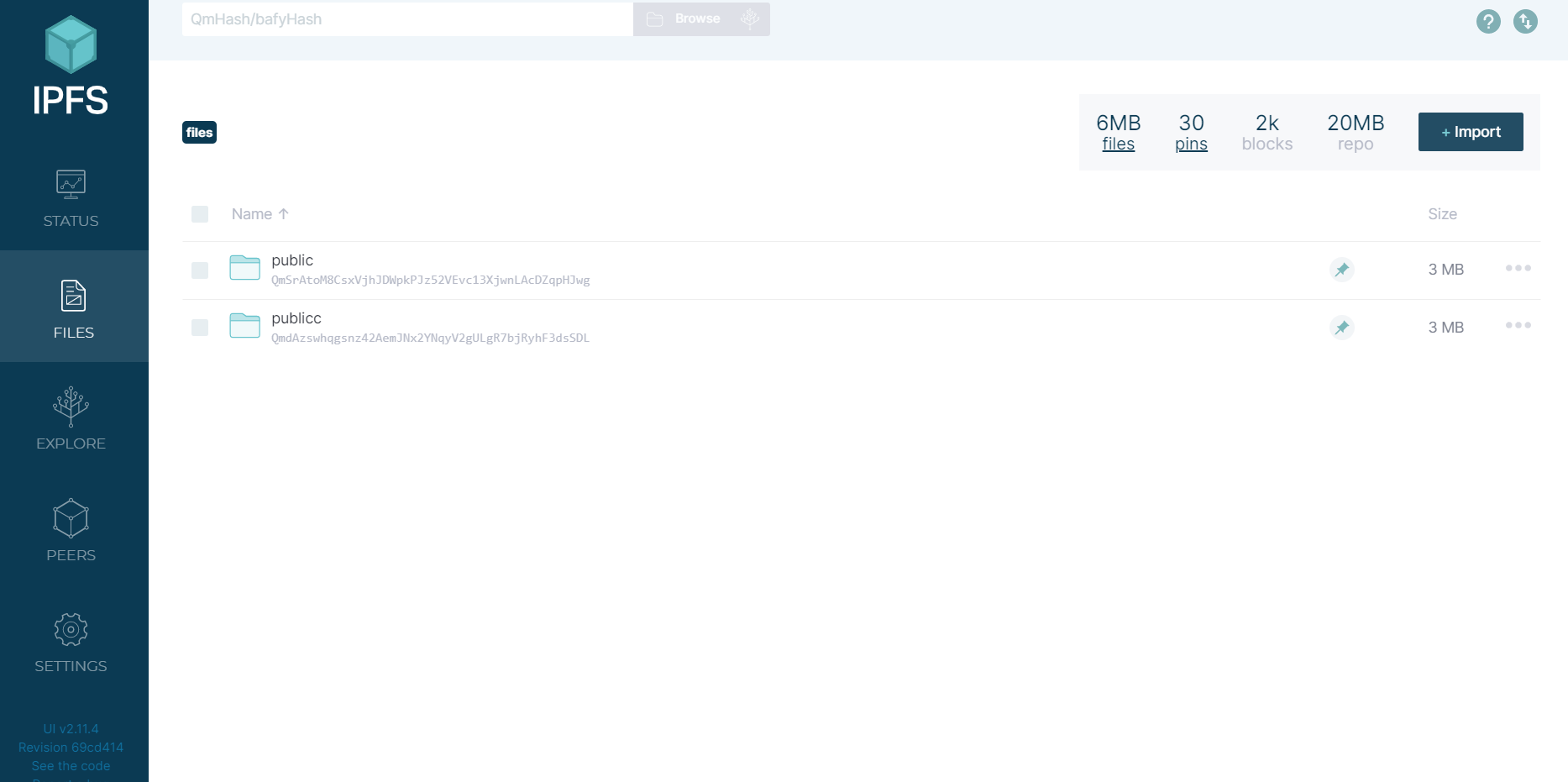


You can see a list of peers that are currently connected to you world wide. If you have installed using command prompt then type these in command prompt

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.

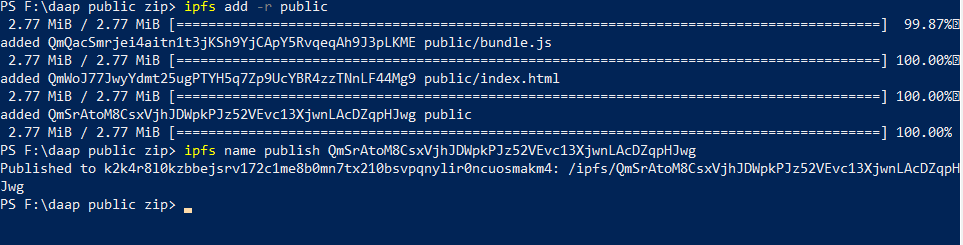
Since, ipfs creates hash for the public folder , so any changes made inside the contents of the folder will change the hash of the public folder. So, the website will also change as the website address has hash of the Public folder.

To solve this issue, we publish our file to ipns (Inter Planetory 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](https://docs.ipfs.io/concepts/hashing) 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.



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](https://ipfs.io/), 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 ***Pin***ata is named [Pinata](http://pinata.cloud/)!

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.

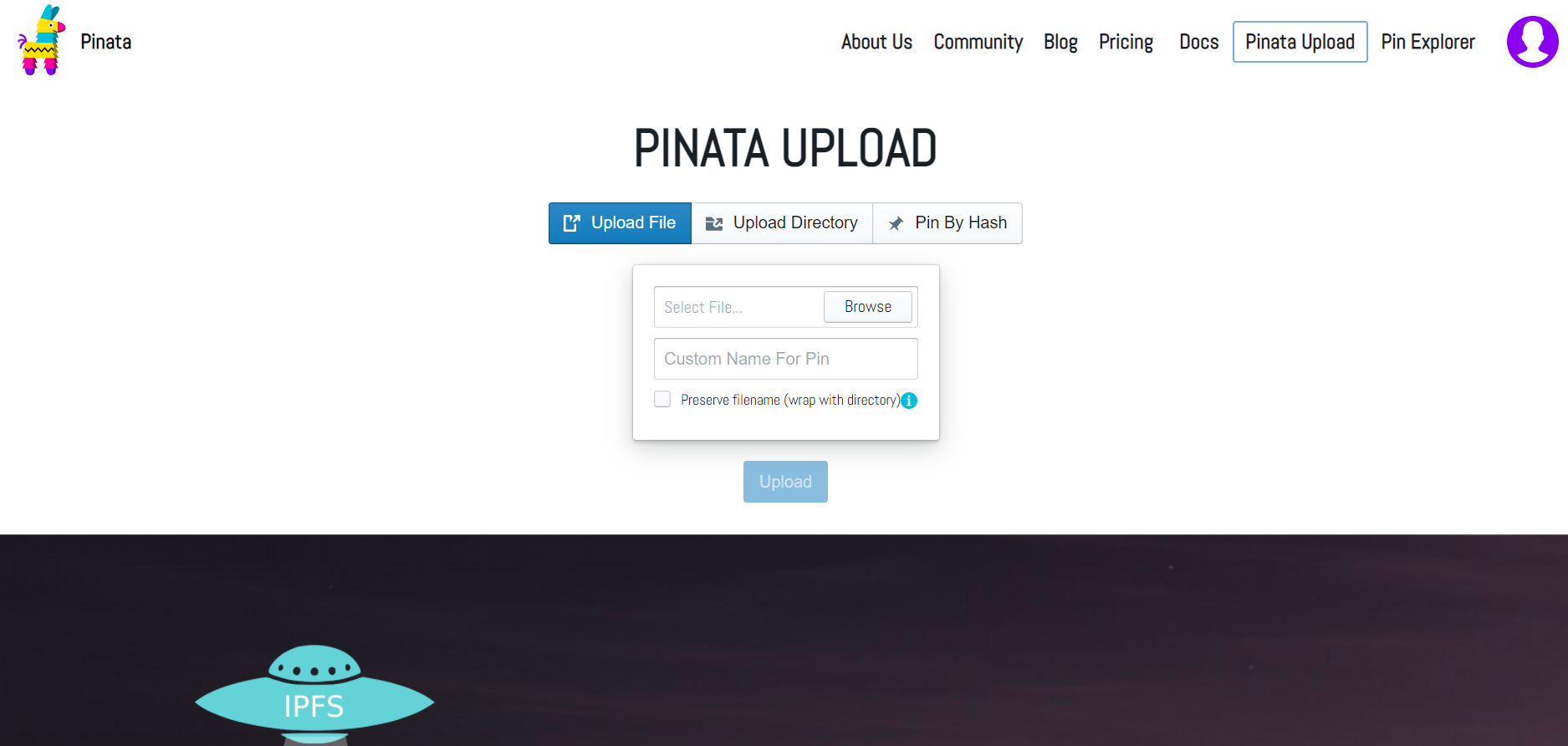
<https://medium.com/pinata/what-is-an-ipfs-pinning-service-f6ed4cd7e475#:~:text=When%20you%20%E2%80%9Cpin%E2%80%9D%20data%20on,on%20your%20node(s)>.

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

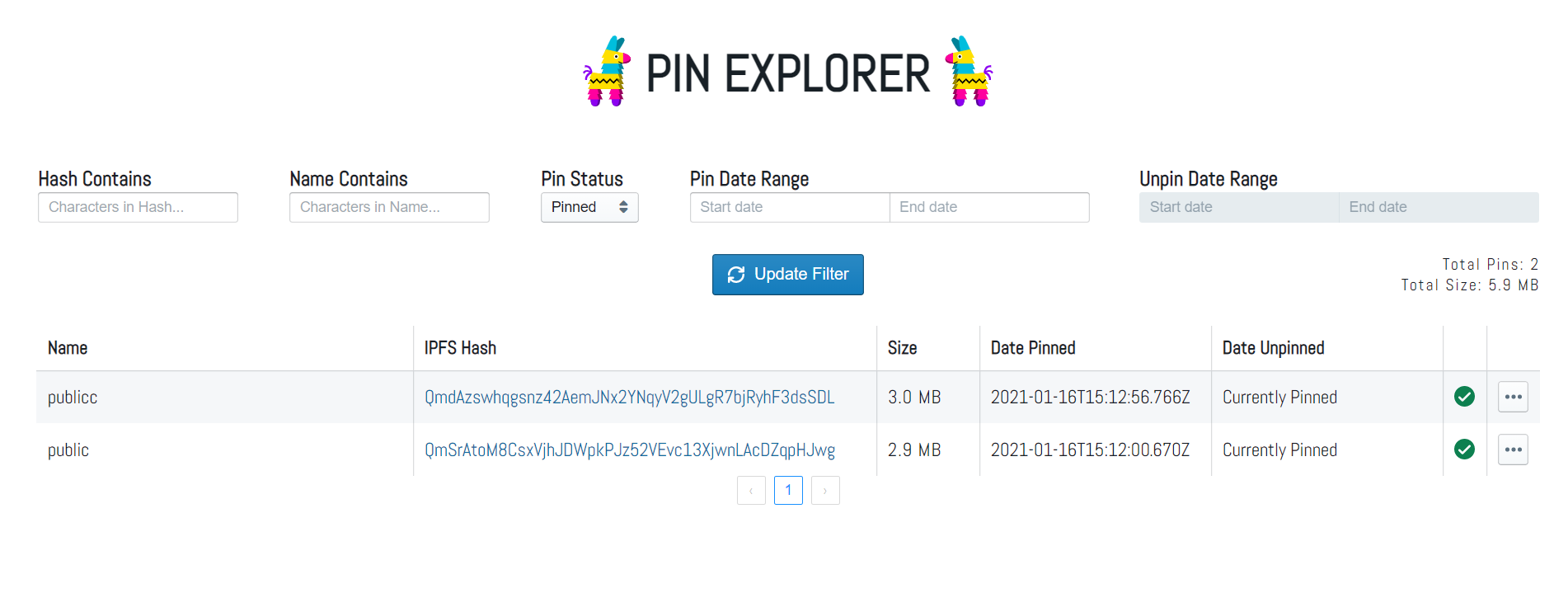
<https://pinata.cloud/>

Create an account and then login with the same.

Then , go to Pinata upload and upload your files.



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



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.