Interplanetary File System with Python

python™ + IPFS

What we will be talking about

- What is IPFS?
- How is it different from Databases and Torrents?
- Installing in your system and running a peer
- Connecting to IPFS via Python
- Uploading to IPFS using Python
- Fetching from IPFS via Python
- Where to use IPFS?

Pre-requisites

- Knowledge of how internet communication works (Esp. protocols like HTTP).
- Knowledge of what a database is.
- Knowledge of basic data structures.
- Basics of Python (Functions, Importing libraries).

What is IPFS?

- It is a file storage system of sorts
- Instead of a central database, all the nodes participating are a server for data storage
- Data is requested using the cryptographic hash of that data instead of IP address
- This means that if two same files are uploaded, they will have the same hash, thus no duplication
- Each node/server/peer (same thing!) has an ID, we will call it peerID

How does IPFS store data?

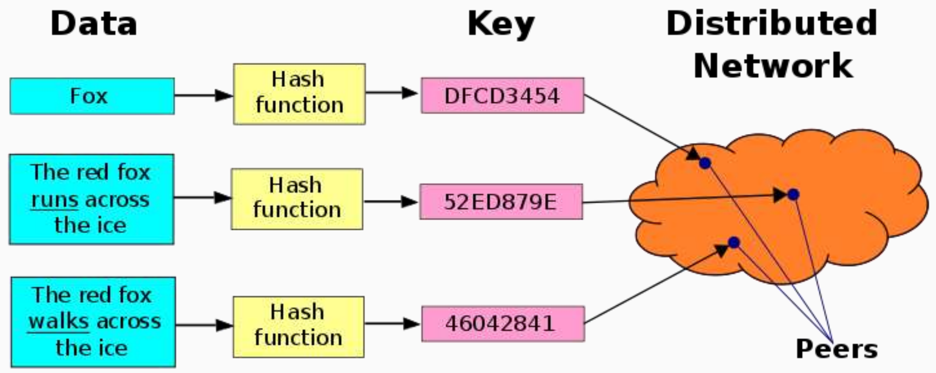
- When you add any content on IPFS network, the data is split into chunks of 256Kb
- These chunks are then distributed to various nodes on network which have there hash closest to peerId
- Now suppose there are 3 peers with the peer IDs 40a, 10ab. 603c
- So suppose a 768 KB data is stored on the IPFS network
- Now the data is broken into 3 chunks of 256kb and the hash of each chunk is calculated

How does IPFS store data?

- Now suppose there are 3 peers with the peer IDs 40a, 10ab. 603c
- Suppose the 3 hashes of the files are 42ab, 9bc, 600abc
- These three chunks are then sent to the peer with the ID closest lexically to the hash of the respective chunk
- Then the address is updated in the Distributed hash table

What is a Distributed Hash Table?

- A distributed hash table is just a hash table which helps in locating the value of asked key
- While theoretically not the same, it helps to think of a hash table as a dictionary
- Now assume that this hash-Table allowed you to query From multiple devices



So how does one retrieve data on IPFS?

- So to fetch data, you request it by typing: “/ipfs/<hash of data>” on a node running IPFS
- It will fetch the data and display it in your web browser using browser’s protocol (HTTP)
- You can also fetch it using a gateway (Let’s talk about it later!)

Let’s start coding!

Download and Start IPFS client

- There are two major implementations of IPFS: Go and JavaScript, We will use Go since it has more peers
- To download IPFS for your system, visit https://docs.ipfs.io/install/
- Now you can use Command Line IPFS or GUI IPFS client, doesn't matter since under the hood, they basically run the same thing
- Now start the client
- While you're on it, Google IPFS companion for your browser, and install it as a browser extension

Install the Python modules

- We need to install “ipfshttpclient”
- pip install ipfshttpclient
- Otherwise you can build from source:

```
git clone https://github.com/ipfs/py-ipfs-http-client.git
cd py-ipfs-http-client
flit install --pth-fil
```

Connecting to the IPFS network from Python

```
import ipfshttpclient
client = ipfshttpclient.connect('/ip4/127.0.0.1/tcp/5001/http')
```

Uploading to IPFS using Python

- First, let’s make a file, I will call it “test.txt”
- Then, let’s connect to IPFS like we did last time

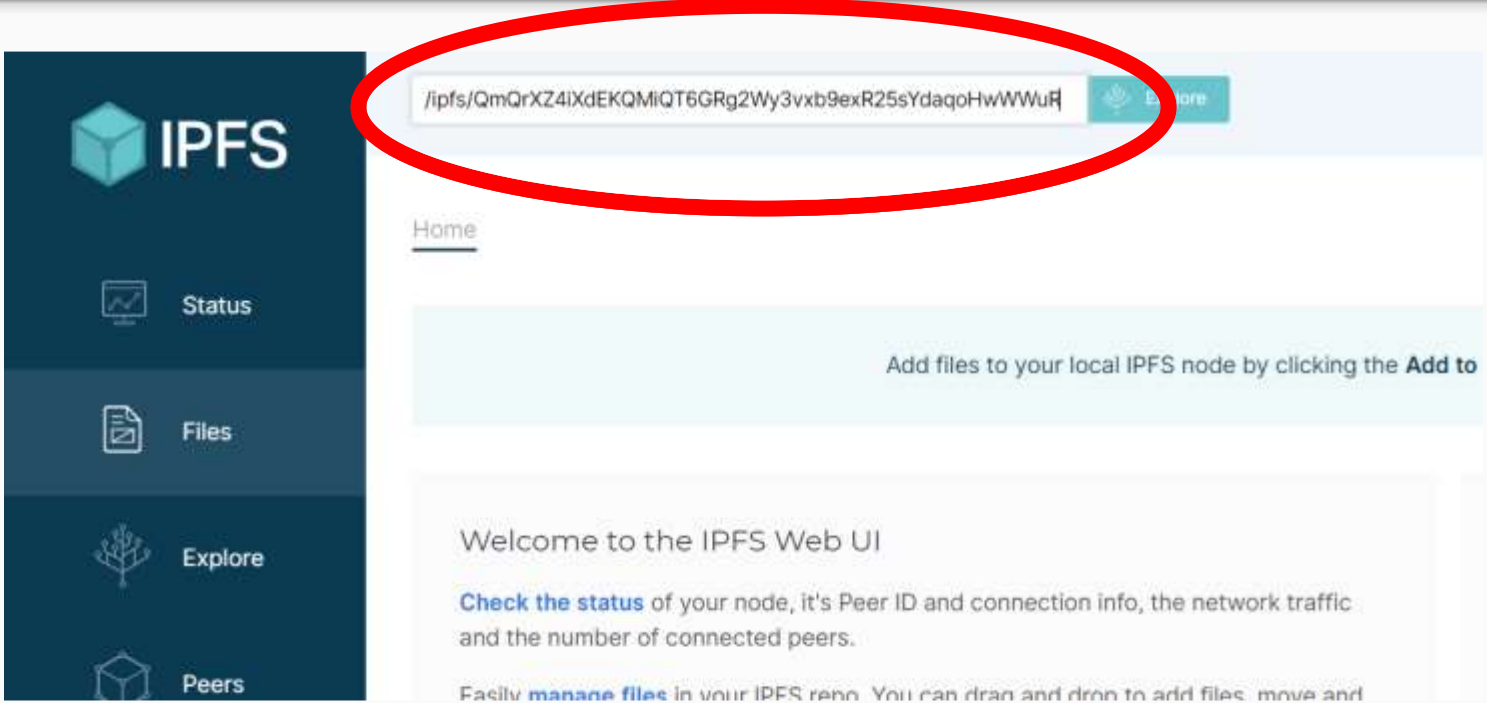
-
  ```
  import ipfshttpclient
  client = ipfshttpclient.connect('/ip4/127.0.0.1/tcp/5001/http')
  res = client.add('test.txt')
  ```
- Now we can do `print(res)` to print the metadata of the uploaded file

Uploading to IPFS using Python

- In test.txt, I had written “Test”
- The output is as follows:

```
{'Name': 'test.txt',
'Hash':'QmQrXZ4iXdEKQMiQT6GRg2Wy3vxb9exR25sYdaqoHwWWuR',
'Size': '12'}
```

Let’s check our file!


/ipfs/QmQrXZ4iXdEKQMiQT6GRg2Wy3vxb9exR25sYdaqoHwWWuR

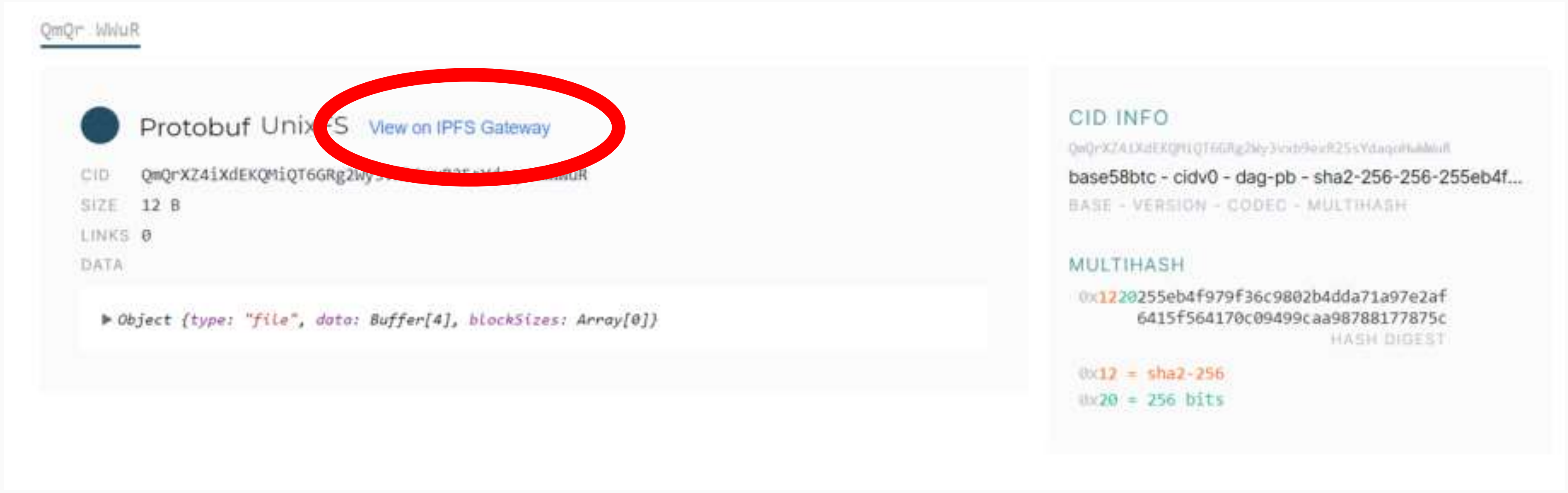
Add files to your local IPFS node by clicking the **Add to**

Welcome to the IPFS Web UI

Check the status of your node, it's Peer ID and connection info, the network traffic and the number of connected peers.

Easily manage files in your IPFS repo. You can drag and drop to add files, move and

File metadata



QmQr WWuR

Protobuf UnixFS View on IPFS Gateway

```
CID    QmQrXZ4iXdEKQMiQT6GRg2Wy[illegible]WWuR
SIZE   12 B
LINKS  0
DATA
▶ Object {type: "file", data: Buffer[4], blockSizes: Array[0]}
```

CID INFO

QmQrXZ4iXdEKQMiQT6GRg2Wy3Vxb9exR25sYdaqoHuWWuR

base58btc - cidv0 - dag-pb - sha2-256-256-255eb4f...

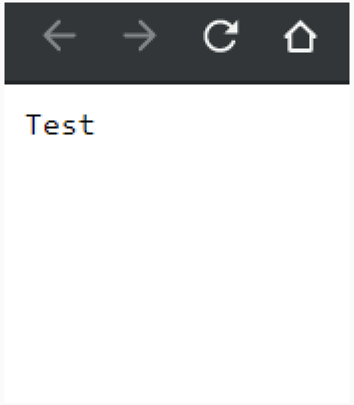
BASE - VERSION - CODEC - MULTIHASH

MULTIHASH

```
0x1220255eb4f979f36c9802b4dda71a97e2af
      6415f564170c09499caa98788177875c
                           HASH DIGEST

0x12 = sha2-256
0x20 = 256 bits
```

Viewing file content from gateway



```
Test
```

IPFS Public Gateways

- You use someone else's system to access IPFS nodes
- A few good, reliable gateways are:

  https://www.eternum.io/
  https://ipfs.infura.io/
  https://cloudflare-ipfs.com/
  https://ipfs.infura.io

Fetching files from IPFS

```
getfile = client.cat(<CONTENT_ID>)
print(getfile.decode('utf-8'))
```

IPFS Use - cases

- For distributed storage in Blockchain application
- For non-tampered web services
- Better content delivery networks

And other exciting things!

Thanks!

I am everywhere as Abhinavmir, thanks for tuning in!