IPFS Dapp - Ethereum

Details about main frameworks and components used:

- **Truffle** - A development framework used for blockchains using the Ethereum Virtual Machine (EVM).
- **Truffle React Box -** Almost ease's our half of the work by providing React components which we can start building using smart contracts.
- **Truffle Ganache** - Easy way to run local private blockchain.
- **Npm -** npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js.
- **Visual Studio Code** - For programming and WSL (Windows Sub-system Linux) terminal support.
- **Solidity** - Solidity is an object-oriented programming language for writing smart contracts.

Output:

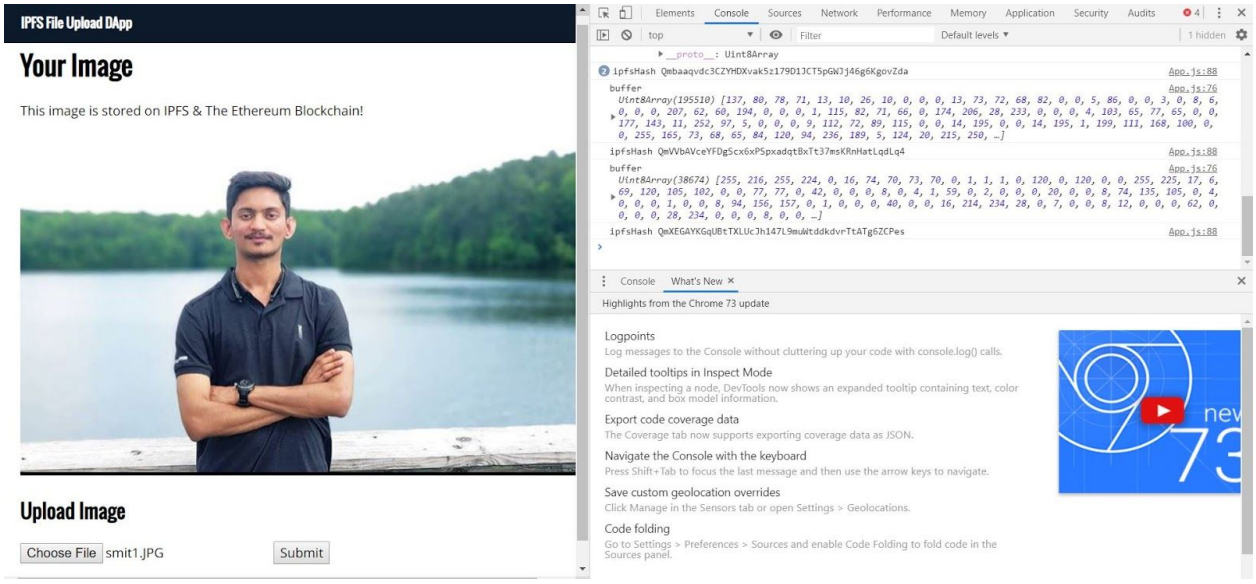


fig1: ipfsHash can be seen in the right corner

```
ipfsHash QmXEGAYKGqUBtTXLUcJh147L9muWtddkdvrTtATg6ZCPes
```

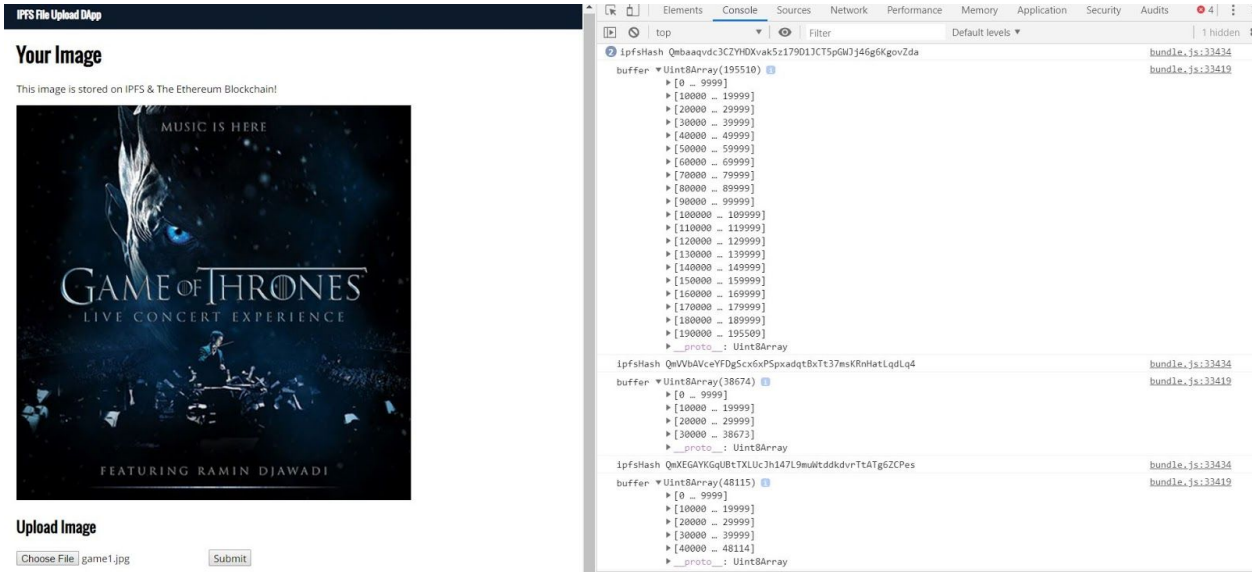


Fig 2: A new ipfsHash can be seen for every new photo that gets uploaded in the Inter Planetary File System

Steps to get started:

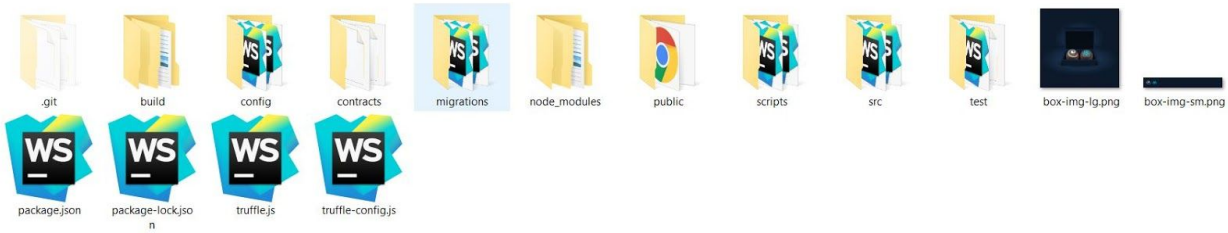
```
mkdir creditcoin
```

```
cd creditcoin
```

- **Clone for a backup. Here below are the steps to follow from scratch. Copy App.js and ipfs.js only when needed. The link is given below.**
- Comments have been used to describe the process and code.

```
npm install -g truffle
truffle unbox react
```

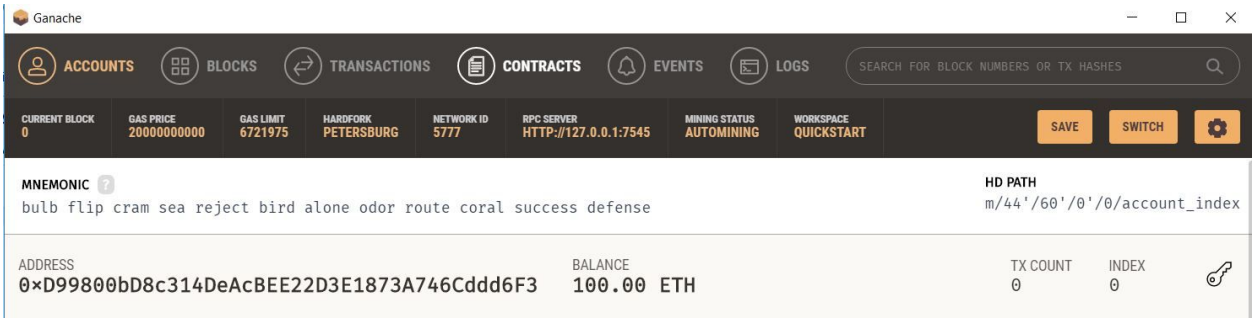
- You need to make sure that all the client files are outside the folder because package.json and node_modules have to be in the same directory. The change in folders might not work execute the modules.



- Node_modules will be created automatically. It contains all the packages. It gets installed locally as the module used in the code will require it's support to run the functionalities.
- We created this DApp in Windows so we should edit the `truffle-config.js` as `truffle.js` is for the users having MacOs or Linux system.
- The file - truffle-config.js contains the credentials and the network id for connecting to the Ethereum Blockchain running on a private node in Ganache.

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1", //Should match Ganache's host
      port: 7545, //should match Ganache's port number
      network_id: "*" // Match any network id
    }
  }
};
```

- It should match exactly to the host's address and the port number in Ganache.



- Our acc address is above **having 100.00 ETH.**
- The network id is **5777**
- RPC server: **HTTP://127.0.0.1:7545**
- **Now we need to check the truffle version and solidity's version for writing a smart contract for file sharing.**

```
truffle version
```

- **This will bring up a version which should be the minimum version for our smart contract.**

truffle-config.js SimpleStorage.sol

```
pragma solidity ^0.5.0; //The version can be found by typing "truffle version", we need to
//put caret to match all the supporting versions of the solidity

contract SimpleStorage { //generic smart contract by default
  uint ipfsHash;

  function set(uint x) public {
    ipfsHash = x; //used to set the ipfsHash in an unsigned datatype x
  }

  function get() public view returns (uint) {
    return ipfsHash; //used to retrive the ipfsHash in the form of an unsigned datatype
  }
}
```

- The solidity version is 0.5.0, which we set by typing the truffle version.
- We put **^ Caret** to match all the supporting versions of the solidity.
- **uint** - An unsigned integer has been used.
- To mine the contract and since we're doing for the first time and from the beginning we need to migrate using truffle. Make sure Ganache is running before typing in this command.

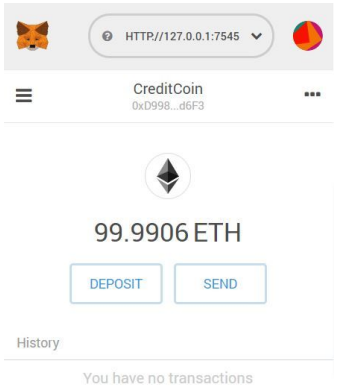
```
truffle migrate --reset
```

- A transaction will take place and in case you want to check, open Ganache and the 100 ETH should reduce as per the transaction cost.
- Now we need to install the ipfs-api (`now changed to ipfs-http-client`) needed for development. We used Infura to connect IPFS node instance.
- Create a new file named - `ipfs.js`
- ipfs.js - Here we will create an instance for the **ipfs-api/ipfs-http-client calls**
- **And then we import instance if ipfs into the App.js.**
- A template layout of the webpage was already ready in the truffle box. So, we just need to code it according to our functionality.

Github Project Link - Use App.js and ipfs.js from src

```
https://github.com/SmitKabrawala/IPFS_EthereumDApp.git
```

- **So, this is how it works**.
    - We create a webpage.
    - Then we create an event onSubmit() which passes a message in the console that the file has been submitted successfully.
    - This is handled by a function captureFile(Event), where a const file is created which is basically a form of data that will remain constant for the entire operation.
    - **We're taking the file reader and passing it to convert into an array that buffer is going to understand. We're going to pass it into a buffer that will allow us to a file that can be sent into IPFS.**
    - Get a Google extension of **MetaMask.**
    - **Open Ganache and copy the key to import the account.**



    - After refreshing the web-page, if the picture returns again without the need for uploading then it shows that the picture is stored and this can be checked in the terminal window by typing in this -

```
truffle(development)> SimpleStorage.deployed().then(function(i) { app = i;})
undefined
truffle(development)> app.get().then(function(value) { ipfsHash = value });
undefined
truffle(development)> ipfsHash
'QmZyZjiuW6TfNWX9a4qAxmw7V1rVMZKCnMP45KZnHoMsUq'
truffle(development)> .exit
```

And hence the picture was stored on the IPFS using Ethereum Blockchain