Decentralised Storage Exercises

Introduction

This module will teach you the fundamentals of what decentralised storage is and how it can be used within an application.

The practical is divided in three parts. Part 1 will introduce simple code snippets that use the IPFS library to store and retrieve data.

Part 2 will expand on this, integrating the code into a simple application with a front-end.

Part 3 will add functionality to the front-end, in order to store images on IPFS such as can be used for NFT's.

Gitpod version of code found here (http://gitpod.io/#https://github.com/ExtropyIO/Academy)

Instructions - Part 1

1. Navigate to the material:
   `$ cd materials/Week\ 3\ -\ web3_technologies/decentralised\ storage/app1`
2. Initialise a Node.JS project, accepting the defaults with the **-y** flag:
   `$ npm init -y`
3. Install the required dependencies for the application using the node package manager (npm). This can take some time to complete:
   `$ npm i ipfs it-all`
4. Create two new files called **ipfs_add.js** and **ipfs_get.js**
   `$ touch ipfs_add.js && touch ipfs_get.js`
5. Copy these contents inside of **ipfs_add.js**, replacing the **data** string with something unique:

```
const IPFS = require('ipfs');

(async () => {
  const node = await IPFS.create();

  const data = 'Hello, <YOUR NAME HERE>';

  const cid = await node.add(data);

  console.log(cid.path);
})();
```

This standalone function will run when executing the file. It:

- Imports the **ipfs** dependency that was installed prior.
- Creates a local IPFS node.
- Stores a string into a variable called **data**. This data could be anything, including pictures,

video, music etc.
- Adds this data to the IPFS network, saving the corresponding CID hash into a variable **cid**.
- Prints out the CID hash onto the console.

6. Run the file:
   `node ipfs_add.js`
   The CID of the data should now be displayed in the terminal. If you change the input data in any way this CID will change - why not have a go.

7. Copy these contents inside of **ipfs_get.js**, replacing the **cid** string with the CID from the terminal:

```
const IPFS = require('ipfs');
const all = require('it-all');

(async () => {
  const node = await IPFS.create();

  const cid = 'QmPChd2hVbrJ6bfo3WBcTW4iZnpHm8TEzWkLHmLpXhF68A';

  const data = Buffer.concat(await all(node.cat(cid)));

  console.log(data.toString());
})();
```

This standalone function will also run when executing the file. It:

- Imports the required dependencies.
- Creates a local IPFS node.
- Stores the string into a variable **cid**.
- Retrieves the data with the content address stored in **cid** from the IPFS network. This also uses the **it-all** library to concatenate the data stream and stores the returned value into the variable **data**.
- The data are printed to the console.

8. Run the file:
   `node ipfs_get.js`
   Whatever the data was in the **ipfs_add.js** file should now be visible on the console. Change the **cid** and if the data exists on the network it will be retrieved.

Instructions - Part 2

This section brings the previous functions into an interactive frontend.

1. Navigate to the material:
   `$ cd materials/Week\ 3\ -\ web3_technologies/decentralised\ storage/app2`

2. Install the dependencies:
   `$ npm install`
   The dependencies used:
   Express (https://www.npmjs.com/package/express) is a web application framework used to create the routes to pass information from the front-end to the back-end.
   IPFS (https://www.npmjs.com/package/ipfs) is the peer-to-peer protocol allowing data to be

stored and retrieved online.
it-all (https://www.npmjs.com/package/it-all) collects data from an async stream and returns as an array.

3. Start the application:
   `$ node src/app.js`
   This will run the application that can be viewed in the web-browser by navigating to **localhost:3000**
   This runs similar code to the snippets used in the previous example with a wrapper allowing interaction via a simple web front-end.

Error on running:
```
$ npm cache clean --force
$ npm update
```

- Allow popups in your browser to view front-end.

Instructions - Part 3

This section introduces more functionality to the application allowing file upload and retrieval.

1. Navigate to the material:
   `$ cd materials/Week\ 3\ -\ web3_technologies/decentralised\ storage/app3`

2. Install the dependencies:
   `$ npm install`

3. Start the application:
   `$ node src/app.js`

Application Notes

- Additional routes for posting and retrieving image files.
- Client-side file retrieval since backend doesn't have privileges to read data from the host.
- Various conversions between filetypes to allow sending data through routes.
- Increased maximum data allowance in app to 50mb.