

Exercises: Create Simple Casino with Solidity, Truffle, MetaMask, Provable and IPFS

Context

Provable Things (<https://provable.xyz/>) is the leading **oracle** service for smart contracts and blockchain applications, serving thousands of requests every day on **Ethereum** and **Bitcoin** (Rootstock). In the **blockchain** space, an **oracle** is a party which provides **real world data**. The need for such a figure arises from the fact that blockchain applications such as Bitcoin scripts and smart contracts cannot access and directly fetch the data they require: price feeds for assets and financial applications; weather-related information for peer-to-peer insurance; random number generation for gambling, etc.

Prerequisites

You must have the following software installed, along with corresponding versions:

- **NodeJS** v13.5.0
 - Check: `node -v`
- **NPM** (includes NPX) v6.13.4
 - Check: `npm -v` or `npx -v`
- Able to access <https://remix.ethereum.org/>
- **Truffle** v5.1.17

Note: If the screenshots in this document seem small/blurry. Zooming in will help improve clarity.

Goal

We are going to create a **decentralized casino application** where users are able to bet money for a number between **1 and 10** and if they are correct, they win a portion of all the ether money staked after 100 total bets.

Setup the Project

1. Clone/Download the project and code templates:

```
git clone https://github.com/kingsland-innovation-center/decentralized-casino.git
```

Or you may go here and manually download the project:

```
https://github.com/kingsland-innovation-center/decentralized-casino
```

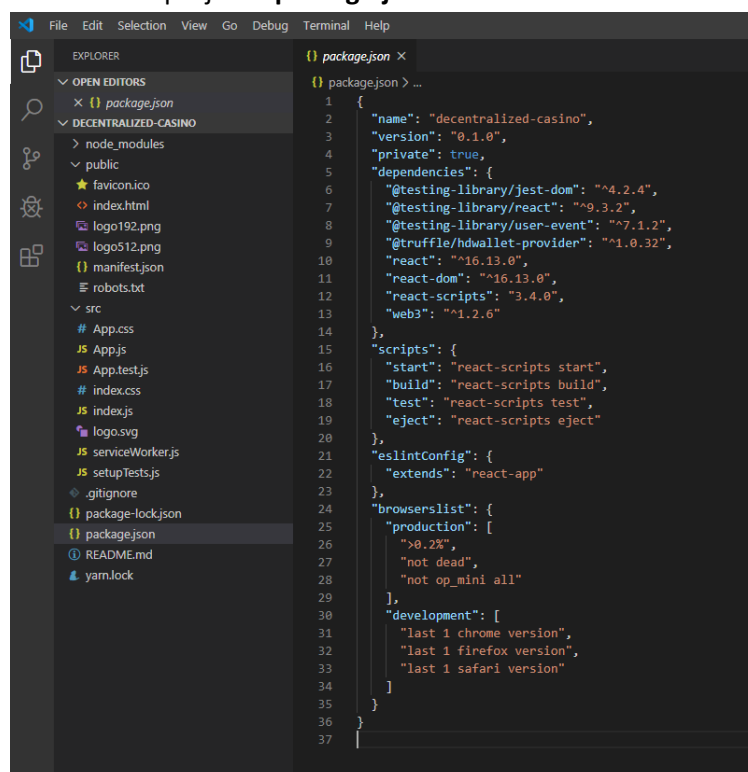
Go to your project folder.

From now on, this will be your workspace. Make sure that the files created and commands executed are being done in this directory:

```
cd decentralized-casino
```

2. Install the dependencies.

Dependency files can be found in the project's **package.json** file.



```
{
  "name": "decentralized-casino",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.3.2",
    "@testing-library/user-event": "^7.1.2",
    "@truffle/hdwallet-provider": "^1.0.32",
    "react": "^16.13.0",
    "react-dom": "^16.13.0",
    "react-scripts": "3.4.0",
    "web3": "^1.2.6"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": "react-app"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

To install these project dependencies, run:

```
npm install
```

Install **Truffle** and its dependencies globally (don't forget to run your command shell as **administrator**):

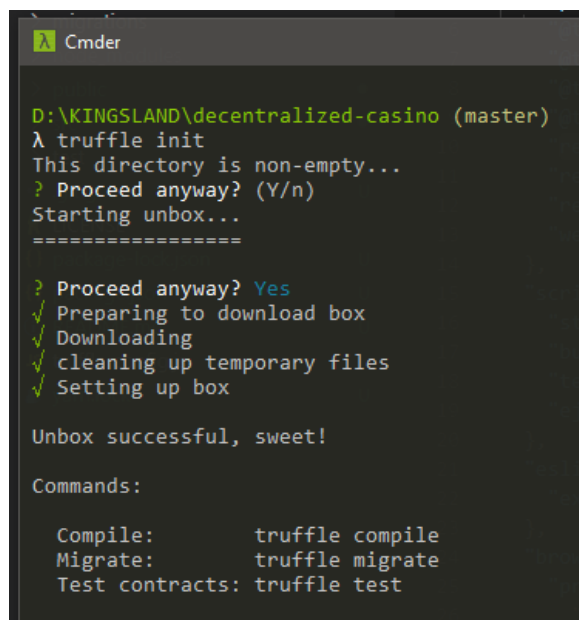
If you are using windows, you may want to install this dependency first if there are errors installing truffle. **This will take a while, go grab a snack, perhaps coffee:**

```
npm install --global --production windows-build-tools
```

```
npm install --global truffle@5.1.17
```

3. If you are building your own project from scratch, use this command to initialize a **truffle** project with recommended directory structures. **Since you have a preset project, there is no need to do this step as this has already been done for you:**

```
truffle init
```

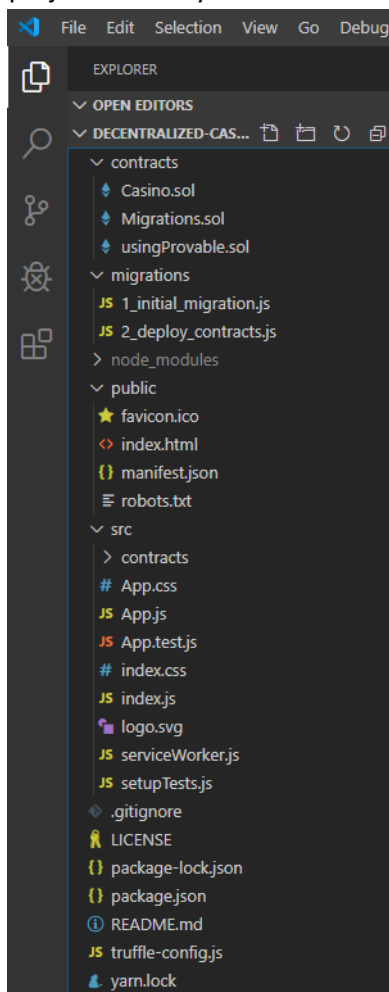


```
Cmdr
D:\KINGSLAND\decentralized-casino (master)
λ truffle init
This directory is non-empty...
? Proceed anyway? (Y/n)
Starting unbox...
=====
? Proceed anyway? Yes
√ Preparing to download box
√ Downloading
√ cleaning up temporary files
√ Setting up box

Unbox successful, sweet!

Commands:
Compile:      truffle compile
Migrate:      truffle migrate
Test contracts: truffle test
```

4. You will now have this structure in your project directory:

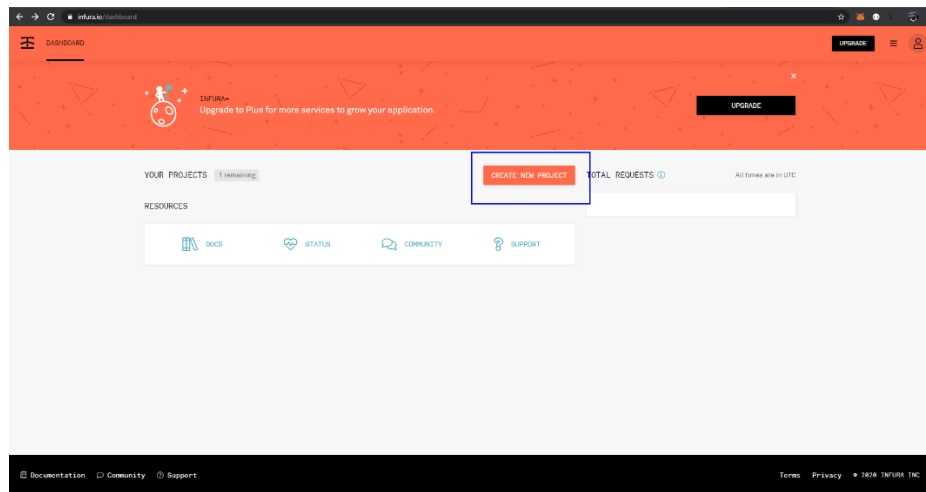


5. Get your Project ID on Infura.

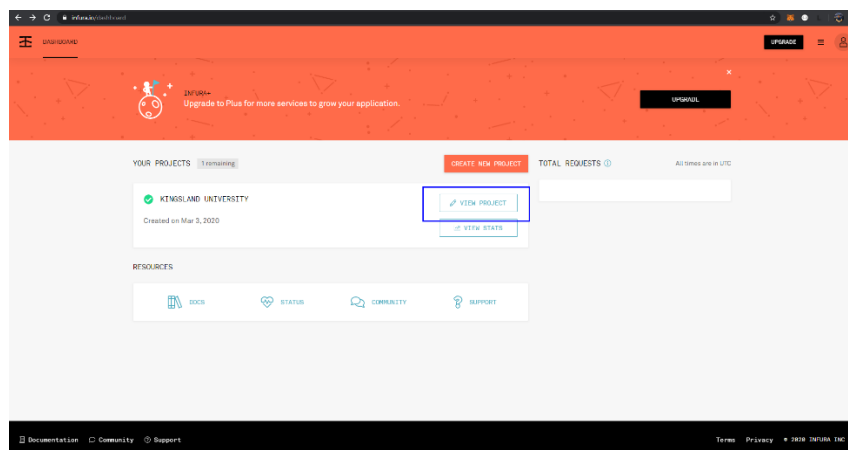
If you are new, register an account: <https://infura.io/register>

If you are an existing user, login: <https://infura.io/login>

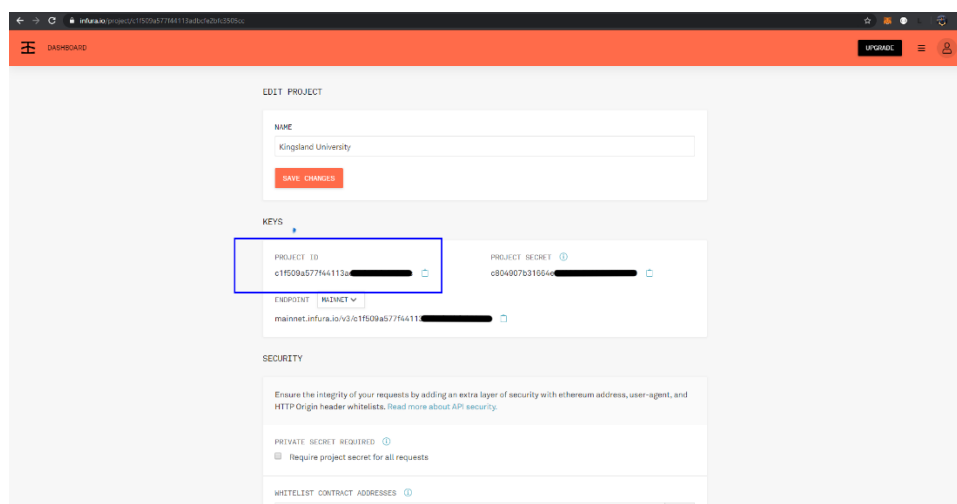
After logging in, you may use an existing key or create one.



View your project:



Take note of your Infura Project ID, you will use this later:

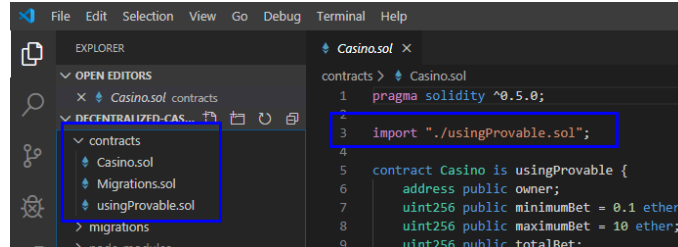


Once you're on the page, click **GENERATE**:

Problem 1. Create Smart Contracts

1. We will use the **ProvableAPI**. Copy the file linked below to the **contracts/** folder with **usingProvable.sol** as the file name if you're starting from scratch. Otherwise, you may already have it in your project.

https://github.com/provable-things/ethereum-api/blob/master/provableAPI_0.5.sol



2. Create the file **contracts/Casino.sol**, this is the main Solidity contract that we will be writing.

The Casino contract will have:

- **owner** – Address of the owner.
- **minimumBet** (default **0.1 ether**) – The minimum bet a user has to make.
- **maximumBet** (default **10 ether**) – The maximum bet that can be made for each game.
- **numberOfBets** – Number of bets that the users have made.
- **maxNumberOfBets** (default **100**) – The maximum number of bets at a given time to avoid excessive gas consumption.
- **winningNumber** – The lucky number that decides the winner.
- **An array of all the players.**
 - `address[] public players;`
- **A structure for storing the players and their bets.**
 - `mapping(address => uint256) public playerBets;`
- **A structure for storing each number and which players have bet on that particular number.**
 - `mapping(uint256 => address payable[]) public bets;`
- **Events** that log some actions of the contract.

```
contracts > Casino.sol
1  pragma solidity ^0.5.0;
2
3  import './usingProvable.sol';
4
5  contract Casino is usingProvable {
6      address public owner;
7      uint256 public minimumBet = 0.1 ether;
8      uint256 public maximumBet = 10 ether;
9      uint256 public numberOfBets;
10     uint256 public maxNumberOfBets = 100;
11     uint256 public winningNumber;
12
13     address[] public players;
14     mapping(address => uint256) public playerBets;
15     mapping(uint256 => address payable[]) public bets;
16
17     event generatedRandomNumber(string randomNumber);
18     event LogNewProvableQuery(string description);
19 }
```

3. Now create the constructor that is used to configure:

- The minimum bet that each user has to make in order to participate in the game.
- The maximum number of bets that are required for each game.

You **may** also set the type of authenticity proof of **provable** which are simply cryptographic guarantees proving the authenticity of the data (read more [here](#)):

```
19
20 ✓ constructor(uint256 _minimumBet, uint256 _maxNumberOfBets) public {
21     owner = msg.sender;
22
23 ✓     if (_minimumBet > 0) {
24         minimumBet = _minimumBet;
25     }
26
27 ✓     if (_maxNumberOfBets > 0) {
28         maxNumberOfBets = _maxNumberOfBets;
29     }
30
31     provable_setProof(proofType_Ledger);
32 }
33
```

4. Implement the **bet** function:

```
33
34 function bet(uint256 numberToBet) public payable {
35     require(numberOfBets < maxNumberOfBets, "Bet table is full.");
36     require(
37         numberToBet >= 1 && numberToBet <= 10,
38         "Choose a bet number between 1 and 10."
39     );
40     require(
41         msg.value >= minimumBet,
42         "Bet is lesser than specified minimum bet."
43     );
44     require(
45         playerBets[msg.sender] == 0,
46         "You are not allowed to change your bet."
47     );
48
49     playerBets[msg.sender] = numberToBet;
50     bets[numberToBet].push(msg.sender);
51     players.push(msg.sender);
52
53     numberOfBets += 1;
54
55     if (numberOfBets >= maxNumberOfBets) {
56         generateWinningNumber();
57     }
58 }
59
```

5. We should generate winner number by using provable function **provable_newRandomDSQuery** which takes **delay**, **numberRandomBytes** and **callbackGas**:

```
59
60 function generateWinningNumber() public payable {
61     uint256 delay = 0;
62     uint256 numberRandomBytes = 7;
63     uint256 callbackGasLimit = 400000;
64
65     provable_newRandomDSQuery(delay, numberRandomBytes, callbackGasLimit);
66     emit LogNewProvableQuery(
67         "Provable query was sent, standing by for the answer..."
68     );
69 }
70
```

6. We should create a **callback** function which gets called by Provable when a random number is generated.
- **_queryID** – A unique ID that identifies a specific query done to Provable and it is returned to the contract as a parameter of the callback transaction.
 - **_result** – A string that contains the generated random number from Provable.
 - **_proof** – A signature which proves that the response indeed came from Provable.

These can then be verified by calling **provable_randomDS_proofVerify__returnCode()** function.

When all goes well, we process the random number to get it within our “bounds” which is from 1 to 10. Then, we distribute the prizes.

```
70
71 function __callback(
72     bytes32 _queryID,
73     string memory _result,
74     bytes memory _proof
75 ) public {
76     require(msg.sender == provable_cbAddress(), "Bad callback");
77     require(
78         provable_randomDS_proofVerify__returnCode(
79             _queryID,
80             _result,
81             _proof
82         ) ==
83         0,
84         "Bad proof"
85     );
86     emit generatedRandomNumber(_result);
87
88     bytes32 encodedRandom = keccak256(abi.encodePacked(_result));
89     winningNumber = (uint256(encodedRandom) % 10) + 1;
90     distributePrizes();
91 }
92
```


7. Implement the function to send the corresponding Ether to each winner then reset the bets by deleting all the players for the next game and resetting the **total bet** and **number of bets**:

Make sure to handle the case when there is no winner: **bets[winningNumber].length != 0**

```
92
93     function distributePrizes() public {
94         if (bets[winningNumber].length != 0) {
95             uint256 prize = address(this).balance / bets[winningNumber].length;
96
97             for (
98                 uint256 index = 0;
99                 index < bets[winningNumber].length;
100                 index++
101             ) {
102                 address payable winner = bets[winningNumber][index];
103                 winner.transfer(prize);
104             }
105         }
106
107         resetBets();
108     }
109
110     function resetBets() private {
111         for (uint256 index = 1; index <= 10; index++) {
112             bets[index].length = 0;
113         }
114
115         for (uint256 index = 0; index < players.length; index++) {
116             playerBets[players[index]] = 0;
117         }
118         numberOfBets = 0;
119     }
120 }
```

8. Finally, implement a **getContractBalance()** view function so that the frontend can see the total contract balance.

```
120
121     function getContractBalance() public view returns(uint256 balance) {
122         return address(this).balance;
123     }
124 }
125
```

9. We are now ready to deploy the contracts to the Ropsten Test Network.

Go to the **migrations/** folder and create the file **2_deploy_contracts.js** and write the code below:

- First, we require the **Casino.sol** contract.
- Then, in the **.deploy()** method we specify the minimum bet, in this case it's **0.1 ether** converted to wei with that function
- Constructor arguments
 - **0.1** is the minimum bet. Use the *web3.utils* library to convert the unit into *wei*.
 - **2** is the maximum number of bets (for testing purposes).
- Finally, the gas limit that we are willing to use to deploy the contract. Let's do **5,000,000**.

```
JS 2_deploy_contracts.js X
migrations > JS 2_deploy_contracts.js > ...
1  /* global artifacts, web3 */
2
3  const Casino = artifacts.require("Casino");
4
5  module.exports = function(deployer) {
6    const minimumBet = "0.1";
7    const maxNumberOfBets = "2";
8    const minimumBetEthers = web3.utils.toWei(minimumBet, "ether");
9
10   deployer.deploy(Casino, minimumBetEthers, maxNumberOfBets, {
11     gas: 5000000
12   });
13 };
14
```

10. Open **truffle-config.js** from the root folder and customize your Truffle configuration

- Use your own Infura Project ID key from the Infura settings in an [earlier step](#)
- Use your own mnemonic you had generated in an [earlier step](#)

```
EXPLORER JS truffle-config.js X
OPEN EDITORS
X JS truffle-config.js M
DECENTRALIZED-CASINO
contracts
  Casino.sol
  Migrations.sol
  usegethtruffle.sol
migrations
  1_initial_migration.js
  2_deploy_contracts.js
node_modules
public
src
.gitignore
LICENSE
package-lock.json
package.json
README.md
JS truffle-config.js M
yam.lock

1  /**
2   * More information about configuration can be found at:
3   * https://truffleframework.com/docs/advanced/configuration
4   *
5   */
6
7  const HDWalletProvider = require("@truffle/hdwallet-provider");
8
9  const INFURA_KEY = "c1f509a577f4411"; // REPLACE WITH YOUR OWN KEY
10 const MNEMONIC = "need smooth gentle season fatigue"; // REPLACE WITH YOUR OWN MNEMONIC
11
12 module.exports = {
13   /**
14    * The default output directory for compiled contracts is ./build/contracts relative to the project root.
15    * This can be changed with the contracts_build_directory key.
16    */
17   contracts_build_directory: "./src/contracts",
18
19   /**
20    * Networks define how you connect to your ethereum client and let you set the
21    * defaults web3 uses to send transactions.
22    * $ truffle migrate --network ropsten
23    */
24   networks: {
25     ropsten: {
26       provider: () =>
27         new HDWalletProvider(
28           MNEMONIC,
29           `https://ropsten.infura.io/v3/${INFURA_KEY}`
30         ),
31       network_id: 3, // Ropsten's id
32       gas: 5000000, // Ropsten has a lower block limit than mainnet
33       confirmations: 0, // # of confs to wait between deployments. (default: 0)
34       timeoutBlocks: 200, // # of blocks before a deployment times out. (minimum/default: 50)
35       skipDryRun: true // Skip dry run before migrations? (default: false for public nets )
36     },
37   },
38
39   // Configure your compilers
40   compilers: {
41     solc: {
42       version: "0.5.0" // Fetch exact version from solc-bin (default: truffle's version)
43     }
44   }
45 };
46
```

Problem 2. Deploy the application online with IPFS

1. Compile your contract:

```
truffle compile
```

2. Deploy the contract on the Ropsten test network.

```
truffle migrate --network ropsten
```

Take note of the deployed Casino contract address:

```
1_initial_migration.js
-----
> Replacing 'Migrations'
> transaction hash: 0xd1715220d6d1c9a51398afae44d6da3e8973e1fe0ce1122d1ad9385dbbde0
> blocks: 2
> contract address: 0x5af0404eA6368e9D298287583890cFe6922A5735
> block number: 7446618
> block timestamp: 153252394
> account: 0xf32ff0849b31fa359c3b508972a2890c763187b2
> balance: 1.06608086
> gas used: 175005
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.0035019 ETH

> Saving migration to chain.
> Saving artifacts
> Total cost: 0.0035019 ETH

2_deploy_contracts.js
-----
> Replacing 'Casino'
> transaction hash: 0x73801d02f805fdec5d627185c61138f1ba14ddee618d902acc8df4a583862
> blocks: 1
> contract address: 0xAcE5f17881651B9e1206eaE01c49d7E5A7c761A6
> block number: 7446619
> block timestamp: 1583253428
> account: 0xf32ff0849b31fa359c3b508972a2890c763187b2
> balance: 1.0001214
> gas used: 2804029
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.05608058 ETH

> Saving migration to chain.
> Saving artifacts
> Total cost: 0.05608058 ETH

Summary
-----
> Total deployments: 2
> Final cost: 0.05958248 ETH

0: KINGSLAND\decentralized-casino (master)
^|
```

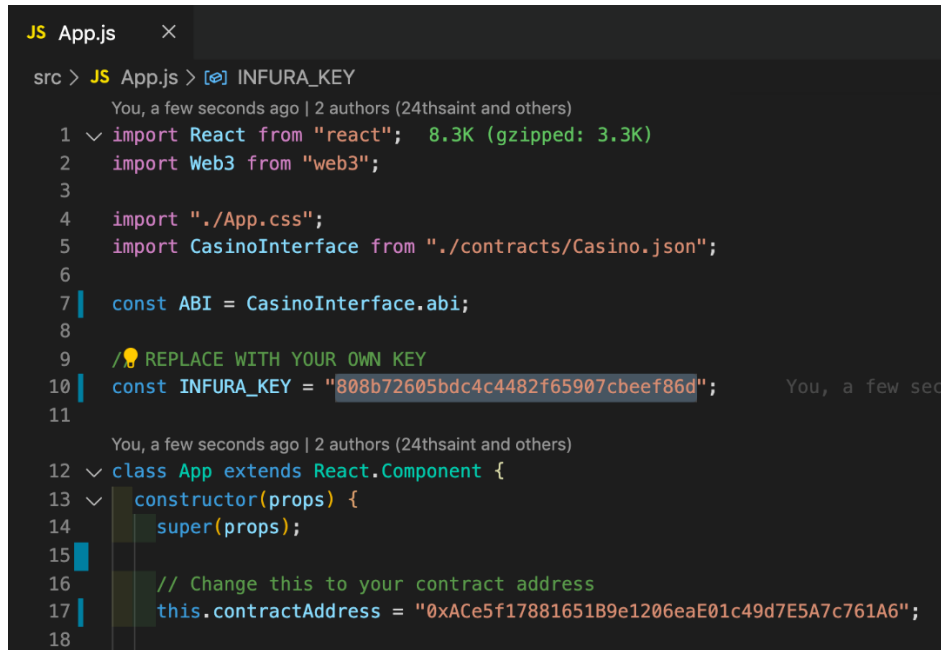
3. Then, in `src/app.js`, change the address of the contract instance to the address of the contract you deployed on Ropsten:

```
// Change this to your contract address
this.contractAddress = "0xAcE5f17881651B9e1206eaE01c49d7E5A7c761A6"
```

```
src > JS App.js > App > constructor
1 import React from "react";
2 import Web3 from "web3";
3
4 import "../App.css";
5 import CasinoInterface from "../contracts/Casino.json";
6
7 const ABI = CasinoInterface.abi
8 const INFURA_KEY = "c1f509a577f44113adbcfe2bfc3505cc";
9
10 class App extends React.Component {
11   constructor(props) {
12     super(props);
13
14     // Change this to your contract address
15     this.contractAddress = "0xAcE5f17881651B9e1206eaE01c49d7E5A7c761A6"
16
17     this.validBets = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
18     this.state = {
19       winningNumber: 0,
20       numberOfBets: 0,
21       minimumBet: 0,
22       totalBet: 0,
23       maxNumberOfBets: 0,
24       currentBet: 0
25     };
26   }
27 }
```

4. Also, in `src/app.js`, change the `INFURA_KEY` to the Project ID Key from the Infura settings in an [earlier step](#):

```
// REPLACE WITH YOUR OWN KEY
const INFURA_KEY = "808b72605bdc4c4482f65907cbeef86d";
```

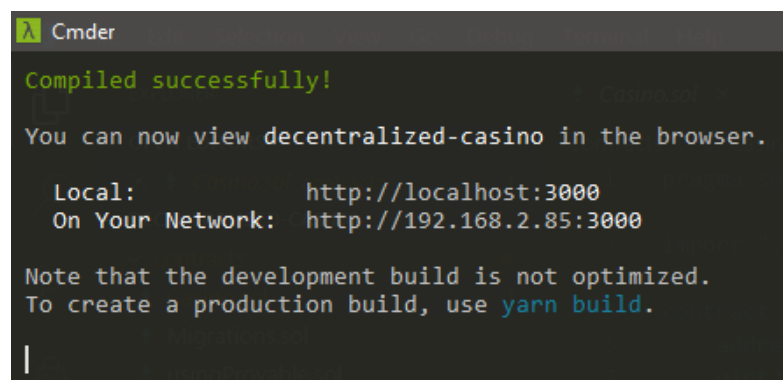


```
JS App.js ×
src > JS App.js > [e] INFURA_KEY
You, a few seconds ago | 2 authors (24thsaint and others)
1  import React from "react";  8.3K (gzipped: 3.3K)
2  import Web3 from "web3";
3
4  import "./App.css";
5  import CasinoInterface from "./contracts/Casino.json";
6
7  const ABI = CasinoInterface.abi;
8
9  // REPLACE WITH YOUR OWN KEY
10 const INFURA_KEY = "808b72605bdc4c4482f65907cbeef86d";
11
12 You, a few seconds ago | 2 authors (24thsaint and others)
13 class App extends React.Component {
14   constructor(props) {
15     super(props);
16
17     // Change this to your contract address
18     this.contractAddress = "0xAce5f17881651B9e1206eaE01c49d7E5A7c761A6";
```

5. Sanity Checking.

Make sure that your dApp will run correctly. This command will create a local server for your files and automatically launch your browser at <http://localhost:3000>

```
npm start
```



```
Cmder
Compiled successfully!
You can now view decentralized-casino in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.2.85:3000

Note that the development build is not optimized.
To create a production build, use yarn build.
```

← → ↻ ⓘ localhost:3000

Bet for your best number and win huge amounts of Ether

Number of bets: 0

Last winning number: No draws yet

Total ether bet: 0 ether

Minimum bet: 0.1 ether

Max number of bets: 2

Vote for the next number

How much Ether do you want to bet? ether

1

2

3

4

5

6

7

8

9

10


Only working with the Ropsten Test Network

You can only vote once per account

Your vote will be reflected when the next block is mined

At this point, you can interact with your dApp.

Try to place some bets, change your account in Metamask, and place another bet.
Keep doing until you reach the maximum number of bets and your smart contract generates a random number from Provable.



KINGSLAND

SCHOOL OF BLOCKCHAIN

<https://kingslanduniversity.com>

Page 13 of 18

6. You are now ready to deploy a **decentralized application on IPFS!**

Compile your ReactJS project.

```
npm run build
```

This will create a directory named **build/**

This contains all your source files in **src/** bundled together with optimizations.

Take note of the location of this directory.

```
D:\KINGSLAND\decentralized-casino (master)
λ npm run build

> decentralized-casino@0.1.0 build D:\KINGSLAND\decentralized-casino
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

 299.73 KB  build\static\js\2.efc03dab.chunk.js
 27.65 KB  build\static\js\main.c1339dd4.chunk.js
 783 B     build\static\js\runtime-main.939ad078.js
 573 B     build\static\css\main.19386453.chunk.css

The project was built assuming it is hosted at /.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
You may serve it with a static server:

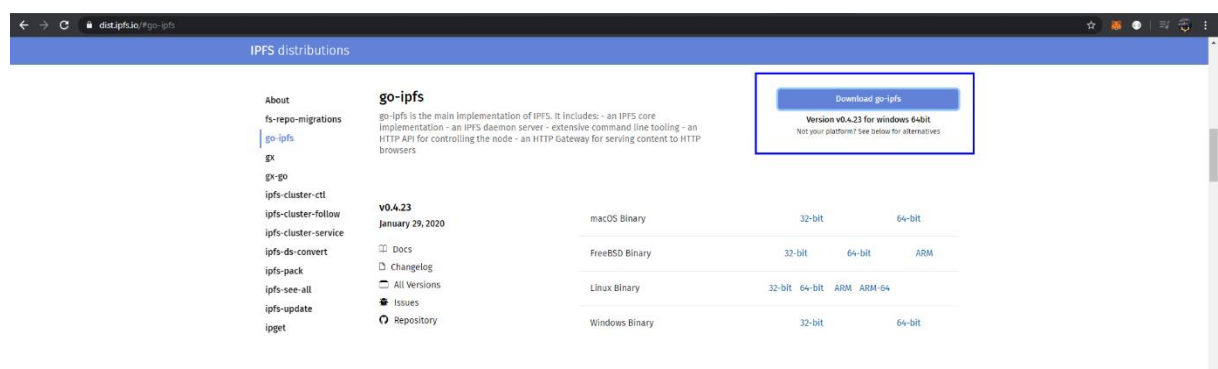
  yarn global add serve
  serve -s build

Find out more about deployment here:

  bit.ly/CRA-deploy

D:\KINGSLAND\decentralized-casino (master)
λ cd build\
D:\KINGSLAND\decentralized-casino\build (master)
λ
```

7. Go to <https://dist.ipfs.io/#go-ipfs> and download **go-ipfs** then extract it.



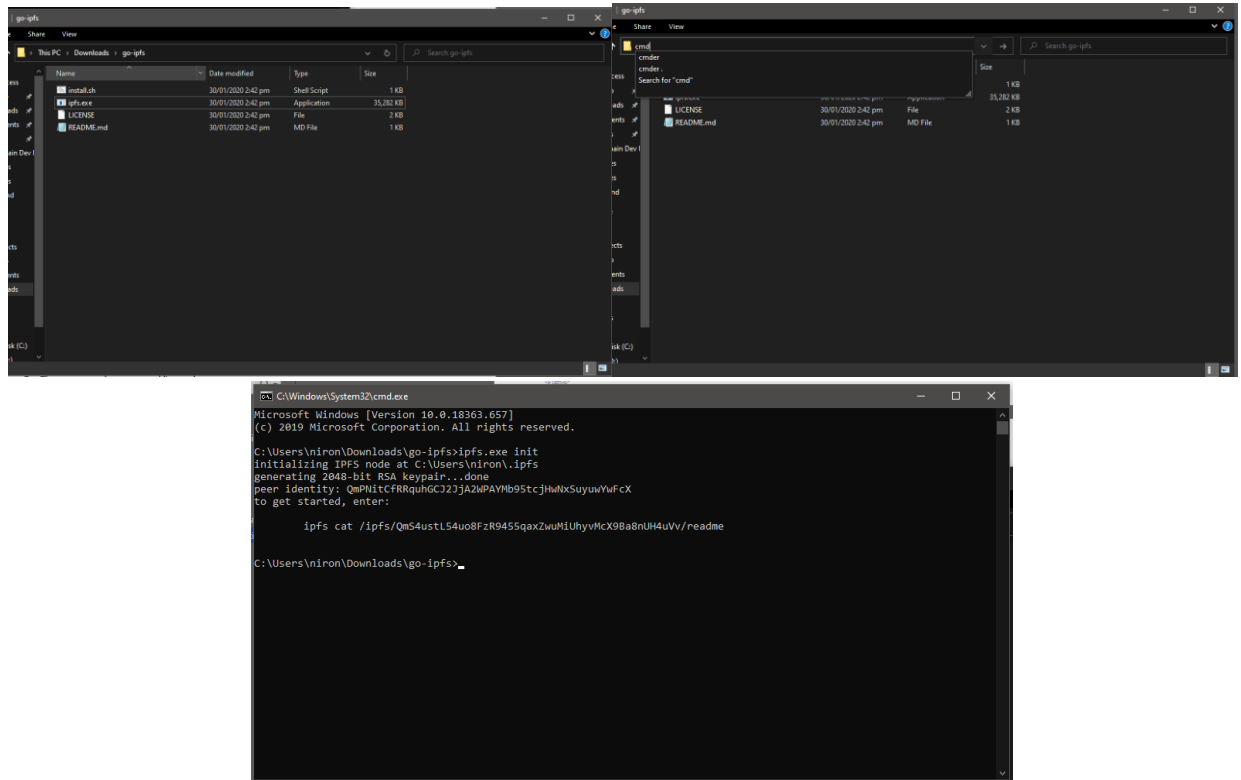
8. Go to the *IPFS* folder and run in Command Prompt:

```
ipfs.exe init
```

If you are using Mac OS, run the install script first, then initialize:

```
./install.sh  
ipfs init
```

(Advanced users: You may add IPFS as a *path variable* to easily access the command anywhere.)

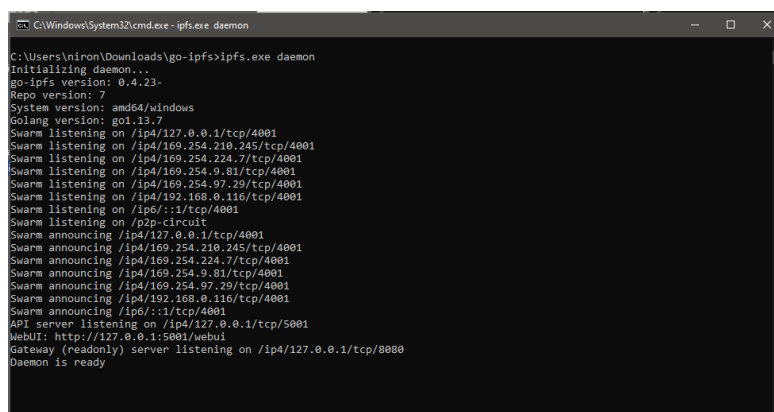


9. Open command line and type **ipfs daemon**. This will make your machine node and IPFS node.

```
ipfs.exe daemon
```

If you are using Mac OS, type:

```
ipfs daemon
```



Keep this terminal running, do not terminate the process.

10. Open another *separate* command line and type:

```
ipfs swarm peers
```

This will get you the peers that your machine has established a connection with. These peers are ready to share your published content.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Niron\Downloads>go-ipfs>ipfs.exe swarm peers
/ip4/104.131.131.82/tcp/4001/ipfs/QmaCPdHwGvV28GHeYERUEnRQAwE3N8SzbUtFsmvsgQLUuvJ
/ip4/104.236.76.40/tcp/4001/ipfs/QmSCLPppRtQSG6WkdZT2H73ULpJvFq3Z6ha4oFGLKngM
/ip4/104.236.76.40/tcp/4001/ipfs/QmSolV48bm51jHqC4gDYZ09cy3U6AXX3dAbzguzf2aDs64
/ip4/111.229.156.247/tcp/4001/ipfs/QmToE4HtLFJ1abgVvYtpMmuSHZ13zVm6T4AM1TuxydvJ
/ip4/119.27.164.219/tcp/4001/ipfs/QmSkEaUgva3EWMnzSTrqvGxZKQMsHurbVQ8U5Y9D9uSiYq
/ip4/128.73.13.130/tcp/46649/ipfs/QmXq3ad7vTChv4pab17bdJuitQv4wXyH51d8dCL4RV9eQ
/ip4/139.178.69.39/tcp/4001/ipfs/QmZMxHdpMkew1VZLMRxaNkUeZpDUB34pWjZ1kZvsd16Z1c
/ip4/139.59.152.171/tcp/4001/ipfs/QmU6hw2VU3CceUXtS7TxxmxE39u4htuHdg4gUvxJjbmIX
/ip4/147.75.107.135/tcp/4001/ipfs/QmLkStLCmpSaptzGUjJSS36vRueK5Jw43JpDqKQ9vQa
/ip4/147.75.195.153/tcp/4001/ipfs/Qm9m57a18DhAKXj9nF5eN7ZqncFfZ54b1psTChbuei
/ip4/147.75.33.135/tcp/4001/ipfs/Qmbut9Yuz9YEDr28Y8Sg5y2k41Uvm2Q3PhwDz3yGfSd6
/ip4/147.75.69.119/tcp/4001/ipfs/QmYUHU1M8t86gRRvUmE425mybPKXqTPEUPDuaCvTLm56Bw
/ip4/147.75.77.187/tcp/4001/ipfs/QmQU2ECmQaQQR219bChdGNJchTbq5TbXJ16u19uLTa
/ip4/147.75.80.110/tcp/4001/ipfs/QmbFgm5zan8P6eWmWeyfncR5fEYEPbht5b1FW1C37aQ7y
/ip4/147.75.83.83/tcp/4001/ipfs/QmbLHAnMoJPMSCRSZhtx6BhJX9K1KMN6tpybucqanJ75Nb
/ip4/147.75.85.167/tcp/4001/ipfs/QmXdxYCUZTgPSLNR6mrfyhm3k543n2y5E8eCQZchRW
/ip4/147.75.96.47/tcp/4001/ipfs/QmVjLYXj8NtBUT1XoEHLmaw9NPZ7JRUSjppi18a3uw29CZ
/ip4/159.283.166.172/tcp/4001/ipfs/QmUj1yjb7Z1GtPSuEhJNUJ2N7m3y9Yf6hczxsPRVog9mPR
/ip4/165.22.169.12/tcp/4001/ipfs/QmQyWkPBRVshokFbVRzU7R6VHY7MDH74Wk9NtK4iJjVtVI
/ip4/167.86.84.202/tcp/4001/ipfs/QmbcK9P5zWpym6MCSwQeMRd5ZvSua1GjzWh4JNjU6RzS
/ip4/18.185.241.99/tcp/20044/ipfs/12D3KoonAQmSDnProBrY89nuZhdDpeGwFvVnM1k3tCP4AMovxwEM
/ip4/203.195.222.205/tcp/4001/ipfs/QmY8maYipKBPHXAGH6gjoEC4hblLntMwLJT3qVFLaKx
/ip4/211.159.188.188/tcp/4001/ipfs/QmFjZKjmmWmerocVes1SS4F20HwUURQ1eferPw
/ip4/213.166.71.109/tcp/4001/ipfs/Qmb5x81U7nAAk3CDuUNJcm6Vx7DCVfMRsGL6lbb6xsb5
/ip4/35.233.189.126/tcp/4001/ipfs/QmT1XNcqsSV9inzX2UbV74mc8qov1yXqTewJaeMdlMVS
/ip4/54.37.156.33/tcp/4001/ipfs/QmTJ2kCapz6d4yRP3vXN1Mu63nnkAGH8cNv55VoxwCdh
/ip4/66.166.227.90/tcp/4001/ipfs/QmV7LgiywdVnp2odFydhG6cpj01z8UbtqXB1r1r6G8bVY
/ip4/66.42.111.225/tcp/20088/ipfs/12D3KoonDz11maQmJCN5d8AQVFTJ1vnm3WuFAM661eg7gRqZtdP
/ip4/66.42.111.225/tcp/20075/ipfs/12D3KoonQmTUXEjEZD9s36a1Z40pprG07zBCBHSdEADNMtCRf32
/ip4/70.74.171.142/tcp/4001/ipfs/QmKwCkU8G6V5aWpFbCj3u1f3XJHny2K5Z3u1V3HfHYrC
/ip4/76.16.47.205/tcp/22924/ipfs/QmW2qvoQvQ59nks3Hf65QYF8nmXX8DneUakPCvexf8
/ip4/78.46.85.216/tcp/4001/ipfs/QmSofF1cXgZqz3nyJ3mG2h7n5s9pFaluqweD8ktCvRk1H
/ip4/88.99.241.124/tcp/4001/ipfs/QmFPZcnVAEjXAB1A7StETRUKK58FzNt96BZHybnJR7oci
/ip4/95.216.204.56/tcp/4001/ipfs/QmRpnS0z3ME85HY9Mq8SzhKAbXto6Xq65Utzx3Ms3UoHn
/ip4/99.98.18.72/tcp/4001/ipfs/QmTbtPCLYmMtr5VRtbsB8TRYS9HzfKfHVZsFhzQRNnfsQk
/ipfs/QmTbtPCLYmMtr5VRtbsB8TRYS9HzfKfHVZsFhzQRNnfsQk/pzp-circuit/ipfs/QmUEKf5F3eQGB9HP2uAvbFTmVo5p3Ere1CXvTuWnu
C:\Users\Niron\Downloads>go-ipfs>
```

11. Get the path of your **build/** folder (review step 6) and run the command:

```
ipfs add -r <build_folder_location>
```

This will add your folder to the IPFS network.

```
C:\Windows\System32\cmd.exe
C:\Users\Niron\Downloads>go-ipfs>ipfs add -r D:\KINGSLAND\decentralized-casino\build
1.01 KiB / 7 [-----]
added QmesuVZ8yBeyw9RG5YzVFjEJeZ2nenerASDrnv83LiZKnh9 build/asset-manifest.json 0.05% 00m34s
2.13 KiB / 4.52 MiB [-----]
added QmZxndFzKvG6nDhKcu15RX7j2yz9EKZ1mmv5boojaj8X8 build/favicon.ico 0.09% 00m24s
4.27 KiB / 4.52 MiB [-----]
added QmVinTaChdLHtceUoh1kVkuBfjF3sSN6dMg39YwT22G build/index.html 0.10% 00m39s
4.58 KiB / 4.52 MiB [-----]
added QmSpHt6NJawSaCjMcbHy8Vp4LPXseyreWap9Rj8bdF2L build/manifest.json 0.11% 00m41s
5.22 KiB / 4.52 MiB [-----]
added QmRG35uBq7n1C1vMDZ5H1477RnLKtbnftJXeEmMcRn6 build/precache-manifest.9ae016205cf88082378898f5f307adc.js 0.11% 00m47s
5.29 KiB / 4.52 MiB [-----]
added QmZxX6rtNN7V3nuPS81r4EctYbQ3vk7XrqQmSnnQzuV7Mt build/robots.txt 0.14% 00m45s
6.44 KiB / 4.52 MiB [-----]
added QmV5sJRBK2m29FgkpbAdCuCFHbZR42c7q2tR2N2oKd build/service-worker.js 0.16% 00m44s
7.36 KiB / 4.52 MiB [-----]
added QmG5Cztuk4hu7Fk4VfV2hm2zoHaZTd3DwZfHq5nhsb6PW build/static/css/main.19386453.chunk.css 0.20% 00m39s
9.13 KiB / 4.52 MiB [-----]
added Qma9q3a2ygpPPVYXEAwaDEtnvXzjkVXcx22BEoUuhZyCwqm build/static/css/main.19386453.chunk.css.map 22.70%
1.03 MiB / 4.52 MiB [-----]
added QmbgX3tpAdh3Jfg5aMurrMzQnRsnf226nqEnkH0ra6UDYpq build/static/js/2.efc03dab.chunk.js 22.73%
1.03 MiB / 4.52 MiB [-----]
added QmVbZgNGF5DTH2zhmym2geVqRjMpSPz5X52E5yt1Gfe build/static/js/2.efc03dab.chunk.js.LICENSE.txt 94.41%
4.27 MiB / 4.52 MiB [-----]
added QmdqUX6dJwKMGebAHvxoURIDVYyZAGmwZakx9bdobxq29L build/static/js/2.efc03dab.chunk.js.map 99.49%
4.50 MiB / 4.52 MiB [-----]
added QmReJjyXT07aNVDFeT3bpP1sY2AUJDBHJZvc4y1i3UNW build/static/js/main.c1339dd4.chunk.js 99.79%
4.51 MiB / 4.52 MiB [-----]
added Qm9RzUCa8NjagQkyXHLGH4HQDQFmudKbWi68xoRTZBHP build/static/js/main.c1339dd4.chunk.js.map 100.00%
4.51 MiB / 4.52 MiB [-----]
added QmTuwGZbZubckrxuuxWdg2Xwgop8B8zo1GxirLmPVfH3Z build/static/js/runtime-main.939ad078.js 100.00%
4.52 MiB / 4.52 MiB [-----]
added QmVNBd37wp64XldLr1P1RjUtv51GHSySL7oxiWmJ9c7S build/static/js/runtime-main.939ad078.js.map 100.00%
4.52 MiB / 4.52 MiB [-----]
added QmZ19LHt1HJPdZpJgTHKmrktkp4KbEGMoIXZuxih3Sw build/static/css 100.00%
4.52 MiB / 4.52 MiB [-----]
added Qm7dnBhKvZbyyqjhoH46cSGFD6KPhaZmaufS7HxYs build/static/js 100.00%
4.52 MiB / 4.52 MiB [-----]
added QmadfVTPgEkip1ACwBhBmBwGDNutFWsp2gaF94PU1528A build/static 100.00%
4.52 MiB / 4.52 MiB [-----]
added Qma47iUG7kqHjt9S3FzXbcSWKjFXDjgN8YdfpMHYJMy1Q build 100.00%
4.52 MiB / 4.52 MiB [-----]
C:\Users\Niron\Downloads>go-ipfs>
```


12. Copy the last hash. For example, “Qma47iUG7KqXHjt9S3FzXbCsWKJFXDjqN8YdfpMHYJMy1Q”:

```
added QmX19LAC1H5puzFjqHkMlCkKpKX0ESqW01X20X1H9SA build/static/css 100.00%
4.52 MiB / 4.52 MiB [=====]
added Qm07dnbWtkVZbwy8Jmoh46sCsGFD6KPhAKizmauF57HXYs build/static/js 100.00%
4.52 MiB / 4.52 MiB [=====]
added QmadFWTPgEKpiYACwBHnBwGDMUtFWsp2gaEf94PUi5z8A build/static 100.00%
4.52 MiB / 4.52 MiB [=====]
added Qma47iUG7KqXHjt9S3FzXbCsWKJFXDjqN8YdfpMHYJMy1Q build 100.00%
4.52 MiB / 4.52 MiB [=====]
C:\Users\niron\Downloads\go-ipfs>
```

13. Run the following command to finally publish your files in the IPFS network:

```
ipfs name publish <hash_of_build_folder>
```

```
C:\Windows\System32\cmd.exe
C:\Users\niron\Downloads\go-ipfs>ipfs name publish Qma47iUG7KqXHjt9S3FzXbCsWKJFXDjqN8YdfpMHYJMy1Q
Published to QmPNitCfRRquhGCJ2JJA2WPAYMb95tcjHwNxSuyuYwFcX: /ipfs/Qma47iUG7KqXHjt9S3FzXbCsWKJFXDjqN8YdfpMHYJMy1Q
C:\Users\niron\Downloads\go-ipfs>
```

14. Open the following link. For example:

<https://gateway.ipfs.io/ipfs/QmbiA7itaE3uomsJq6XPDqbwbwYzMCzrYL6uz7Xa4AwY8/>

15. If you make changes to your files remember to execute:

```
npm run build
ipfs add -r <build_folder_location>
ipfs name publish <hash_of_build_folder>
```

16. Congratulations, you just deployed your decentralized application in the IPFS network!

gateway.ipfs.io/ipfs/QmVgy6AJK1qifxBGer7JXptWRwv5pD3eCnY/C8HxQVHCd/

Bet for your best number and win huge amounts of Ether

Number of bets: 2
Last winning number: No draws yet
Total ether bet: 1 ether
Minimum bet: 0.1 ether
Max number of bets: 2

Vote for the next number

How much Ether do you want to bet? ether

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|----|

Only working with the Ropsten Test Network
You can only vote once per account
Your vote will be reflected when the next block is mined

What to Submit?

Create a **zip file** (e.g. **your-name-decentralized-casino-metamask-provable-exercise.zip**) containing the following:

1. A **links.txt** file containing your:
 - a. IPFS dApp link.
 - b. Etherscan contract link.
2. Screenshots of the terminal: **truffle compilation, truffle migration**.

Submit your **zip** file as **homework** at the course platform.