GAME TUTORIAL

MISE EN PLACE DE L'ENVIRONNEMENT

Installation de nodejs

sudo apt-get update

sudo apt-get install nodejs npm

```
nathan@nathan-VirtualBox:~$ node -v
v10.19.0
```

Installation de truffle

sudo npm install -g truffle

```
nathan@nathan-VirtualBox:~$ sudo npm install -g truffle
[ extract es-abstract@^1.17.0-ne
```

Téléchargement et installation de ganache

Aller sur le lien suivant : https://www.trufflesuite.com/ganache et télécharger ganache



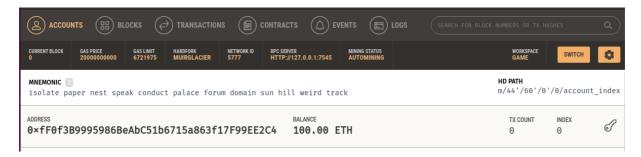
Une fois téléchargé, faire clique droit dessus et aller dans propriété et permission et cocher autoriser l'exécution du fichier comme programme

Exécution : Autoriser l'exécution du fichier comme un programme

```
nathan@nathan-VirtualBox:~$ cd Bureau/
nathan@nathan-VirtualBox:~/Bureau$ mkdir Ganache
nathan@nathan-VirtualBox:~/Bureau$ cd Ganache/
nathan@nathan-VirtualBox:~/Bureau/Ganache$ ls
ganache-2.5.4-linux-x86_64.AppImage
nathan@nathan-VirtualBox:~/Bureau/Ganache$ ./ganache-2.5.4-linux-x86_64.AppImage
bash: ./ganache-2.5.4-linux-x86_64.AppImage: Permission non accordée
nathan@nathan-VirtualBox:~/Bureau/Ganache$ sudo ./ganache-2.5.4-linux-x86_64.AppImage
sudo: ./ganache-2.5.4-linux-x86_64.AppImage : commande introuvable
nathan@nathan-VirtualBox:~/Bureau/Ganache$ ./ganache-2.5.4-linux-x86_64.AppImage

19:05:33.053 > Checking for update
19:05:33.327 > Generated new staging user ID: 996b3761-904d-5e11-b961-931feecc19
d4
19:05:35.797 > Update for version 2.5.4 is not available (latest version: 2.5.4, downgrade is disallowed).
```

Maintenant, cliquer sur New workspace et le nommer comme voulu. J'ai opté pour Game



Installation de Metamask

Ouvrir le navigateur et aller dans plug-in et extensions puis chercher metamask et cliquer sur ajouter



MetaMask

par danfinlay, kumavis



Ensuite ouvrir metamask et choisir importer le portefeuille





Importez un compte avec une phrase mnémotechnique

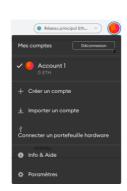


Pour la seed du portefeuille, elle se trouve sur ganache

MNEMONIC ?

isolate paper nest speak conduct palace forum domain sun hill weird track

Pour le mdp j'ai choisi : Metamask



Installation de SublimeText

wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add - sudo apt-get install apt-transport-https sudo apt-get update

sudo apt-get update
sudo apt-get install sublime-text



MISE EN PLACE DU PROJET

Maintenant que les différents outils sont installés, nous pouvons commencer le projet

Commençons par importer la base du projet

sudo apt install git

git clone -b starter-code https://github.com/dappuniversity/blockchain_game blockchain_game

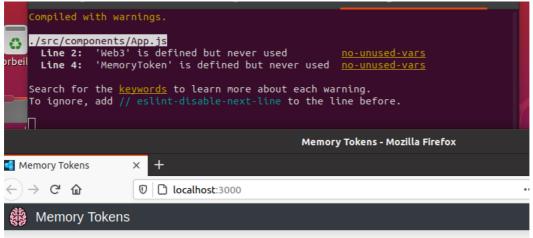
```
nathan@nathan-VirtualBox:~/Bureau$ git clone -b starter-code https://github.com/dappuniversity/blockchain_game blockchain_game Clonage dans 'blockchain_game'... remote: Enumerating objects: 91, done. remote: Counting objects: 100% (91/91), done. remote: Compressing objects: 100% (52/52), done. remote: Total 91 (delta 41), reused 86 (delta 36), pack-reused 0 Dépaquetage des objets: 100% (91/91), 556.53 Kio | 202.00 Kio/s, fait. nathan@nathan-VirtualBox:~/Bureau$
```

Aller dans le repertoire blockchain_game et installer

```
nathan@nathan-VirtualBox:~/Bureau$ cd blockchain_game/
nathan@nathan-VirtualBox:~/Bureau/blockchain_game$ npm install
[________] \ extract:source-map: sill extract source-map@0.6.1 extrac
```

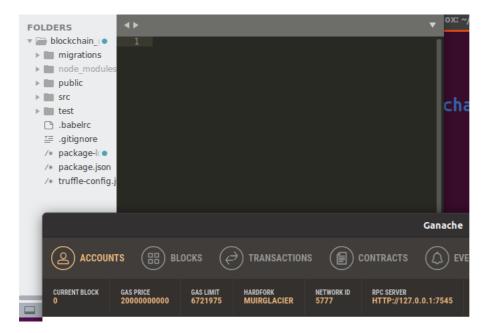
added 2390 packages from 1217 contributors and audited 2460 packages in 194.293s

Une fois fini, lancer le serveur : npm run start



Edit this file in App.js!

Le serveur fonctionne bien. Maintenant on lance ganache et on ouvre le projet dans sublime text



Avec sublimeText, il faut aller dans contracts/MemoryToken et on va remplir cette partie

```
pragma solidity ^0.5.0;
import "./ERC721Full.sol";

contract MemoryToken is ERC721Full {
    constructor() ERC721Full("Memory Token", "MEMORY") public {
    }

    function mint(address _to, string memory _tokenURI) public returns(bool) {
        uint _tokenId = totalSupply().add(1);
        _mint(_to, _tokenId);
        _setTokenURI(_tokenId, _tokenURI);
        return true;
    }
}
```

Ce code permet de nommer le contrat « Memory Token »

Il faut donc aller dans Migration/2_deploy... et modifier afin de cibler le contrat « Memory Token »

```
const MemoryToken = artifacts.require("MemoryToken");
module.exports = function(deployer) {
  deployer.deploy(MemoryToken);
};
```

Maintenant, taper la commande truffle migrate—reset dans le terminal pour être sûr d'exécuter le bon smart contrat

```
nathan@nathan-VirtualBox:~/Bureau/blockchain_game$ truffle migrate --reset
Compiling your contracts...
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling ./src/contracts/ERC721Full.sol
> Compiling ./src/contracts/MemoryToken.sol
> Compiling ./src/contracts/Migrations.sol
> Artifacts written to /home/nathan/Bureau/blockchain_game/src/abis
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
Starting migrations...
============
                  'development'
> Network name:
> Network id:
                  5777
 Block gas limit: 6721975 (0x6691b7)
```

Tout semble fonctionner, les contrats migrations et Memory Token ont migré correctement.

Maintenant, lancer la console truffle avec : truffle console et taper token = await MemoryToken.deployed()

Puis token pour voire toutes les informations

On peut par exemple trouver l'adresse avec token.address et le nom avec name = await token.name() et name

```
truffle(development)> token.address
'0x9956CdA43D55E7B4A61C7CC4F1217DA9aE266615'
truffle(development)> name = await token.name()
undefined
truffle(development)> name
'Memory Token'
truffle(development)>
```

Pour quitter, écrire : .exit

Ainsi, on fait les tests directement dans la console. On les stock dans une variable pour on affiche la variable.

Maintenant, le smart contrat est sur la blockchain. On a la garantie que tout fonctionne bien.

On va faire un test avec le fichier Memory token test ; taper le code suivant :

```
const MemoryToken = artifacts.require('./MemoryToken.sol')
require('chai')
   use(require('chai-as-promised'))
   should()
contract('Memory Token', (accounts) => {
  let token
     before(async () => {
  token = await MemoryToken.deployed()
    })
    describe('deployment', async () => {
    it('deploys successfully', async () => {
        const address = token.address
        assert.notEqual(address, 0x0)
        assert.notEqual(address, untl)
        assert.notEqual(address, untl)
        assert.notEqual(address, undefined)
})
        it('has a name', async () => {
  const name = await token.name()
  assert.equal(name, 'Memory Token')
})
   it('has a symbol', async () => {
    const symbol = await token.symbol()
    assert.equal(symbol, 'MEMORY')
})
     describe('token distribution', async () => {
  let result
        it('mints tokens', async () => {
   await token.mint(accounts[0], 'https://www.token-uri.com/nft')
              // It should increase the total supply
result = await token.totalSupply()
assert.equal(result.toString(), '1', 'total supply is correct')
              // It increments owner balance
result = await token.balanceOf(accounts[0])
assert.equal(result.toString(), '1', 'balanceOf is correct')
              // Token should belong to owner
result = await token.ownerOf('1')
assert.equal(result.toString(), accounts[0].toString(), 'ownerOf is correct')
result = await token.tokenOfOwnerByIndex(accounts[0], 0)
              // Owner can see all tokens
let balance0f = await token.balance0f(accounts[0])
let tokendfos = []
for (let i = 0; i < balance0f; i++) {
let id = await token.token0fOwnerByIndex(accounts[0], i)
tokenIds.push(id.toString())</pre>
              let expected = ['1']
assert.equal(tokenIds.toString(), expected.toString(), 'tokenIds are correct')
              // Token URI Correct
let tokenURI = await token.tokenURI('1')
assert.equal(tokenURI, 'https://www.token-uri.com/nft')
   })
```

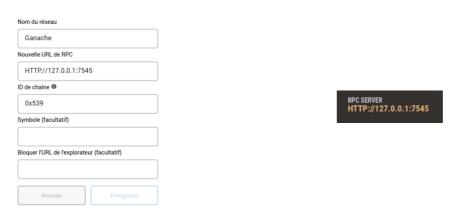
Ce code permet de faire les tests directement depuis le fichier ainsi lorsque l'on effectue la migration, on a les résultats. De cette façon, on a plus besoin de faire les tests directement dans la console avec la commande await.

Et le lancer avec la commande truffle test

Puis reset la migration avec la commande truffle migrate -reset

Maintenant, on va vérifier que metamask est connecté à la blockchain

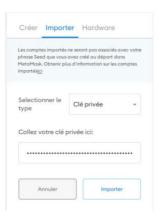
Aller dans Réseaux/ RPC personnalisé et choisir un nom et mettre comme nouvelle URL de RPC celle de ganche



Maintenant, on va importer un compte sur metamask, pour ce faire il faut copier la clé privée du compte



Et coller cette clé dans importer compte sur metamsk



CODAGE DE L'APPLICATION

Voici le code total à écrire :

```
import React, { Component } from 'react';
import web3 from 'web3'
import '.App.css';
import MemoryToken from '../abis/MemoryToken.json'
import brain from '../brain.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 }
else if (window.web3) {
window.web3 = new Web3(window.web3.currentProvider)
const CARD_ARRAY = [
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          async loadBlockchainData() {
  const web3 = window.web3
  const accounts = await web3.eth.getAccounts()
  this.setState({ account: accounts[0] })
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 this.setState(f account; accountied))

// Load smart contract
const networkid = must veb3.eth.net.getId()

// Load smart contract
const networkid = must veb3.eth.net.getId()

// Const const const = must veb3.eth.networkid;
const tobe = now web1.eth.networkid;
const tobe = now web2.eth.networkid;
const tobe
                                   name: 'cheeseburger',
img: '/images/cheeseburger.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        SMane—"container—Tuul mt-5">
as bhome" row;
as bhome" row;
as bhome "row;
v class blane="content mr-auto mt-auto">
int class blane="d-4">fart matching nouto'>
int class blane="d-4">fart matching nouto'>
int class blane="d-4">fart matching nouto'>
it class blane="grid mb-4">
it this, state.cardArray.map((card, key) => {
return[
                                 name: 'pizza',
img: '/images/pizza.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                word to the property of the pr
                                 name: 'milkshake',
img: '/images/milkshake.png'
                                   name: 'fries',
img: '/images/fries.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      }
else if(this.state.cardsChosenId.includes(cardId)) {
  return CARD_ARRAY[cardId].ing
                                 name: 'ice-cream',
img: '/images/ice-cream.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  this.setState({
    cardsChosen: [...this.state.cardsChosen, this.state.cardArray[cardId].nc
    cardsChosenId: [...this.state.cardsChosenId, cardId]
}}
                                 name: 'pizza',
img: '/images/pizza.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      if (alreadyChosen === 1) {
   setTimeout(this.checkForMatch, 100)
                                 name: 'milkshake',
img: '/images/milkshake.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      heckForMatch = async () => {
  const optionOmeId = this.state.cardsChosenId[0]
  const optionTwoId = this.state.cardsChosenId[1]
                                   name: 'hotdog',
img: '/images/hotdog.png'
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       send({ from: this.state.account })
.on('transactionWash', (hash) ≈ {
this.setState.i., this.state.cardsWon, optionOmeId, optionTwoId),
.cardsWont [...,this.state.tokenRits, cAMO_AMRAY(optionOmeId), ing]
.cokenRits [...,this.state.tokenRits, cAMO_AMRAY(optionOmeId), ing]
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    })
} else {
alert('Sorry, try again')
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    this.setState({
    cardsChosen: [],
    cardsChosenId: []
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         (this.state.cardsWon.length === CARD_ARRAY.length) {
alert('Congratulations! You found them all!')
```

Je vais en commenter quelques parties mais tout est très bien expliqué dans la vidéo avec des exemples à chaque fois : https://www.youtube.com/watch?v=x-6rugmNS30

Ici, on charge deux fonctions: loadweb3 et loadBlockchaindata. La première sert à nous permettre d'intéragir avec un nœud Ethereum local utilisant HTTP, IPC ou WebSocket. On vérifie qu'une fenêtre est ouverte sinon on l'ouvre et sinon on conseil d'ouvrir metamask. La seconde sert à vérifier qu'une connexion est bien établie entre un compte metamask et la session de jeu en vérifiant id, adresse etc.

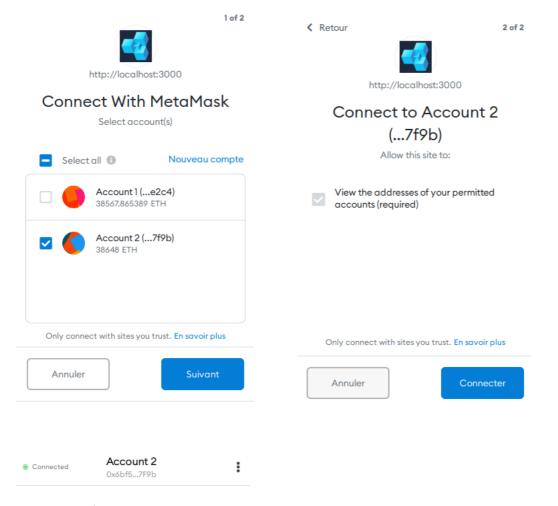
Ici, les deux premières fonctions servent à comprendre sur quelle image on a cliqué et vérifier qu'on a le droit de le faire puis si tout est bon la carte se retourne. La troisième fonction sert à tester si deux cartes retournées sont les mêmes. Si oui le message « Congratulation.... » apparaît et les cartes disparaissent du jeu sinon « try again » et les cartes sont retournées.

```
chosestage = (corde) = (
corded = corded) = (
corded = corded = corded) = (
corded = corded =
```

Ici, cette fonction sert à mettre en place l'aspect visuel du jeu. Taille des cartes etc. De plus, elle sert aussi à gérer les token : c'est-à-dire incrémenter quand une paire est trouvée.

Maintenant, on effectue la migration : truffle migrate --reset

Après avoir fait la migration finale, il faut connecter un compte à metamask. C'est le compte qui va être utilisé pour jouer la partie.



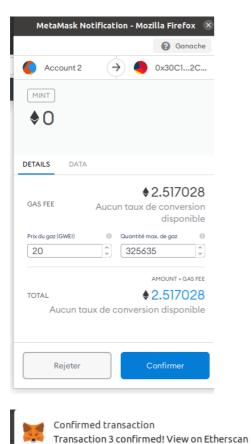
Ensuite, on refresh la page et on peut jouer



Lorsque l'on a une paire, cette fenêtre s'ouvre



Il faut confirmer la transaction pour inscrire dans la blockchain la transaction et ainsi prouver que l'on a bien trouvé une paire



On confirme et le nombre de token est incrémenté ainsi que l'image de la paire trouvée

Tokens Collected: 1



Quand on se trompe, cette fenêtre s'ouvre



Une fois terminé, voilà le résultat

Start matching now!



Pour conclure, à la différence du jeu classique, ici chacune des actions est inscrite sous forme d'une transaction dans la blockchain.