|  |  |  |  |
| --- | --- | --- | --- |
| VERSIONS | DATE | OBJET | NOM |
| A | 03/01/2018 | First draft | FH |
| A | 04/01/2018 | modification suite là la réunion | FH |
| B | 17/05/2018 | Migration vers Infura | YH |
| B | 23/07/2018 | Documentation Process Privilège | YH |

Sommaire

[1 Introduction](#__RefHeading___Toc309310063) 3

[1.1 Pré-requis](#__RefHeading___Toc309310064) 3

[1.2 Principe de fonctionnement](#__RefHeading___Toc309310065) 3

2 [Démarrage de pré-production](#__RefHeading___Toc309310071) 4

2[.1 Démarrage](#__RefHeading___Toc309310072) 4

2[.2 Test du contrat DonationV2 via MyEtherWallet et metamask](#__RefHeading___Toc43_969533433) 4

2[.3](#__RefHeading___Toc45_969533433) Test du contrat DonationV2 via remix et metamask6

3 Tests et interaction avec le smart contrat déployé 8

4 Déploiement via remix (donation process privilège [1](#__RefHeading___Toc30931007211)0

5 Tests [1](#__RefHeading___Toc3093100731)2

5.[1 Déroulement du fichier test donation.js 1](#__RefHeading___Toc693_8737130582)2

5.2 Main scénario sur ropsten………………………………………………………………………………….13

# Introduction

Ce document contient les divers éléments relatifs à la supervision et à l’exploitation de l’application POC Donation

Il s’adresse à des exploitants et à des administrateurs

## Pré-requis

**Liste des pré-requis nécessaires à l’exploitation de l’application et l’exécution des actions décrites dans ce documents**

| N° | Pré requis | Illustration |
| --- | --- | --- |
|  | truffle | sudo npm install -g truffle |
|  | metamask | Extension google |

## **Principe de fonctionnement**

Ethereum est un logiciel permettant la validation de transaction de façon décentralisée via un réseau peer-to-peer.

La mise en place du service de donation nécessite un nœud du réseau ethereum sur lequel un smart contrat sera deployé.

Ce document explique comment lancer un nœud ethereum et déployer un service de donation à travers un smart contrat.

# Démarrage de pré-production

## Démarrage

Décrire le mode opératoire de démarrage initial de l’application

## Test du contrat DonationV2 via MyEtherWallet et metamask (version initiale)

Une fois le contrat déployé sur la la blockchain de test (ropsten), le contrat peut être utilisé.

| Étape | Action | Illustration |
| --- | --- | --- |
| 1 | Aller sur le site de MyEtherwallet.  <https://www.myetherwallet.com/>  Sous l’onglet « Contracts », il est possible d’interagir avec le smart contract.   * Coller l’adresse du contrat déployé (cercle rouge) sous 0x8dce0cb938afb96420f3c373cd1a2c7432bf11c4 * Coller l’abi du smart contrat dans la partie ABI/Json interface (cercle vert)   Ps : l’abi du contrat déployé :  [ { "anonymous": false, "inputs": [ { "indexed": false, "name": "amount", "type": "uint256" }, { "indexed": false, "name": "nbBenef", "type": "uint256" } ], "name": "evtSpread", "type": "event" }, { "constant": false, "inputs": [], "name": "flush", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": false, "inputs": [], "name": "kill", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": false, "inputs": [ { "name": "\_beneficiary", "type": "address" } ], "name": "registerBeneficiary", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "constant": false, "inputs": [ { "name": "\_newAdmin", "type": "address" } ], "name": "transferAdminRights", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "anonymous": false, "inputs": [ { "indexed": false, "name": "benef", "type": "address" }, { "indexed": false, "name": "amount", "type": "uint256" } ], "name": "evtSendSuccess", "type": "event" }, { "constant": false, "inputs": [ { "name": "\_newCertifier", "type": "address" } ], "name": "transferCertifierRights", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "anonymous": false, "inputs": [ { "indexed": false, "name": "benef", "type": "address" }, { "indexed": false, "name": "amount", "type": "uint256" } ], "name": "evtSendFailed", "type": "event" }, { "constant": false, "inputs": [ { "name": "\_beneficiary", "type": "address" } ], "name": "unregisterBeneficiary", "outputs": [], "payable": false, "stateMutability": "nonpayable", "type": "function" }, { "payable": true, "stateMutability": "payable", "type": "fallback" }, { "inputs": [ { "name": "\_certifier", "type": "address" } ], "payable": false, "stateMutability": "nonpayable", "type": "constructor" }, { "constant": true, "inputs": [], "name": "getBeneficiaryCount", "outputs": [ { "name": "", "type": "uint256" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": true, "inputs": [ { "name": "startIndex", "type": "uint256" }, { "name": "size", "type": "uint256" } ], "name": "getPaginateActiveBeneficiaries", "outputs": [ { "name": "", "type": "address[100]" } ], "payable": false, "stateMutability": "view", "type": "function" }, { "constant": true, "inputs": [ { "name": "startIndex", "type": "uint256" }, { "name": "size", "type": "uint256" } ], "name": "getPaginateBeneficiaries", "outputs": [ { "name": "", "type": "address[100]" } ], "payable": false, "stateMutability": "view", "type": "function" } ]   * Cliquer bouton « Access »   Une fois l’accès à l’interface du smart contrat est effectué. Il suffit de choisir une fonction et l’exécuter. |  |

## Test du contrat DonationV2 via remix et metamask (version initiale)

Une fois le contrat déployé sur la la blockchain de test (testnet), le contrat peut être utilisé. Le plugin Metamask permet de se connecter à un réseau ethereum (ici ropsten testnet) et propose une interface graphique pour la création d’un compte, et l’accès au réseau (unlock). Le site web Remix ([https://remix.ethereum.org](https://remix.ethereum.org/)) est un framework de smart contrat permettant la création/compilation/deploiement de contrat solidity. Remix et metamask peuvent fonctionner ensemble et ainsi permettre la gestion/adminstration/test des « smart contract ».

| Étape | Action | Illustration |
| --- | --- | --- |
| 1 | Aller sur le site de Remix IDE.  [https://remix.ethereum.org](https://remix.ethereum.org/)  Dans l’onglet « Settings », il est possible de préciser la version du compilateur solidity. (4.19 actuellement).  Dans l’onglet « Compile », il faut coller le smart contrat.  Dans l’onglet « Run », si metamask est installé et le compte ethereum débloqué alors Remix initialise son envirronnement (cercle rouge) automatiquement avec le réseau choisi par metamask (ici ropsten(3) … le 3 represente l’identfiant du réseau).  Injected web3 signifie que remix injecte les commandes sur le nœud ethereum spécifié (dans notre cas metamask indique ropsten test net).  On peux se connecter à un contrat particuliers en précisant son adresse et en cliquant sur « at address » ou bien créer une nouvelle instance de contrat en précisant l’adresse du certifieur (entre guillemet) et en cliquant sur « create ». (cercle vert) |  |
| 2 | Une fois Remix connecté au réseau et au contrat, il est possible d’interagir avec le contrat soit via l’interface graphique de remix soit via metamask.  les adresses passées en parametres aux fonction du contrat doivent etre entourée de guillemets. |  |
| 3 | Toutes les fonctions sont alors testables directement via remix.  exemple :  pour détruire le smart contract, il suffit de cliquer sur la fonction kill et signer la transaction (via metamask) |  |

# 3 Tests et interaction avec le smart contrat déployé (donationV2.sol version initiale)

| Étape | Action | Illustration |
| --- | --- | --- |
| 1 | Aller sur le site de Remix IDE.  [https://remix.ethereum.org](https://remix.ethereum.org/)  Dans l’onglet « Settings », il est possible de préciser la version du compilateur solidity. (4.19 actuellement).  Dans l’onglet « Compile », il faut coller le smart contrat.  Dans l’onglet « Run », si metamask est installé et le compte ethereum débloqué alors Remix initialise son envirronnement (cercle rouge) automatiquement avec le réseau choisi par metamask (ici ropsten(3) … le 3 represente l’identfiant du réseau).  Injected web3 signifie que remix injecte les commandes sur le nœud ethereum spécifié (dans notre cas metamask indique ropsten test net).  On peux se connecter à un contrat particuliers en précisant son adresse et en cliquant sur « at address » ou bien créer une nouvelle instance de contrat en précisant l’adresse du certifieur (entre guillemet) et en cliquant sur « create ». (cercle vert) |  |
| 2 | Une fois Remix connecté au réseau et au contrat, il est possible d’interagir avec le contrat soit via l’interface graphique de remix soit via metamask.  les adresses passées en parametres aux fonction du contrat doivent etre entourée de guillemets. |  |
| 3 | Pour détruire le smart contract, il suffit de cliquer sur la fonction kill et signer la transaction (via metamask) |  |

# **4** **Déploiement via remix (smart contact donation process privilège)**

**Cette procédure de déploiement est valable sur ropsten (testnet) ou bien sur le mainnet (réseau production d’ethereum)**

| Étape | Action | Illustration |
| --- | --- | --- |
| 1 | Aller sur le site de Remix IDE.  [https://remix.ethereum.org](https://remix.ethereum.org/)  Dans l’onglet « Settings », il est possible de préciser la version du compilateur solidity. (4.25 actuellement).  Dans l’onglet « Compile », il faut coller le smart contrat.  Dans l’onglet « Run », si metamask est installé et le compte ethereum débloqué alors Remix initialise son envirronnement (cercle rouge) automatiquement avec le réseau choisi par metamask (ici ropsten(3) … le 3 represente l’identfiant du réseau).  Injected web3 signifie que remix injecte les commandes sur le nœud ethereum spécifié (dans notre cas metamask indique ropsten test net).  On peut se connecter à un contrat particuliers en précisant son adresse et en cliquant sur « at address » ou bien créer une nouvelle instance de contrat en cliquant sur « create ». (cercle vert) |  |
| 2 | **Deploy the donation contract**   1. Une fois les contrats importés sur remix, on clique sur le boutton « deploy » 2. Une fois le contrat est déployé, on récupère l’adresse pour déployer le contrat privilège request   => contrat déployé sous l’adresse ‘’0x4dd5f5d49cb78c1b2b73e6e42fafe23f528bc495’’  **Deploy the privilegeRequest contract**   1. Sélectionner le contrat « privilegeRequest », on passe comme paramètre l’adresse du contrat donation   **notre cas**  ‘’0x4dd5f5d49cb78c1b2b73e6e42fafe23f528bc495’’   1. Une fois le contrat est déployé, on récupère l’adresse pour mettre à jour l’adresse du contrat privilège request dans le contrat donation.   => contrat déployé sous ‘’0xba3ca10fd248886890141616fbe3e437ea9f0d40’’  **Mettre à jour l’adresse du privilege request sur le contrat donation**   1. Faire appel à la fonction setPrivilegeRequestAddress en passant l’adresse du contrat privilegeRequest ‘’0xba3ca10fd248886890141616fbe3e437ea9f0d40’’ | **Deploy the donation contract**  Transaction Hash <https://ropsten.etherscan.io/tx/0x9780bb620ad297d68176247d9baf17b955583dcdcd314c0950314258fc9bc375>  **Deploy the privilegeRequest contract**  Transaction Hash  <https://ropsten.etherscan.io/tx/0xb75aa29a8ea4a613767d8070a5909811fbc2601517d3afc041ebd59131210c5d>  **Mettre à jour l’adresse privilegeRequest sur le contrat donation**  Transaction hash  <https://ropsten.etherscan.io/tx/0xe66e32deb9a9c39472bdd3b9aa9726b24ecf360ead8578a9ada4b89f09d86ff6> |

**5 Tests**

**5.1 Déroulement du fichier test donation.js**

| Étape | Action | Illustration |
| --- | --- | --- |
| 1 | Lancer l’application ganache | Avec la configuration décrite sur le document « Installation POC Donation.doc » |
| 2 | Dérouler le fichier donation.js (une fois sur le répertoire du projet) | Truffle test test/donation.js  **Résultats** |

**5.2 Main scénario sur ropsten**

**Adresse contrat donation** 0x4dd5f5d49cb78c1b2b73e6e42fafe23f528bc495

* Transaction Hash <https://ropsten.etherscan.io/tx/0x9780bb620ad297d68176247d9baf17b955583dcdcd314c0950314258fc9bc375>

**Adresse contrat privilegeRequest** 0xba3ca10fd248886890141616fbe3e437ea9f0d40

* Transaction Hash <https://ropsten.etherscan.io/tx/0xb75aa29a8ea4a613767d8070a5909811fbc2601517d3afc041ebd59131210c5d>

**Initialisation de l’adresse privilegeRequest sur le contrat donation**

* Transaction Hash <https://ropsten.etherscan.io/tx/0xe66e32deb9a9c39472bdd3b9aa9726b24ecf360ead8578a9ada4b89f09d86ff6>

**Alice envoie une transaction avec 0.01 ether au contrat de privilegeRequest**

**Alice Adresse** 0xAF13F12EBE00cd02F486E000927408148e768802

* Transaction Hash[**https://ropsten.etherscan.io/tx/0xfd5019e630a5031851d689ac396c5938c5727a2594d37a9b62970baa20ed1ad2**](https://ropsten.etherscan.io/tx/0xfd5019e630a5031851d689ac396c5938c5727a2594d37a9b62970baa20ed1ad2)
* **Résultat de la transaction** 
  + Le contrat privilège d’Alice est créé ‘’0xfEfdA2c7cd6B5C8e9481404E7D64Fac19E545439’’
  + Alice reçoit 0.01 ether de la part de son contrat de privilège
  + Le contrat d’Alice est ajouté comme bénéficiaire sur le contrat de donation
  + Alice a 0 privilège sur le contrat donation

**Bernard envoie 1 ether au contrat de privilège d’Alice**

**Bernard adresse** 0x86dda0810BdfAE8d9EceD3618Ef2C219bC7b05a7

* Transaction Hash <https://ropsten.etherscan.io/tx/0x90d9e7a76b4a84c8b7bac9b1d3fdfc0d141b708ac4ffa9da4342503ecfbd7869>
* Résultat de la transaction
  + Le contrat de privilège d’Alice transfer 1 ether au contrat donation
  + Le contrat de privilège d’Alice met à jour ses privilèges, Alice a alors 1 privilège

**David envoie une transaction avec 0.01 ether au contrat de privilegeRequest**

**David Adresse** 0x9de9eda2dFE06D77886B34D2F24964D05B38F74f

* **Transaction Hash** <https://ropsten.etherscan.io/tx/0x8c053815d5a79911515ed896910aa89ae83906b3149ca9218fea7d52656e1ce6>
* **Résultat de la transaction** 
  + Le contrat privilège de David est créé ‘’0xDa221D345950eC7246ab70f03D9dE11a747998A0’’
  + David reçoit 0.01 ether de la part de son contrat de privilège
  + Le contrat de David est ajouté comme bénéficiaire sur le contrat de donation
  + David a 0 privilège sur le contrat donation

**Bernard appelle la fonction « distribute »**

* **Transaction Hash** <https://ropsten.etherscan.io/tx/0xf1ce8e14a6b3ce03e07e6dab144de1ca0ecf4715d600acf2d0911bb58c710e6b>
* **Résultat de la transaction**
  + Alice reçoit 0.01 ether du contrat donation, car elle a un privilège de 1.
  + Alice a maintenant 0.995 privilèges
  + David ne reçoit pas une donation car il a 0 privilèges

**Bernard appelle la fonction « distribute » une deuxième fois sans attendre le délai d’une heure**

* **Transaction Hash** <https://ropsten.etherscan.io/tx/0x5931d6e6299af413d5956fdd4f74b1d6901539b34f9c35a090463b433d583a09>
* **Résultat de la transaction** 
  + La transaction échoue