

Département des Systèmes d'Information

Application POC Ethereum (Donation)

Dossier d'exploitation

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

Sommaire

1 Introduction.....	3
1.1 Pré-requis.....	3
1.2 Principe de fonctionnement.....	3
2 Démarrage de pré-production.....	4
2.1 Démarrage.....	4
2.2 Test du contrat DonationV2 via MyEtherWallet et metamask	4
2.3 Test du contrat DonationV2 via remix et metamask.....	6
3 Tests et interaction avec le smart contrat déployé	8
4 Déploiement via remix (donation process privilège).....	10
5 Tests.....	12
5.1 Déroulement du fichier test donation.js	12
5.2 Main scénario sur ropsten.....	13

1 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

1.1 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
1.	truffle	sudo npm install -g truffle
2.	metamask	Extension google

1.2 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 déployé.

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


2 Démarrage de pré-production

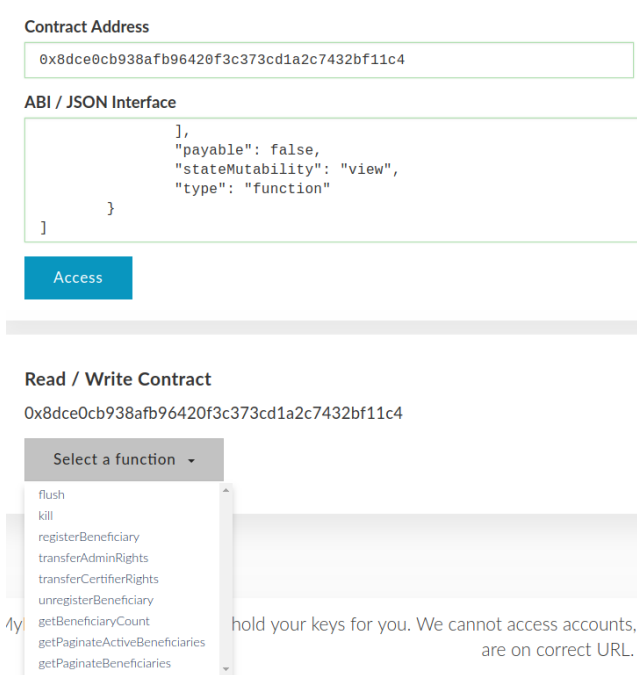
2.1 Démarrage

Décrire le mode opératoire de démarrage initial de l'application

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

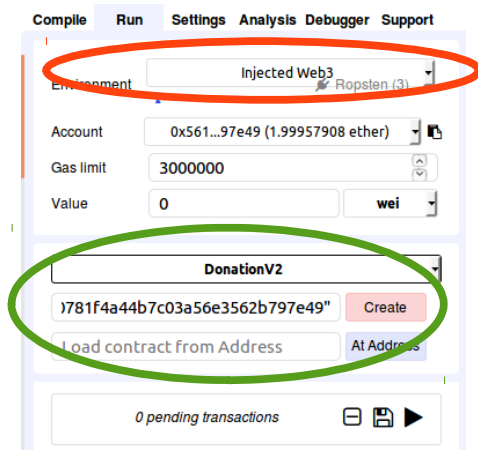
Une fois le contrat déployé sur la blockchain de test (ropsten), le contrat peut être utilisé.

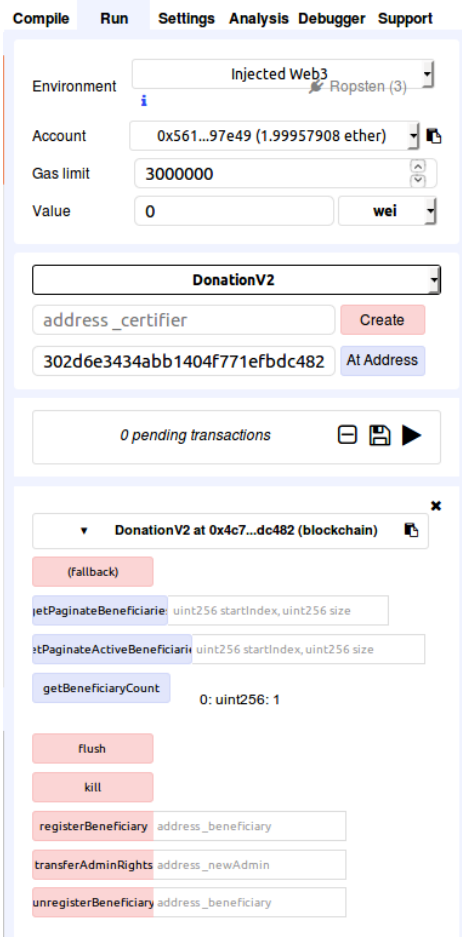
Éta	Action	Illustration
1	<p>Aller sur le site de MyEtherwallet.</p> <p>https://www.myetherwallet.com/</p> <p>Sous l'onglet « Contracts », il est possible d'interagir avec le smart contract.</p> <ul style="list-style-type: none"> Coller l'adresse du contrat déployé (cercle rouge) sous 0x8dce0cb938afb96420f3c373cd1a2c7432bf11c4 Coller l'abi du smart contrat dans la partie ABI/Json interface (cercle vert) <p>Ps : l'abi du contrat déployé :</p> <pre>[{ "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" },</pre>	

Éta	Action	Illustration
	<pre>{ "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" }]</pre> <ul style="list-style-type: none"> • Cliquer bouton « Access » <p>Une fois l'accès à l'interface du smart contrat est effectué. Il suffit de choisir une fonction et l'exécuter.</p>	 <p>The screenshot shows the 'Access' button being clicked, which leads to the 'Read / Write Contract' page. The contract address is 0x8dce0cb938afb96420f3c373cd1a2c7432bf11c4. A dropdown menu lists the available functions: flush, kill, registerBeneficiary, transferAdminRights, transferCertifierRights, unregisterBeneficiary, getBeneficiaryCount, getPaginateActiveBeneficiaries, and getPaginateBeneficiaries.</p>

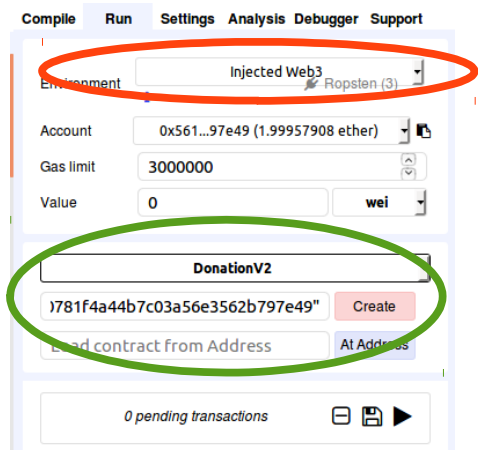
2.3 Test du contrat DonationV2 via remix et metamask (version initiale)

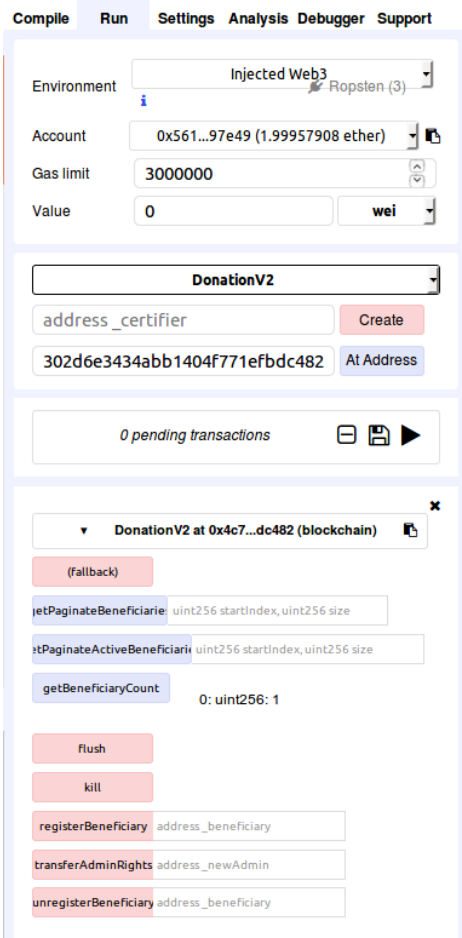
Une fois le contrat déployé sur la blockchain de test (testnet), le contrat peut être utilisé. Le plugin Metamask permet de se connecter à un réseau ethereum (ici ropsten testnet) et propose une interface graphique pour la création d'un compte, et l'accès au réseau (unlock). Le site web Remix (<https://remix.ethereum.org>)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

Éta	Action	Illustration
1	<p>Aller sur le site de Remix IDE.</p> <p>https://remix.ethereum.org</p> <p>Dans l'onglet « Settings », il est possible de préciser la version du compilateur solidity. (4.19 actuellement).</p> <p>Dans l'onglet « Compile », il faut coller le smart contrat.</p> <p>Dans l'onglet « Run », si metamask est installé et le compte ethereum débloqué alors Remix initialise son environnement (cercle rouge) automatiquement avec le réseau choisi par metamask (ici ropsten(3) ... le 3 représente l'identifiant du réseau).</p> <p>Injected web3 signifie que remix injecte les commandes sur le nœud ethereum spécifié (dans notre cas metamask indique ropsten test net).</p> <p>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 précisant l'adresse du certifieur (entre guillemet) et en cliquant sur « create ». (cercle vert)</p>	

Éta	Action	Illustration
2	<p>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.</p> <p>les adresses passées en parametres aux fonction du contrat doivent etre entourée de guillemets.</p>	
3	<p>Toutes les fonctions sont alors testables directement via remix.</p> <p>exemple :</p> <p>pour détruire le smart contract, il suffit de cliquer sur la fonction kill et signer la transaction (via metamask)</p>	

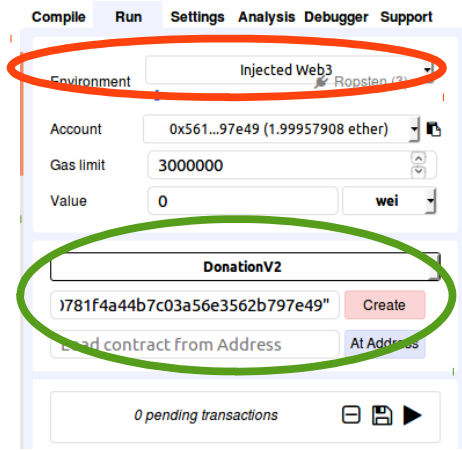
3 Tests et interaction avec le smart contrat déployé (donationV2.sol version initiale)

Éta	Action	Illustration
1	<p>Aller sur le site de Remix IDE.</p> <p>https://remix.ethereum.org</p> <p>Dans l'onglet « Settings », il est possible de préciser la version du compilateur solidity. (4.19 actuellement).</p> <p>Dans l'onglet « Compile », il faut coller le smart contrat.</p> <p>Dans l'onglet « Run », si metamask est installé et le compte ethereum débloqué alors Remix initialise son environnement (cercle rouge) automatiquement avec le réseau choisi par metamask (ici ropsten(3) ... le 3 représente l'identifiant du réseau).</p> <p>Injected web3 signifie que remix injecte les commandes sur le nœud ethereum spécifié (dans notre cas metamask indique ropsten test net).</p> <p>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 précisant l'adresse du certifieur (entre guillemet) et en cliquant sur « create ». (cercle vert)</p>	

Éta	Action	Illustration
2	<p>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.</p> <p>les adresses passées en parametres aux fonction du contrat doivent etre entourée de guillemets.</p>	
3	<p>Pour détruire le smart contract, il suffit de cliquer sur la fonction kill et signer la transaction (via metamask)</p>	

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)

Éta	Action	Illustration
1	<p>Aller sur le site de Remix IDE.</p> <p>https://remix.ethereum.org</p> <p>Dans l'onglet « Settings », il est possible de préciser la version du compilateur solidity. (4.25 actuellement).</p> <p>Dans l'onglet « Compile », il faut coller le smart contrat.</p> <p>Dans l'onglet « Run », si metamask est installé et le compte ethereum débloqué alors Remix initialise son environnement (cercle rouge) automatiquement avec le réseau choisi par metamask (ici ropsten(3) ... le 3 représente l'identifiant du réseau).</p> <p>Injected web3 signifie que remix injecte les commandes sur le nœud ethereum spécifié (dans notre cas metamask indique ropsten test net).</p> <p>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)</p>	
2	<p>Deploy the donation contract</p> <ol style="list-style-type: none"> Une fois les contrats importés sur remix, on clique sur le bouton « deploy » Une fois le contrat est déployé, on récupère l'adresse pour déployer le contrat privilège request <p>=> contrat déployé sous l'adresse "0x4dd5f5d49cb78c1b2b73e6e42fafe23f528bc495"</p> <p>Deploy the privilegeRequest contract</p> <ol style="list-style-type: none"> Sélectionner le contrat « privilegeRequest », on passe comme paramètre l'adresse du contrat donation <p>notre cas "0x4dd5f5d49cb78c1b2b73e6e42fafe23f528bc495"</p> <ol style="list-style-type: none"> Une fois le contrat est déployé, on récupère 	<p>Deploy the donation contract</p> <p>Transaction Hash https://ropsten.etherscan.io/tx/0x9780bb620ad297d68176247d9baf17b955583dcdcd314c0950314258fc9bc375 </p> <p>Deploy the privilegeRequest contract</p>

Éta	Action	Illustration
	<p>l'adresse pour mettre à jour l'adresse du contrat privilège request dans le contrat donation.</p> <p>=> contrat déployé sous "0xba3ca10fd248886890141616fbe3e437ea9f0d40"</p> <p>Mettre à jour l'adresse du privilege request sur le contrat donation</p> <ol style="list-style-type: none"> 1. Faire appel à la fonction setPrivilegeRequestAddress en passant l'adresse du contrat privilegeRequest "0xba3ca10fd248886890141616fbe3e437ea9f0d40" 	<p>Transaction Hash</p> <p>https://ropsten.etherscan.io/tx/0xb75aa29a8ea4a613767d8070a5909811fbc2601517d3afc041ebd59131210c5d</p> <p>Mettre à jour l'adresse privilegeRequest sur le contrat donation</p> <p>Transaction hash</p> <p>https://ropsten.etherscan.io/tx/0xe66e32deb9a9c39472bdd3b9aa9726b24ecf360ead8578a9ada4b89f09d86ff6</p>

5 Tests

5.1 Déroulement du fichier test donation.js

Éta	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)	<p>Truffle test test/donation.js</p> <p>Résultats</p> <pre> Contract: Donation >> Donation deployed at address 0xa3393d99da0fe2de31739796acc0838bcbcc1610 >> Privilege request contract deployed at address 0x53e1f51214bc77758c7f8c06b19b52a13bcb795b >> Privilege request contract set 0x53e1f51214bc77758c7f8c06b19b52a13bcb795b ✓ it check if the donation contract has a balance of 0 ether at first (110ms) >> Donation deployed at address 0xb8924ec4ccff825cc4d478cee07fe67594d32908 >> Privilege request contract deployed at address 0xb73892a36771736789a22102451b483ed2cf6be5 >> Privilege request contract set 0xb73892a36771736789a22102451b483ed2cf6be5 The address of privilege contract of Charles is 0x5c3978cc88f0ad715e5ac41a27414050d0162f02 ✓ it check the creation of a privilege contract to Charles (570ms) >> Donation deployed at address 0x55796019f4fa70f184856c5881a112c5e875b9bd >> Privilege request contract deployed at address 0xae4954fa9d3e3b69d0c23e36dabea1535741219 >> Privilege request contract set 0xae4954fa9d3e3b69d0c23e36dabea1535741219 The address of privilege contract of Alice is 0x217f5a8379cae79cbb66bb635d67908b61aee01f The address of privilege contract of Charles is 0x6baa9fe2ab9c473e445d66c43716c0fdc4c08666 ✓ it simulate the distribution process (3811ms) Try to make a distribute when beneficiaries count == 0 >> Donation deployed at address 0xbf3974ba5a9064fbbf42c0ee914ae0b8ae5505eb >> Privilege request contract deployed at address 0x2f2e8ac8b4b1e28edc3f93c12a21304bde4b7119 >> Privilege request contract set 0x2f2e8ac8b4b1e28edc3f93c12a21304bde4b7119 ✓ it donate 1 ether and try to make a distribute (88ms) Register a beneficiary account and try to make a distribute with a balance == 0 >> Donation deployed at address 0x0b1ed705852e4e873dc1df3ed870cc608fab8d95 >> Privilege request contract deployed at address 0x24c8a6374ef6c45812c17e7ebc3cd6ff51234c28 >> Privilege request contract set 0x24c8a6374ef6c45812c17e7ebc3cd6ff51234c28 ✓ it register a beneficiary (128ms) Try to register a beneficiary account from account != privilege Request contract >> Donation deployed at address 0x302184856f197e61337eaff8c0cfd5bbab380418 >> Privilege request contract deployed at address 0x3f5525cf986c50999dc7b7a070979dd073085f27 >> Privilege request contract set 0x3f5525cf986c50999dc7b7a070979dd073085f27 ✓ it try to register a beneficiary Update privileges from another account >> Donation deployed at address 0x9cc5256edc64506f2cc6250c0876fd1940ac3ef5 >> Privilege request contract deployed at address 0x1c161abd82eb6e34060ab5c49c7df2c7b83fc69c >> Privilege request contract set 0x1c161abd82eb6e34060ab5c49c7df2c7b83fc69c ✓ it register a beneficiary & try to update privileges (160ms) Make a donation >> Donation deployed at address 0xf64ecca7355173f01a5627ac530b23c2db4c150 >> Privilege request contract deployed at address 0x3fa33c535c095f1596590b0d9c88387a17ba4c4d >> Privilege request contract set 0x3fa33c535c095f1596590b0d9c88387a17ba4c4d ✓ donate 1000000000000000000 wei (172ms) Try to make a distribution twice in one hour >> Donation deployed at address 0x6acff8484752fbb92c1babb807a828ae33183a8c >> Privilege request contract deployed at address 0x0efb4998ea18262209f4745a1216d40df0f42013 >> Privilege request contract set 0x0efb4998ea18262209f4745a1216d40df0f42013 ✓ it register, donate and make two distribution (345ms) Try to make a first distribution and another one after 1h >> Donation deployed at address 0x8e1334a9e51fe7371fb8c150f5bcbf622e4d7d2c >> Privilege request contract deployed at address 0xed0124adf03028339c1eedc34623404fba46f336 >> Privilege request contract set 0xed0124adf03028339c1eedc34623404fba46f336 ✓ it register, donate and make two distribution (691ms) 10 passing (9s) </pre>

5.2 Main scénario sur ropsten

Adresse contrat donation 0x4dd5f5d49cb78c1b2b73e6e42fafa23f528bc495

- 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>
- **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/0x90d9e7a76b4a84c8b7bac9b1d3fdcf0d141b708ac4ffa9da4342503ecfbd7869>
- **Résultat de la transaction**
 - Le contrat de privilège d'Alice transfert 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