Créer un compte [https://labs.play-with-docker.com](https://labs.play-with-docker.com/)

Créer un compte <https://infura.io/> et créer un projet dans infura.io

1. Open a terminal on [https://labs.play-with-docker.com](https://labs.play-with-docker.com/).
2. Run : $ docker container run -it ubuntu
3. Run: # apt update
4. # apt-get install -y nano virtualenv python-pip python3-pip python3-venv python3.6-dev
5. Run the following to configure the python virtual env (see here for more details about venv: [https://web3py.readthedocs.io/en/stable/troubleshooting.html#setup-environment](https://web3py.readthedocs.io/en/stable/troubleshooting.html" \l "setup-environment))
6. root@045efa6dd3b1:/# pip install virtualenv
7. root@045efa6dd3b1:/# virtualenv venv
8. root@045efa6dd3b1:/# virtualenv -p /usr/bin/python3.6 venv
9. root@045efa6dd3b1:/# source venv/bin/activate
10. (venv) root@045efa6dd3b1:/# pip install web3

Ou bien

# python3 -m venv venv

# . venv/bin/activate

# pip install web3

Tester l’API Web3py dans un terminal

1. Run : # python
2. **>>> from** **web3** **import** Web3, HTTPProvider
3. **>>>** web3 = Web3(HTTPProvider('http://localhost:8545'))
4. **>>>** web3
5. <web3.main.Web3 object at 0x7f7c38b0dcf8>
6. **>>>** Web3.toWei(1, 'ether')
7. 1000000000000000000
8. **>>>** web3.fromWei(1000000000000000000, 'ether')
9. Decimal('1')
10. **>>> from** **web3.auto** **import** w3
11. **>>>** existing\_filter = web3.eth.filter(filter\_id="0x0")
12. **>>> existing\_filter**
13. **<web3.utils.filters.LogFilter object at 0x7f7c371e8048>**

**Tutorial**

**>>> infura\_url = 'https://mainnet.infura.io/v3/00919abdfabf4cecae37fe653ae1a9a1'**

**>>> web3 = Web3(Web3.HTTPProvider(infura\_url))**

**>>> web3.isConnected()**

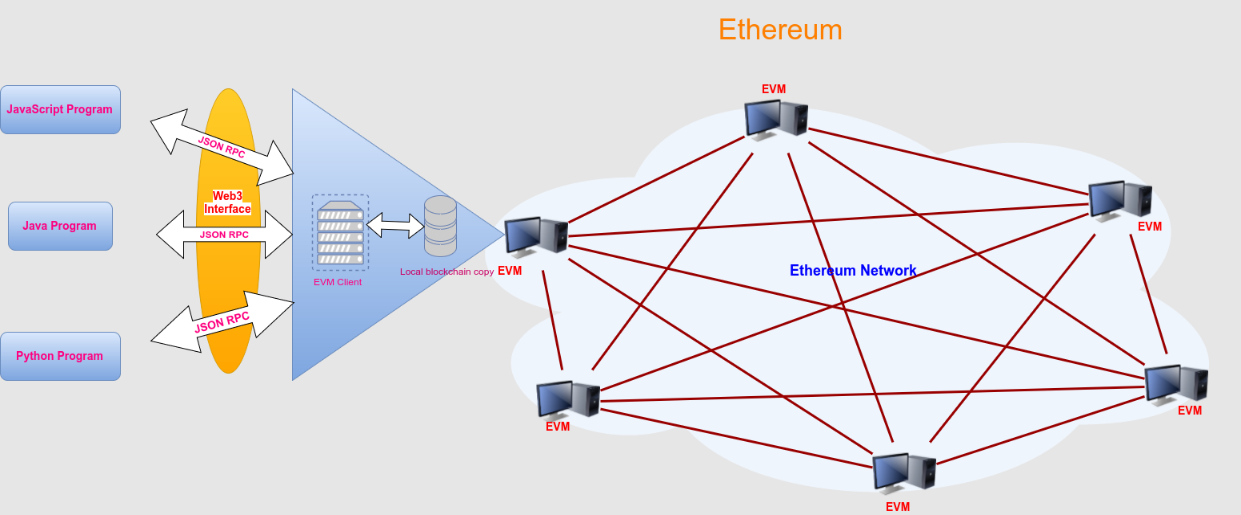
**True**

**>>> web3.eth.blockNumber**

**7578773**

**>>> web3.eth.blockNumber**

**7578773**



Tutorial 1 : On va écrire un script python pour interagir avec Ethereum blockchain (**Run : $ python app.py**)

**from web3 import Web3**

**# the url is the link to my project in my profile on infura**

**# this need you to create a url to your own project on infura**

**infura\_url\_ropsten = 'https://ropsten.infura.io/v3/00919abdfabf4cecae37fe653ae1a9a1'**

**web3 = Web3(Web3.HTTPProvider(infura\_url\_ropsten))**

**web3.isConnected()**

**print(web3.eth.blockNumber)**

**# account 1 is my own account on metamask on ropsten**

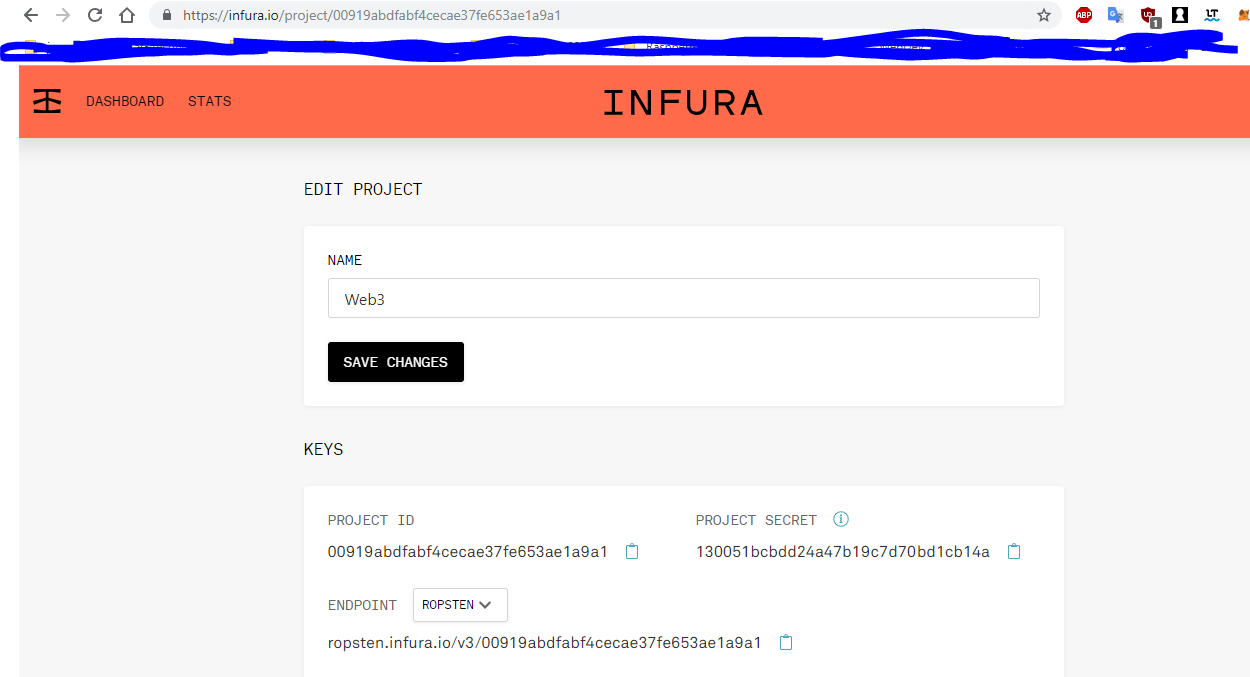
**account1 = "0x80e9B8f891c326bc8c25dA59672DC5F363C4bC9A"**

**balance = web3.eth.getBalance(account1)**

**print(balance)**

**web3.fromWei(balance,'ether')**

**print(web3.fromWei(balance,'ether'))**



Tutorial 2 : Écrire un script python pour envoyer de la cryptomonnaie dans la blockchain avec Web3.py

**Ouvrir** [**https://etherscan.io**](https://etherscan.io) **et chercher le jeton (Token) OmiseGO dans le lien suivant** <https://etherscan.io/token/0xd26114cd6EE289AccF82350c8d8487fedB8A0C07>

Vous pouvez remarquer le smart contract de ce Token, cliquer sur l’adresse et aller jusqu’au code du smart contract :

Script Python

**from web3 import Web3**

**import json**

**# the url is the link to my project in my profile on infura**

**# this need you to create a url to your own project on infura**

**infura\_url\_mainnet = 'https://mainnet.infura.io/v3/00919abdfabf4cecae37fe653ae1a9a1'**

**web3 = Web3(Web3.HTTPProvider(infura\_url\_mainnet))**

**print(web3.isConnected()) # you should see true**

**# ABI of the smart contract**

**abi = json.loads('[{"constant":true,"inputs":[],"name":"mintingFinished","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"name","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_spender","type":"address"},{"name":"\_value","type":"uint256"}],"name":"approve","outputs":[],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"totalSupply","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_from","type":"address"},{"name":"\_to","type":"address"},{"name":"\_value","type":"uint256"}],"name":"transferFrom","outputs":[],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"decimals","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"},{"constant":false,"inputs":[],"name":"unpause","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_to","type":"address"},{"name":"\_amount","type":"uint256"}],"name":"mint","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"paused","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[{"name":"\_owner","type":"address"}],"name":"balanceOf","outputs":[{"name":"balance","type":"uint256"}],"payable":false,"type":"function"},{"constant":false,"inputs":[],"name":"finishMinting","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":false,"inputs":[],"name":"pause","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"owner","outputs":[{"name":"","type":"address"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"symbol","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_to","type":"address"},{"name":"\_value","type":"uint256"}],"name":"transfer","outputs":[],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_to","type":"address"},{"name":"\_amount","type":"uint256"},{"name":"\_releaseTime","type":"uint256"}],"name":"mintTimelocked","outputs":[{"name":"","type":"address"}],"payable":false,"type":"function"},{"constant":true,"inputs":[{"name":"\_owner","type":"address"},{"name":"\_spender","type":"address"}],"name":"allowance","outputs":[{"name":"remaining","type":"uint256"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"newOwner","type":"address"}],"name":"transferOwnership","outputs":[],"payable":false,"type":"function"},{"anonymous":false,"inputs":[{"indexed":true,"name":"to","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"name":"Mint","type":"event"},{"anonymous":false,"inputs":[],"name":"MintFinished","type":"event"},{"anonymous":false,"inputs":[],"name":"Pause","type":"event"},{"anonymous":false,"inputs":[],"name":"Unpause","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"name":"owner","type":"address"},{"indexed":true,"name":"spender","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"name":"Approval","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"name":"from","type":"address"},{"indexed":true,"name":"to","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"name":"Transfer","type":"event"}]')**

**#address of the smart contract**

**address = "0xd26114cd6EE289AccF82350c8d8487fedB8A0C07"**

**contract = web3.eth.contract(address=address,abi=abi)**

**print(contract)**

**# interagir avec le smart contract**

**totalSupply = contract.functions.totalSupply().call()**

**print(contract.functions.totalSupply().call())**

**# print the totalSupply (th same)**

**print(totalSupply)**

**print(web3.fromWei(totalSupply,'ether'))**

**# call functions**

**print(contract.functions.name().call())**

**print(contract.functions.symbol().call())**

**holderAddress = '0x2551d2357c8da54b7d330917e0e769d33f1f5b93'**

**valid\_address = web3.toChecksumAddress(holderAddress)**

**balance = contract.functions.balanceOf(valid\_address).call()**

**print(web3.fromWei(balance,'ether'))**

Resultat de l’execution :

**(venv) root@045efa6dd3b1:/# nano app2.py**

**(venv) root@045efa6dd3b1:/# python app2.py**

**True**

**<web3.utils.datatypes.Contract object at 0x7f0621b19b00>**

**140245398245132780789239631**

**140245398.245132780789239631**

**OMGToken**

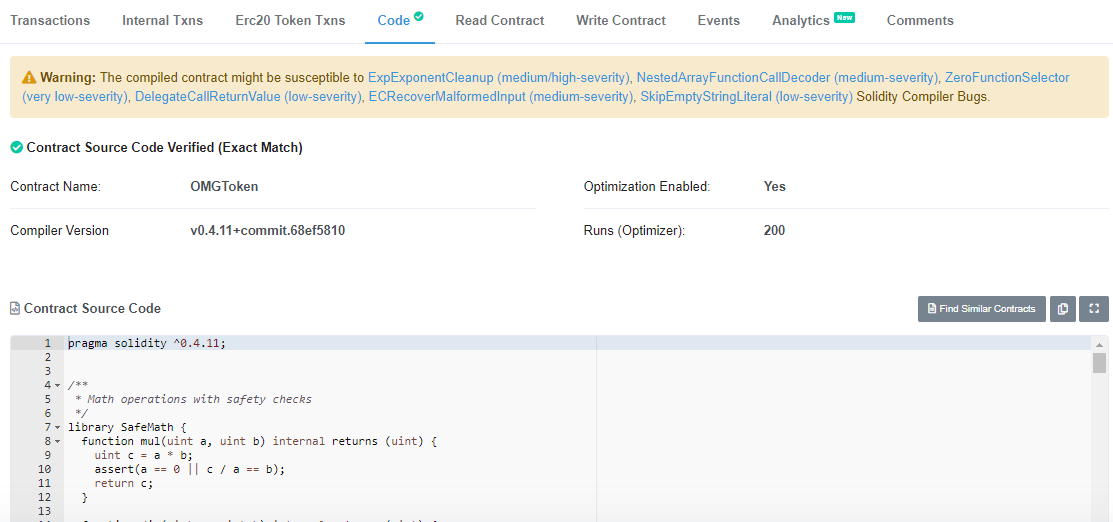
**OMG**

**26770491.168525792348307853**

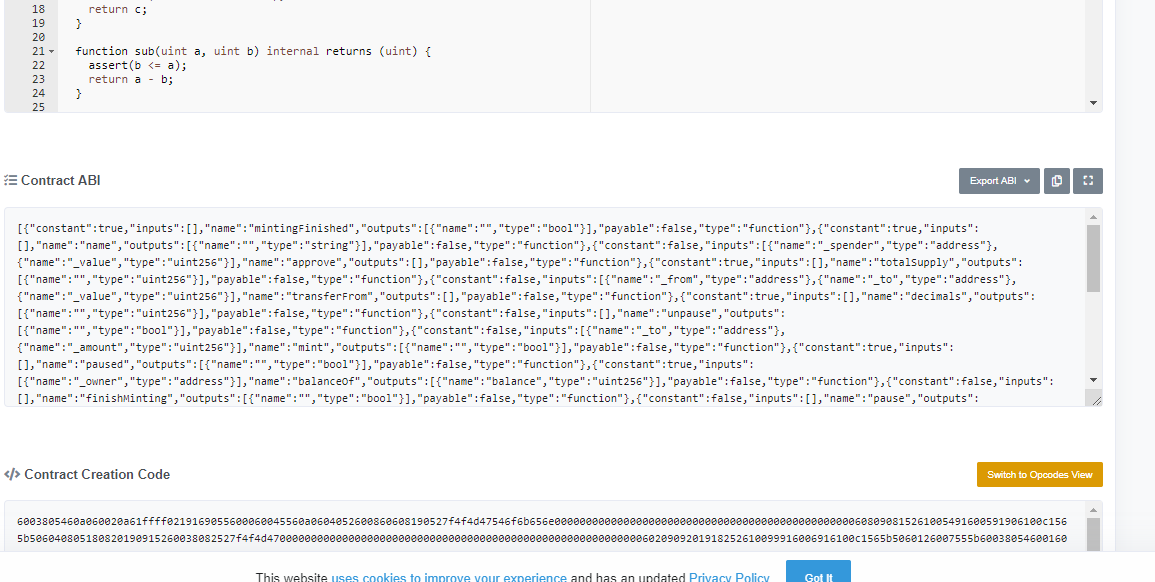
**(venv) root@045efa6dd3b1:/#**



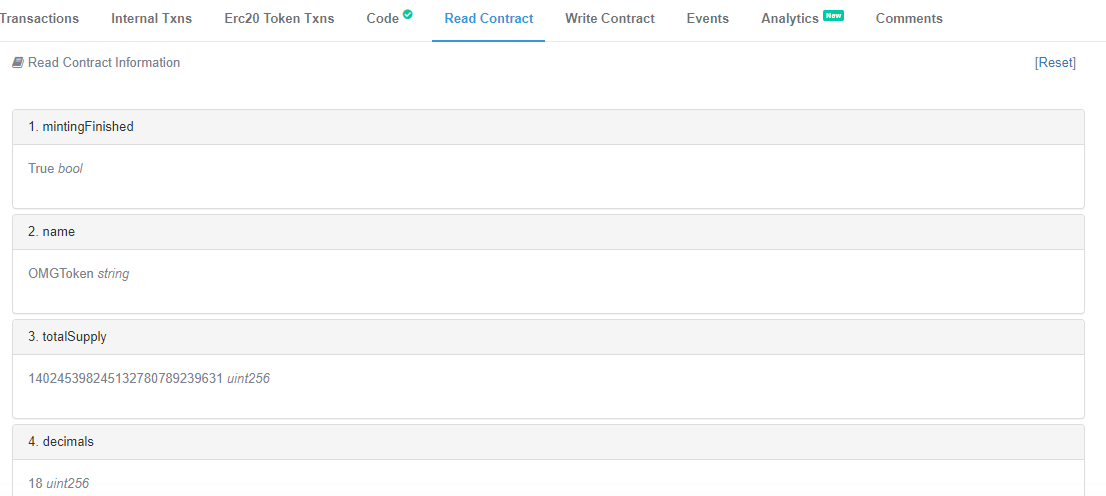
**Adresse du smart contract à ajouter à votre script**



**Smart contract : code source disponible sur le site etherscan**

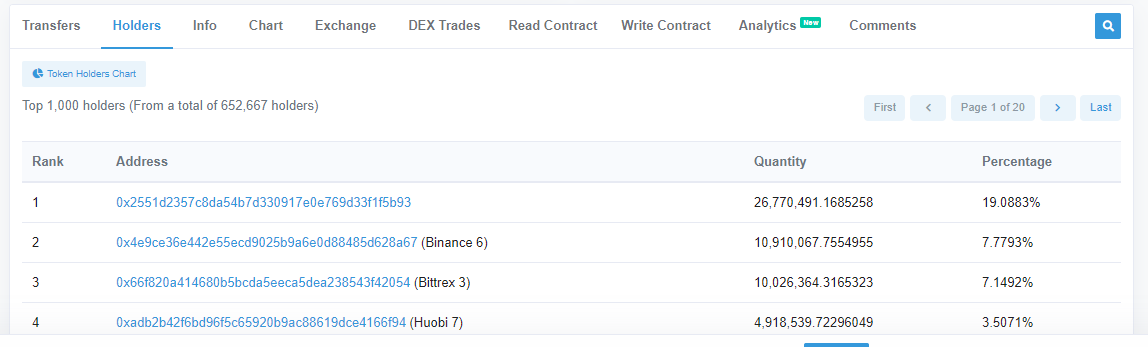


**ABI du smart contract à copier et ajouter au script**





**Liste des fonctions du smart contract disponible, à utiliser dans le script python**



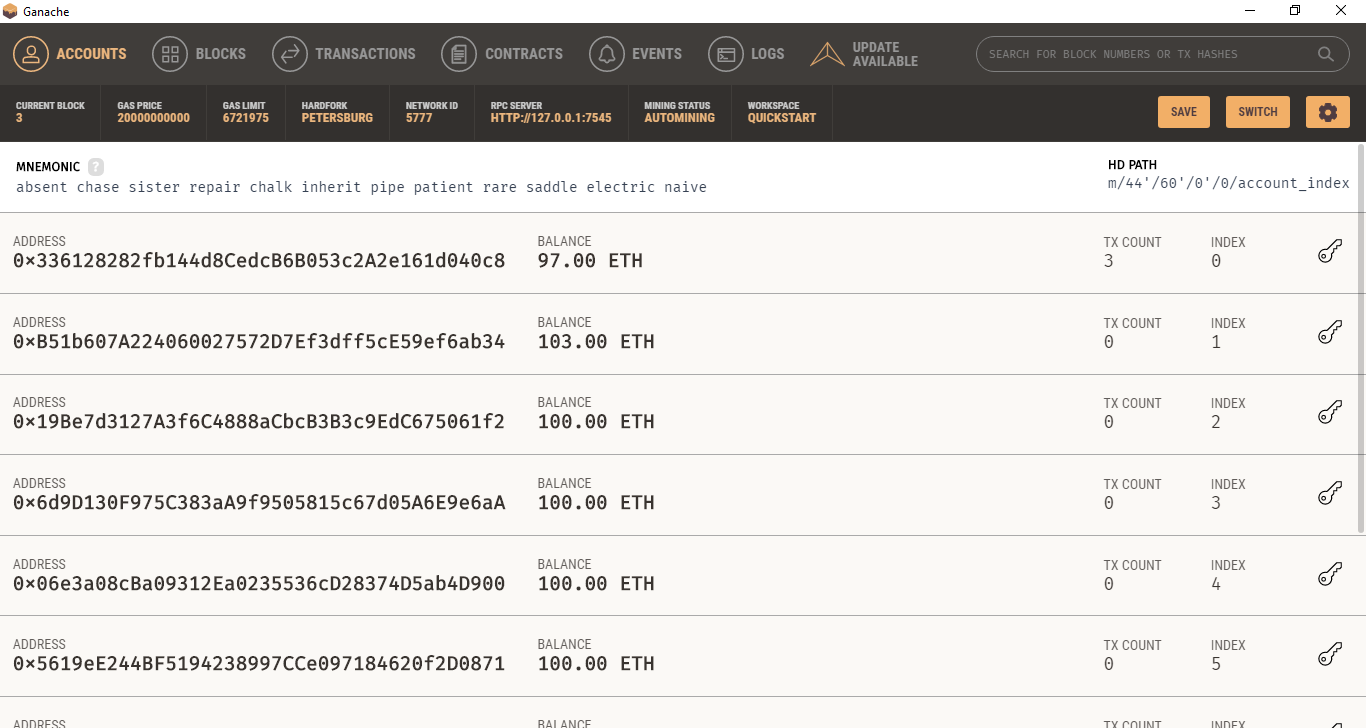
**Liste des utilisateurs (holders) qui utilisent le Token, on souhaite afficher leurs balances dans le script (méthode balanceOf(address du holder))**

Tutorial 3 : Manipulation de smart contract dans une blockchain locale personnelle avec ganache

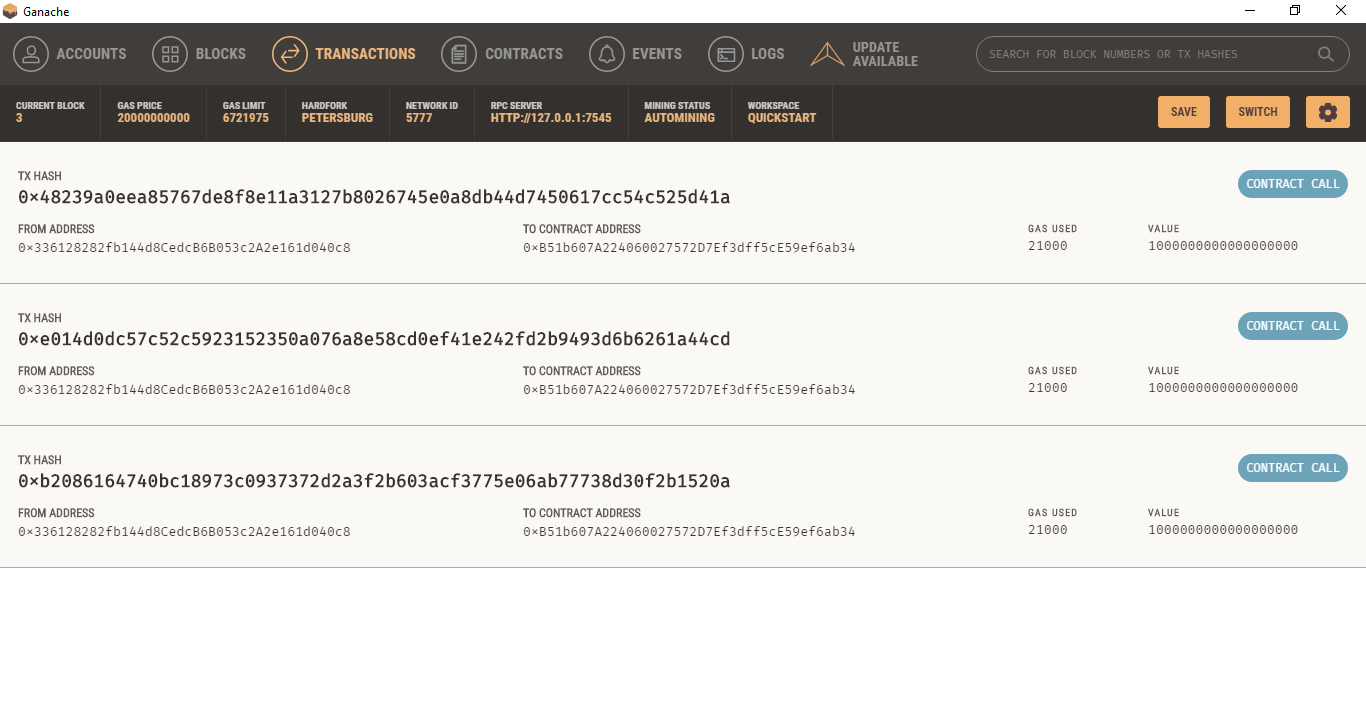
Télécharger et installer ganache : <https://truffleframework.com/ganache>

Sous Windows : Télécharger VSCode et Visual Studio 2019 ou 2017 (avec C++ build tools, python dev tools) et activer virtualenv (c:\>pip install virtualen && virtualenv venv)

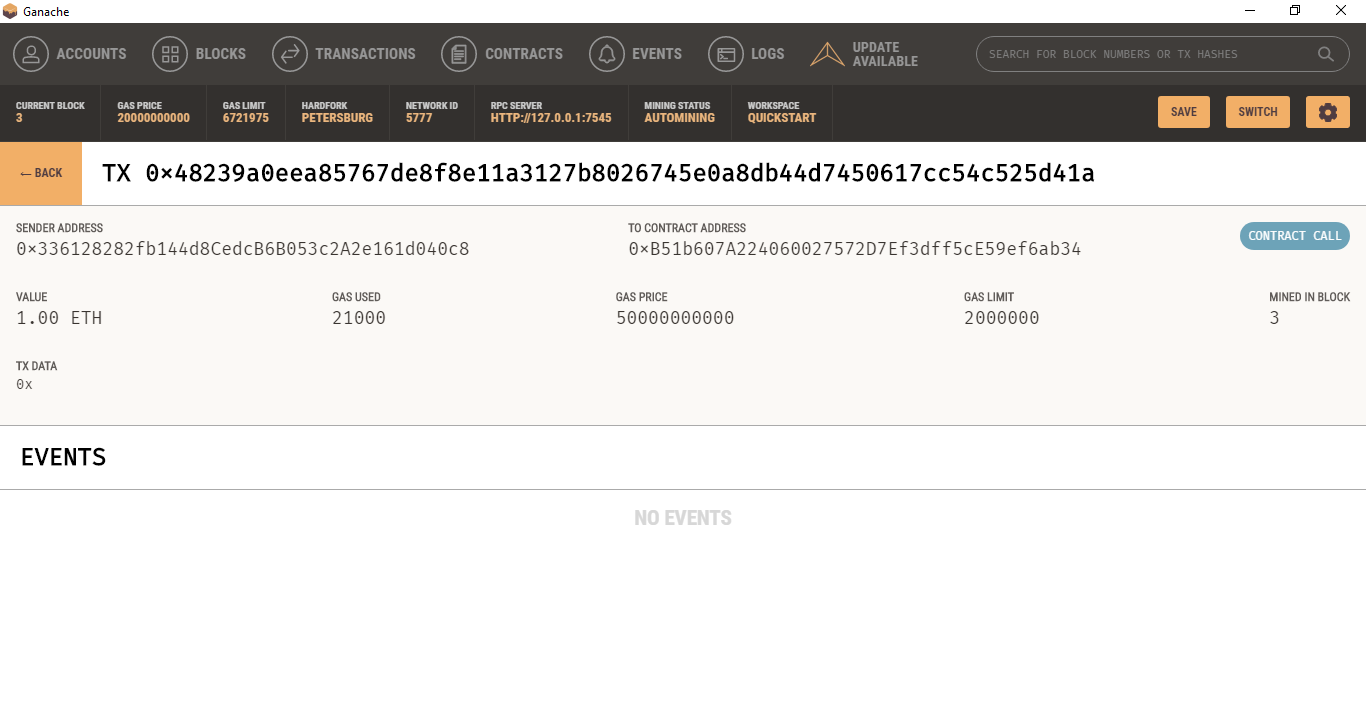
Sous Linux : telecharger les packages python en dessus et activer virtualenv



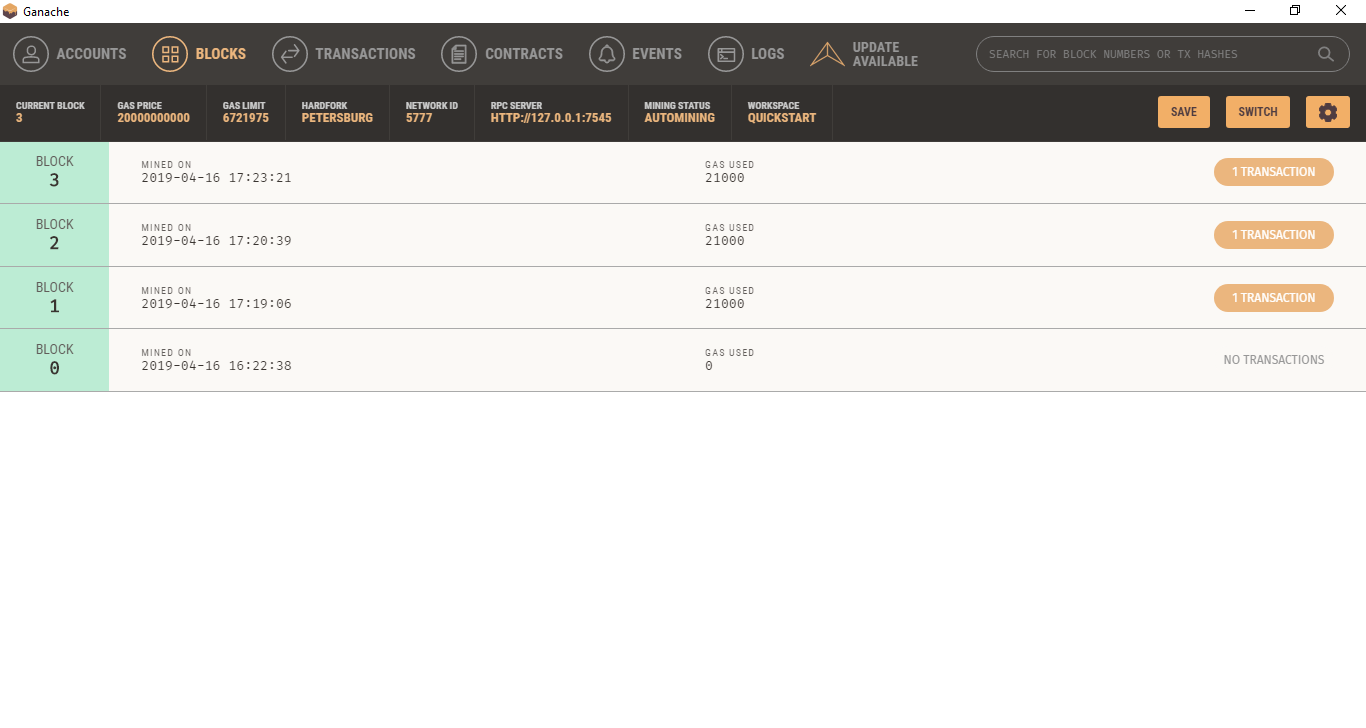
Account 1 send 1 Ether (3 times) to Account 2



Three (03) transactions generated



Contain of the last transaction



See all the blocks (04 blocks) used to send the transactions