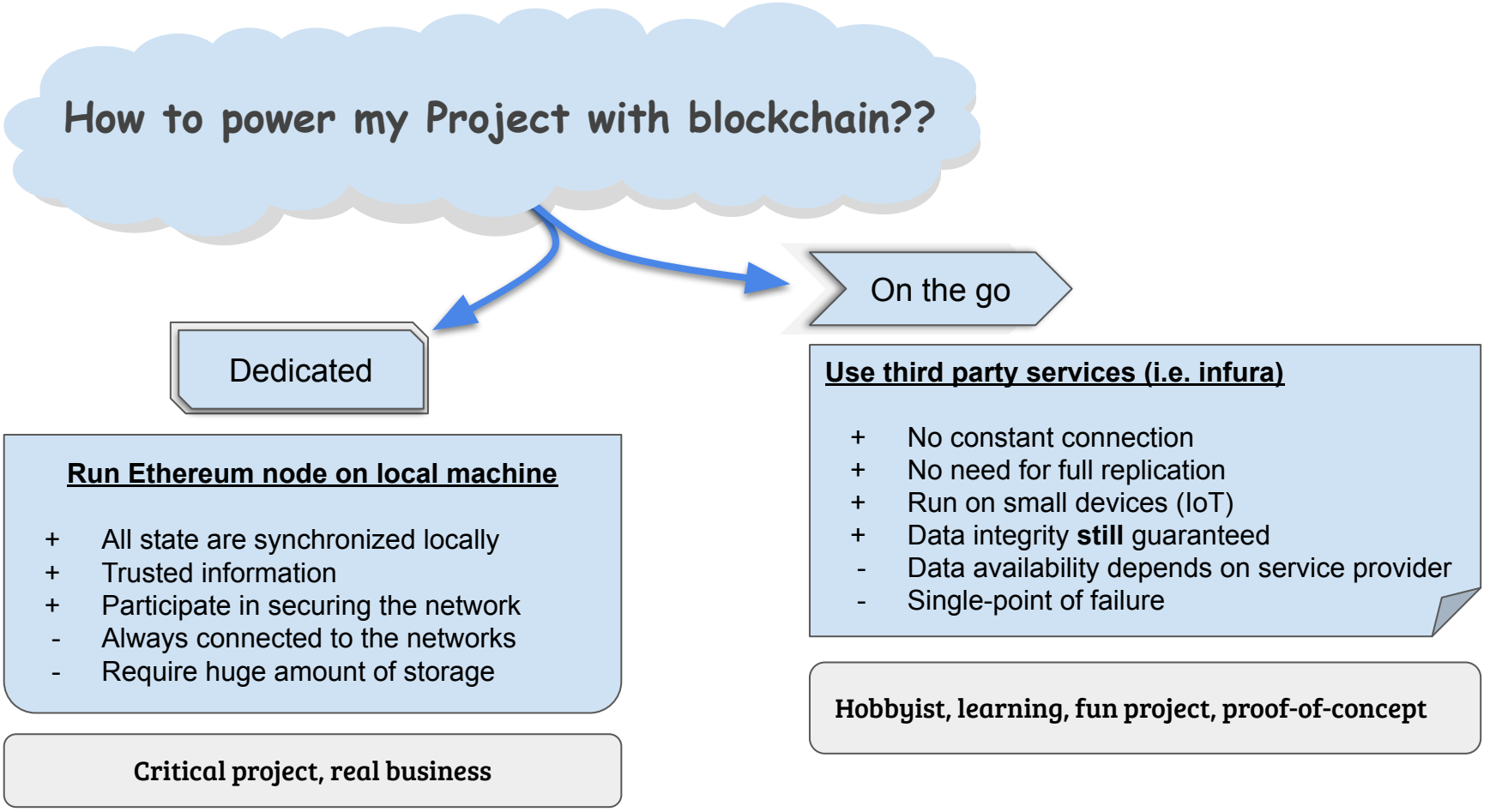
Blockchain
Client-side Programming


How to power my Project with blockchain??
On the go
Dedicated
Use third party services (i.e. infura)
+ No constant connection
+ No need for full replication
+ Run on small devices (IoT)
+ Data integrity still guaranteed
- Data availability depends on service provider
- Single-point of failure
Hobbyist, learning, fun project, proof-of-concept
Run Ethereum node on local machine
+ All state are synchronized locally
+ Trusted information
+ Participate in securing the network
- Always connected to the networks
- Require huge amount of storage
Critical project, real business

Outline

1. Setup & Testing
2. Simple Transaction
3. Smart contracts Transaction

Source code: `https://github.com/acbari/ethereum-introduction`

1. Setup & Testing

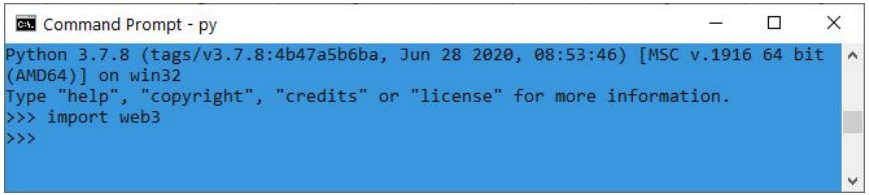
- Instalasi
- Environment setup
- Test connection
- Wallet

Instalasi Web3

Web3

- Install web3
  - python -m pip install web3
- Test:
  - python<enter>
  - import web3

- Test program: import web3

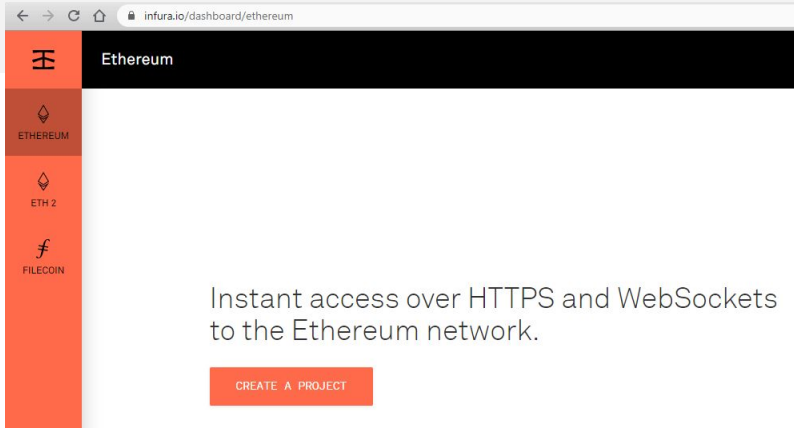


Sesuaikan perintah python dengan versi python yang terinstall di komputer, contoh:

- python 2.7 → python
- Python 3.8 → python3

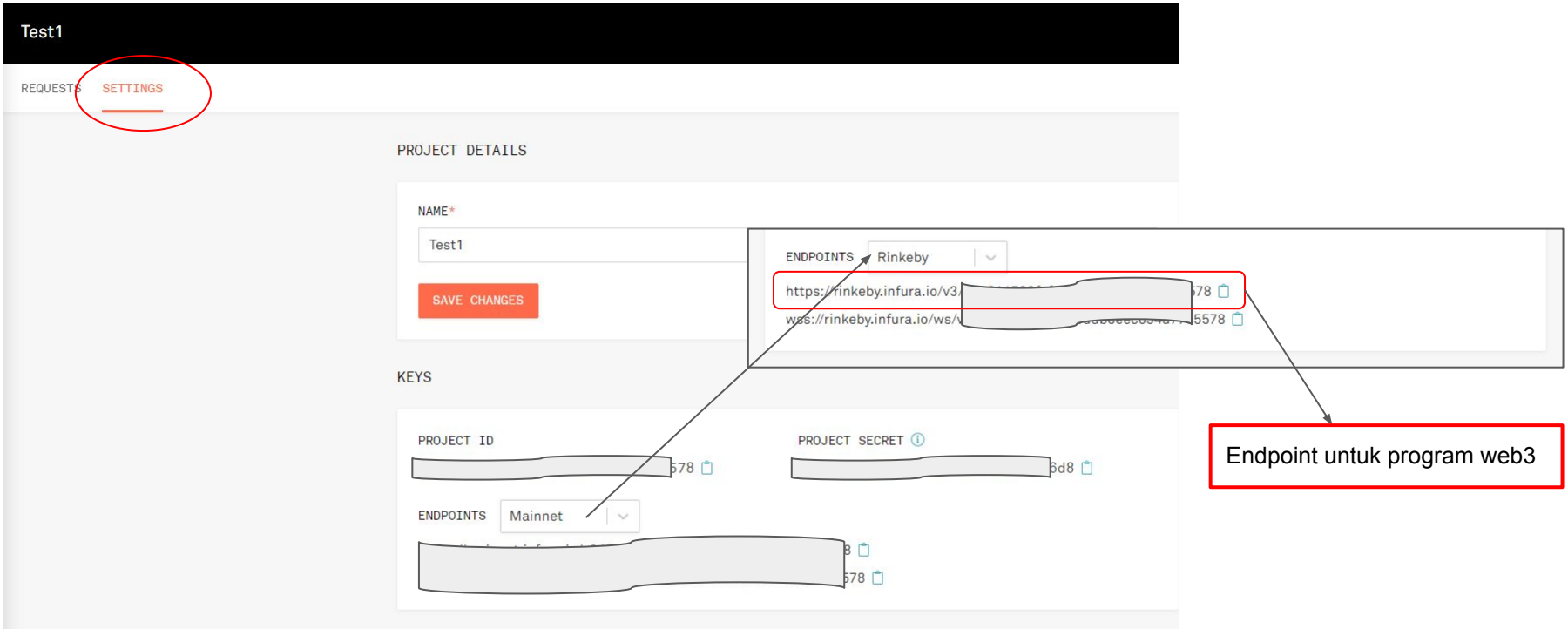
Setup Infura Account

- Create a new account
  - https://infura.io/register
- Check email, confirm email address
  - Redirect: https://infura.io/dashboard



- Buat proyek ethereum baru:
  - Pilih menu ETHEREUM, → Create A Project
  - Buat nama untuk proyek, misal: Test1
- Buka menu setting untuk project Test1 dan copy infura-project-id sebagai token untuk program web3

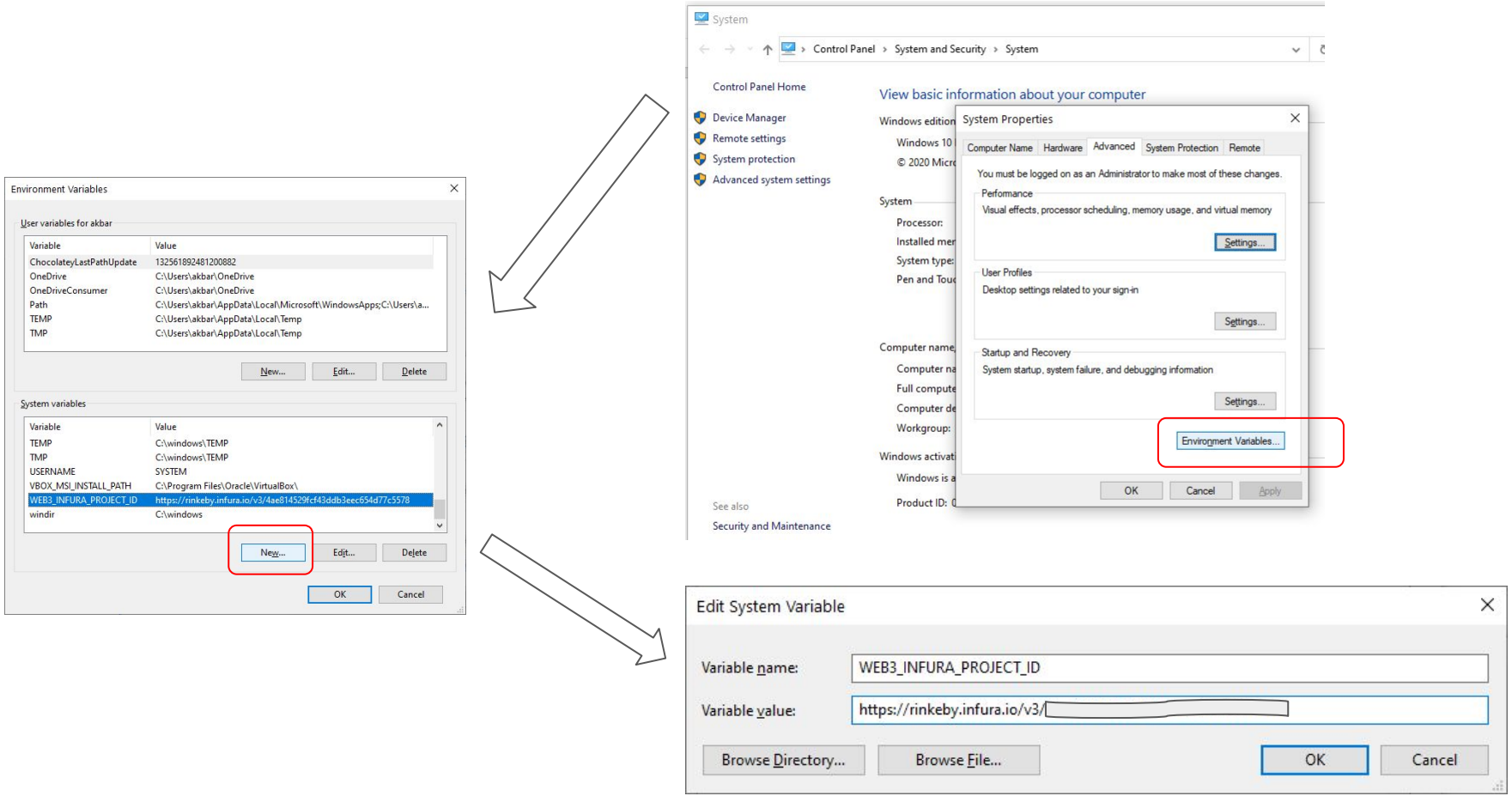
Pastikan jenis jaringan ethereum (endpoint) adalah Rinkeby



Environment setup

Atur Project ID Infura kedalam system environment variable

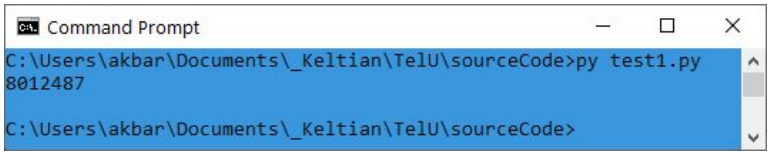
- Linux:
  - From cli:
  - $ export WEB3_INFURA_PROJECT_ID=<yout infura project id>
- Windows
  - Buka environment variable window
  - Buat system variables baru
    - Variable name: WEB3_INFURA_PROJECT_ID
    - Variable value: <yout infura project id>
  - Reset/buka ulang jendela terminal, sebelum menjalankan program testing/hello world


Environment Variables
User variables for akbar
Variable
Value
ChocolateyLastPathUpdate
132561892481200882
OneDrive
C:\Users\akbar\OneDrive
OneDriveConsumer
C:\Users\akbar\OneDrive
Path
C:\Users\akbar\AppData\Local\Microsoft\WindowsApps;C:\Users\a...
TEMP
C:\Users\akbar\AppData\Local\Temp
TMP
C:\Users\akbar\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
TEMP
C:\windows\TEMP
TMP
C:\windows\TEMP
USERNAME
SYSTEM
VBOX_MSI_INSTALL_PATH
C:\Program Files\Oracle\VirtualBox\
WEB3_INFURA_PROJECT_ID
https://rinkeby.infura.io/v3/4ae814529fcf43ddb3eec654d77c5578
windir
C:\windows
New...
Edit...
Delete
OK
Cancel
System
Control Panel
System and Security
System
Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings
View basic information about your computer
Windows edition
Windows 10
© 2020 Micro
System
Processor:
Installed mer
System type:
Pen and Touc
Computer name
Computer na
Full compute
Computer de
Workgroup:
Windows activat
Windows is a
Product ID:
See also
Security and Maintenance
System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply
Edit System Variable
Variable name:
WEB3_INFURA_PROJECT_ID
Variable value:
https://rinkeby.infura.io/v3/
Browse Directory...
Browse File...
OK
Cancel

Test connection: the latest block number

Test1.py: Mengambil informasi nomor blok terakhir yang dibuat (blok terkini)

```
from web3 import Web3
from web3.auto.infura import w3

w3 = Web3(Web3.HTTPProvider('https://rinkeby.infura.io/v3/4ae814529fcf43ddb3eec654d77c5578'))
LastBlock = w3.eth.blockNumber
print(LastBlock )
```



Block terkini: 8012487 (> 8 juta blok)

Test connection: block information

```
...
print(LastBlock)

from web3.middleware import geth_poa_middleware
w3.middleware_onion.inject(geth_poa_middleware,
layer=0)

BlockInfo = w3.eth.get_block(LastBlock)
print(BlockInfo)
print(BlockInfo['difficulty'])
print(BlockInfo['timestamp'])
print(BlockInfo['totalDifficulty'])
```

Khusus PoA (Rinkeby)

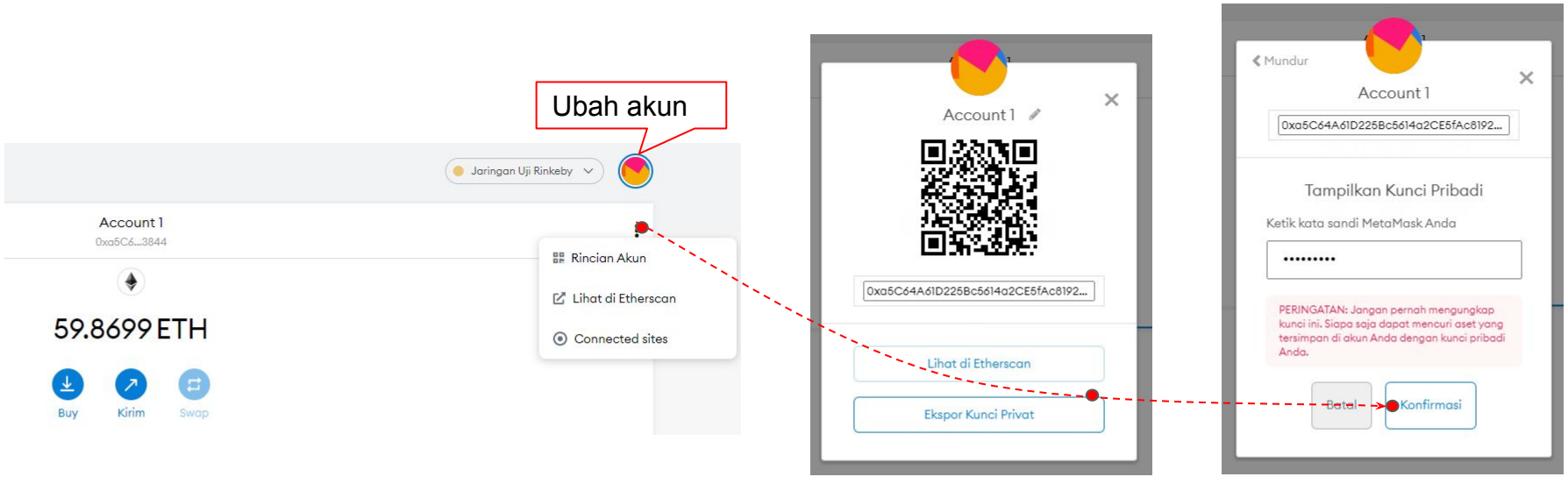
```
C:\Users\akbar\Documents\_Keltian\TelU\sourceCode>py test1.py
8012736
AttributeDict({'difficulty': 1, 'proofOfAuthorityData': HexBytes('0xd883010917846765746888676f312e31352e32856c696e7578000000000000000009f8de74f882b2a6e780d42c67f1c729c2b8de0c5ece
26f424151923abe0791c1151b1a0300c8cd746a1f0befe413e6af54dc88c83d101a4b33928971470cfb00'), 'gasLimit': 10000000, 'gasUsed': 1957881, 'hash': HexBytes('0xd6b9fdbaabbf0aea31033ba7
b762b5617c681b6343c8f672510a58afd817f83e'), 'logsBloom': HexBytes('0x000010000000001a000000000000000100100000401040000000000480002000000000400000180800090000001000c004000004
2420000000000020008010100080002000000000008002000004000000104000080000000100000000000400200000262000000000800400000004800010024000101000008000000000000000000000000200
1000000004800000000010000000000000000200002810000022002100000008000080040010010000000000200080001400010000600000000040000000000000800401000008001000200100008002000041000400060
4010202008000020007000000080000000000000000100810001'), 'miner': '0x0000000000000000000000000000000000000000', 'mixHash': HexBytes('0x0000000000000000000000000000000000000000000
0000000000000000000000'), 'nonce': HexBytes('0x0000000000000000'), 'number': 8012736, 'parentHash': HexBytes('0x60abe1a86392562c744872cb14a1c249b030e11b66411e8a488bafbd506cb98
3'), 'receiptsRoot': HexBytes('0x8ff48cd1fd26990de86b7ad0b07e9a1547009fb9828b483eb79c015e476deb9c'), 'sha3Uncles': HexBytes('0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f
0a142fd40d49347'), 'size': 5367, 'stateRoot': HexBytes('0x4fa319b39e4b7e83a07ec5db5d634ee16bf18ba17907ca1fc914ff12ba055621'), 'timestamp': 1612419329, 'totalDifficulty': 139515
47, 'transactions': [HexBytes('0x34cf742d1bef21e790cf53f871ec63a33a0ffdf1869b04c853638c5be3d00a8e'), HexBytes('0xa4c2d23faf74d73979fcaba4f8954b4c2dc153c12a531db3973b547cbf4b10
0'), HexBytes('0x2ac3af48e14d5a4e3f601eca16ed98aece706f86463c658770e5a4ea6cd5bd9f'), HexBytes('0xcfbeaa4e5e9c00bacadbddac2d8a6ad7a67db7cf1023a03e8674d9b2e58f3e2c'), HexBytes('0
x82cab17d0b50e6e6fadbddee9c5541906211724121ed936c088a5d865e0c72d9'), HexBytes('0x54b33d447bd0ff54afea5a57879179faca7fc16aebdfeea175e4240ff975e653'), HexBytes('0xd89bffebd86cce0
e76c24c0418410ff4a5e636c8c3013e1a188b63ac24a645aa'), HexBytes('0x1f3782897972e45b6c0ddcd428b7e29e7eb30c6236a26fc61bc80446446de132'), HexBytes('0xdbbb7f1db29fb202fd2612d0d520aa9
f5f154884d2d94da96c00d281ad06d554'), HexBytes('0xd400ebd5399e12d478e6af2e24485dfa765c20df4071a2d80f057e5c9b47375e'), HexBytes('0x0e8d03f2dc7fc2eca0b3c8334ba9a32a195376cf883a89f
d3c80b5d1ae0977c2'), HexBytes('0xff8d2cf55bf4f2758e253a533708e2bdea2641fcdfc13b7b15c3d1341bb24e32'), HexBytes('0xd4e7fc07da3c687e6f7c7e1fcfc7825a944e8b90f5edd5a16a39cc12ba861f3
c'), HexBytes('0x11ea08e61b2f44a0d560f1abbb93fd7f2ee96dcb293a07befbf5fc336b81607c'), HexBytes('0x9cc991293ee6645e5958d469c69823a1778f5113e303d1d743fabea48e1f7ea2'), HexBytes('0
xa95859c4baa62ce7815c0d733106d18363e4f02aed237cd9742ddcc7a6597177'), HexBytes('0xa482a8f8feea6556d3b09b0eb23098c8e17985b368b315a6c7574e6750512b9e')], 'transactionsRoot': HexByt
es('0x0dc3753cb8abffb1f992a2d92782a2b9740c59c979ab37f162d96945fb4e3be9'), 'uncles': []})
1
1612419329
13951547

C:\Users\akbar\Documents\_Keltian\TelU\sourceCode>
```

Wallet

- Semua transaksi di blockchain harus memiliki identitas yang valid
- Identitas → akun blockchain
- Setiap transaksi harus memiliki digital signature yang menandakan siapa(akun mana) yang membuat transaksi tersebut
  - Transaksi pembayaran → Update nilai ether dari akun pengirim dan penerima
- Digital signature dihasilkan menggunakan private key dari akun blockchain
- Cara mendapatkan kunci private:
  - Membuat akun baru (membuat wallet sendiri)
  - Mengimport dari akun (contoh: wallet metamask)

Import private key dari metamask

- Buka jendela metamask di web browser
- Pilih rincian akun → eksport kunci private
- Masukkan password metamask
- Salin kunci private yang tayang ke dalam kode program



Buat wallet sendiri

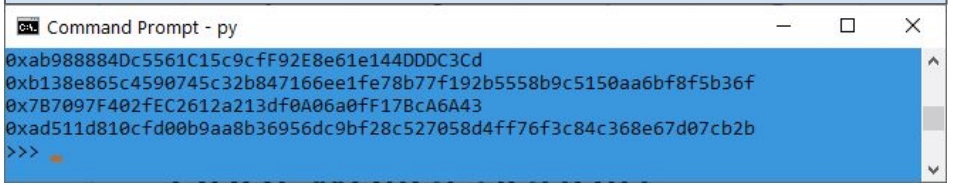
account.py → **`eth_acount`** library

- Buat akun: createAccount(seed) / createAccountAuto()
- Private key: <account>.key.hex()

account.py

```
createAccount()
createAccountAuto()
loadAccount()
encryptAccount()
decryptAccount()
signMessage()
signTransaction()
```

```
from account import createAccount, createAccountAuto

acc1 = createAccount( <random string> )
print(acc1.address)
print(acc1.key.hex())

acc2, mnemonic = createAccountAuto()
print(acc2.address)
print(acc2.key.hex())
```



2. Simple Transaction

- Transfer transaction
- Zero-payment transaction

Transaksi transfer ether

- Import module
- Pilih koneksi third party memakai server infura dengan menyertakan project ID

```
from web3 import Web3
from web3.auto.infura import w3
from web3.middleware import geth_poa_middleware

w3 =
Web3(Web3.HTTPProvider(' https://rinkeby.infura.io/v3/4ae814529fcf43d
db3eec654d77c5578 '))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)
```

```
myAdr = '0xa5C64A61D225Bc5614a2CE5fAc81926438e93844'
private_key = < private_key >
nonce = w3.eth.getTransactionCount(myAdr)

transaction = {
    'from': myAdr,
    'to': '0x1C49AA3FEAb57bc27Ad79F0De64862786766F611',
    'value': 1000000000,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}


signed_tx = w3.eth.account.signTransaction(transaction, private_key)
print("Sending transaction...")
txn_hash = w3.eth.sendRawTransaction(signed_tx.rawTransaction)
print(txn_hash)
txn_receipt = w3.eth.waitForTransactionReceipt(txn_hash)
print(txn_receipt)
```

Susun transaksi yang akan dikirim

- Import akun → private key
- Nonce:
  - nomor transaksi terakhir yang dibuat oleh akun
  - berfungsi untuk mencegah replay attack
- Susun struktur transaksi:
  - From: pengirim
  - To: penerima
  - Value: jumlah ether yang ditransfer
    - 1 Gwei
  - Gas & gas price: nilai standar
  - Nonce: isi dengan nonce diatas
  - chainId: Id jaringan,
    - Mainnet: 1
    - Rinkeby: 4

```
from web3 import Web3
from web3.auto.infura import w3
from web3.middleware import geth_poa_middleware

w3 =
Web3(Web3.HTTPProvider('https://rinkeby.infura.io/v3/4ae814529fcf43d
db3eec654d77c5578'))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)
```

```
myAdr = '0xa5C64A61D225Bc5614a2CE5fAc81926438e93844'
private_key = <private_key>
nonce = w3.eth.getTransactionCount(myAdr)

transaction = {
    'from': myAdr,
    'to': '0x1C49AA3FEAb57bc27Ad79F0De64862786766F611',
    'value': 1000000000,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}
```

```
signed_tx = w3.eth.account.signTransaction(transaction, private_key)
print("Sending transaction...")
txn_hash = w3.eth.sendRawTransaction(signed_tx.rawTransaction)
print(txn_hash)
txn_receipt = w3.eth.waitForTransactionReceipt(txn_hash)
print(txn_receipt)
```

CHAIN_ID	Chain(s)
1	Ethereum mainnet
2	Morden (disused), Expanse mainnet
3	Ropsten
4	Rinkeby
5	Goerli
42	Kovan
1337	Geth private chains (default)

1 Ether = 1 Milyar Gwei = 1 M x 1 M Wei

denomination	amount in wei
wei	1
kwei	1000
babbage	1000
femtoether	1000
mwei	1000000
lovelace	1000000
picoether	1000000
gwei	1000000000
shannon	1000000000
nanoether	1000000000
nano	1000000000
szabo	1000000000000
microether	1000000000000
micro	1000000000000
finney	1000000000000000
milliether	1000000000000000
milli	1000000000000000
ether	1000000000000000000
kether	1000000000000000000000
grand	1000000000000000000000
mether	1000000000000000000000000
gether	1000000000000000000000000000
tether	1000000000000000000000000000000

Kirim transaksi

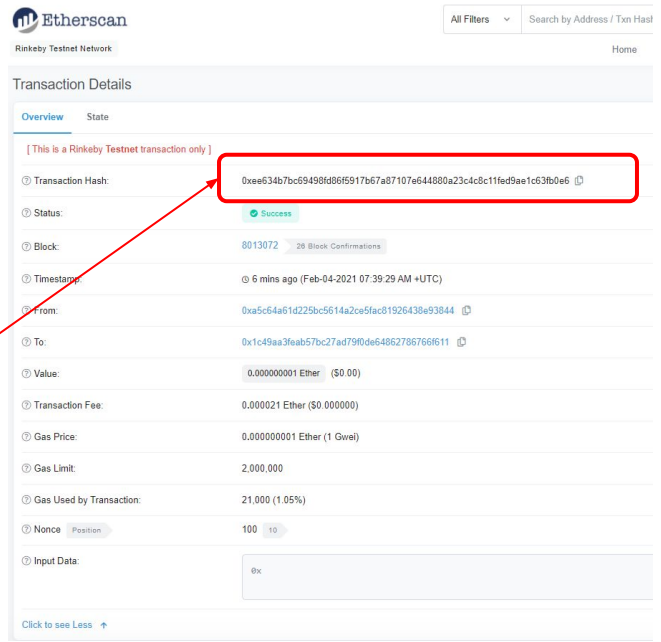
- Beri digital signature pada transaksi yang akan dikirim menggunakan kunci private
- Kirim transaksi melalui server infura
- Tunggu transkrip pengiriman transaksi dari server infura

```
from web3 import Web3
from web3.auto.infura import w3
from web3.middleware import geth_poa_middleware

w3 =
Web3(Web3.HTTPProvider('https://rinkeby.infura.io/v3/4ae814529fcf43d
db3eec654d77c5578'))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)

myAdr = '0xa5C64A61D225Bc5614a2CE5fAc81926438e93844'
private_key = <private_key>
nonce = w3.eth.getTransactionCount(myAdr)

transaction = {
    'from': myAdr,
    'to': '0x1C49AA3FEAb57bc27Ad79F0De64862786766F611',
    'value': 1000000000,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}
```

```
signed_tx = w3.eth.account.signTransaction(transaction, private_key)
print("Sending transaction...")
txn_hash = w3.eth.sendRawTransaction(signed_tx.rawTransaction)
print(txn_hash)
txn_receipt = w3.eth.waitForTransactionReceipt(txn_hash)
print(txn_receipt)
```

Verifikasi transaksi

- Bagaimana cara memastikan bahwa server infura benar-benar telah mencatatkan transaksi ke jejaring blockchain?
- Cek transaksi menggunakan jasa blok eksplorer:
  - Ethereum rinkeby: rinkeby.etherscan.io
  - Salin hash transaksi ke kolom pencarian di website etherscan



Create a zero-payment transaction

Transaksi pengiriman dengan nilai 0 ether.

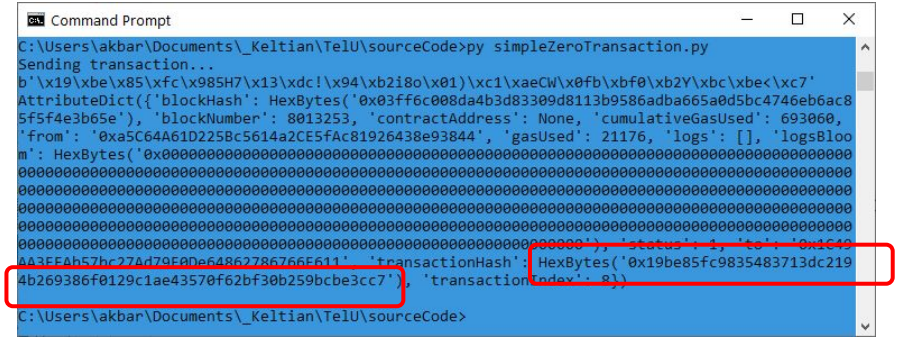
- Bertujuan untuk pengiriman data melalui kolom data dari transaksi.
- Biaya transaksi semakin besar sesuai ukuran data
- <msg>.encode() → format biner
- Kirim transaksi dengan cara yang sama

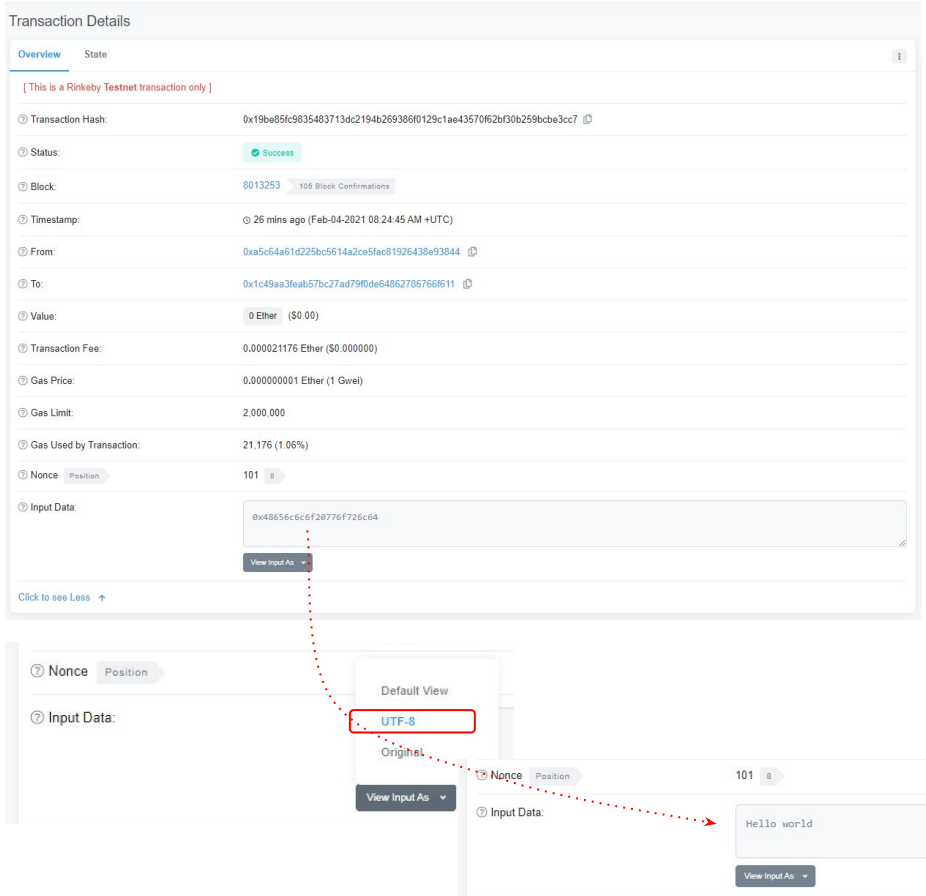
- Data yang tercatat di transaksi bersifat publik dan dapat dibaca oleh semua orang
- Gunakan enkripsi untuk menjamin kerahasiaan data
- Zero-payment: transaksi pencatatan data paling murah (mainnet)
    - Smart contract: lebih mahal (transaction fee), tetapi dapat memiliki fitur access control

```
from web3 import Web3
from web3.auto.infura import w3
from web3.middleware import geth_poa_middleware

w3 =
Web3(Web3.HTTPProvider('https://rinkeby.infura.io/v3/4ae814529fcf43d
db3eec654d77c5578'))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)

myAdr = '0xa5C64A61D225Bc5614a2CE5fAc81926438e93844'
nonce = w3.eth.getTransactionCount(myAdr)
message = "Hello world"

transaction = {
    'from': myAdr,
    'to': '0x1C49AA3FEAb57bc27Ad79F0De64862786766F611',
    'value': 0,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'data': message.encode(),
    'chainId': 4
}
private_key = <private_key>
signed_tx = w3.eth.account.signTransaction(transaction, private_key)
print("Sending transaction...")
txn_hash = w3.eth.sendRawTransaction(signed_tx.rawTransaction)
print(txn_hash)
txn_receipt = w3.eth.waitForTransactionReceipt(txn_hash)
print(txn_receipt)
```

Crosscheck via etherscan

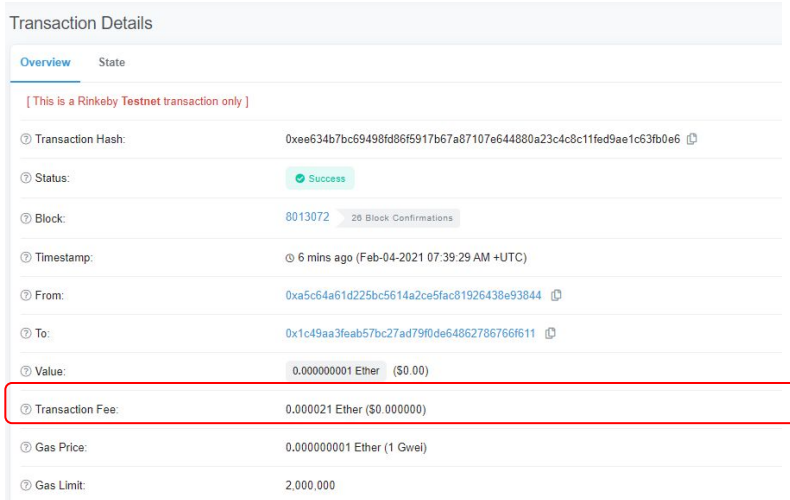
Ubah format data menjadi utf-8 untuk melihat pesan dalam format teks.



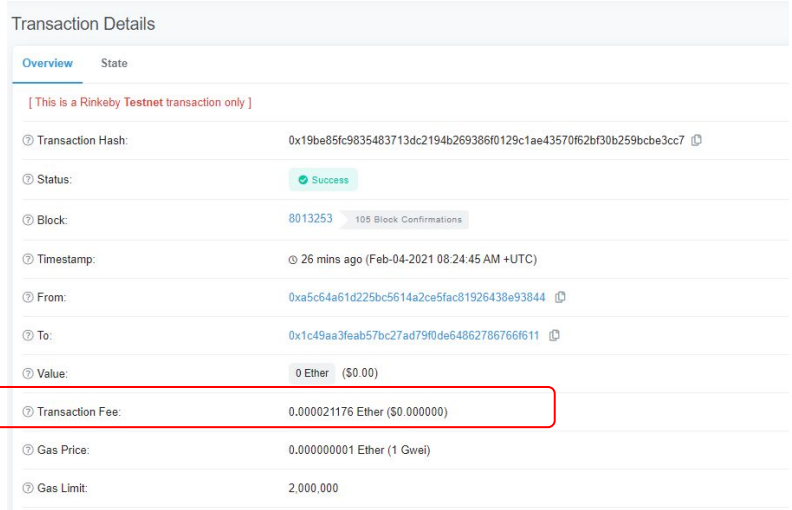


Perbandingan biaya transaksi

Transaksi transfer



Transaksi dengan data = “Hello world”
Extra fee: 1176 Gwei



3. Smart contracts Transaction

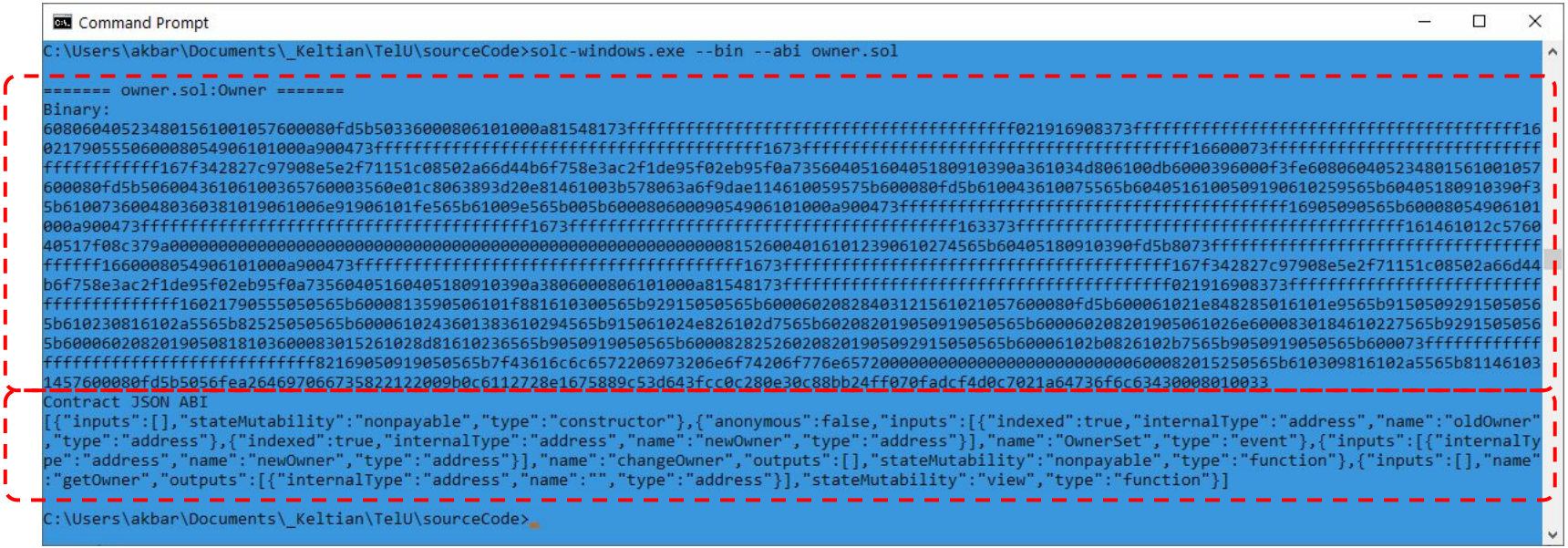
- Compile smart contract
- Deploy smart contract
- Call smart contract function
- Getter/view transaction

Kompilasi smart contract

- *Compile* solidity program untuk menghasilkan **bytecode** dan *application binary interface* (**ABI**)
- Bytecode: kode program dalam format binary
- ABI: kumpulan nama dan struktur fungsi dari program solidity dalam format .json

Bytecode

ABI



Compile smart contract

- Compiler: solc (https://github.com/ethereum/solidity/releases)
  - Lokasi download file .exe → file path dari compiler
- solc-windows.exe --bin --abi owner.sol
  - --bin: menampilkan bytecode
  - --abi: menampilkan abi

- Module contract.py:
  - from contract import compileContract
  - abi, bytecode = compileContract(<file path compiler ".exe">, <file solidity ".sol">)
  - <contract> = w3.eth.contract(abi=abi, bytecode=bytecode)
  -
  - from contract import genContract
  - <contract> = genContract(<file path compiler ".exe">, <file solidity ".sol">)

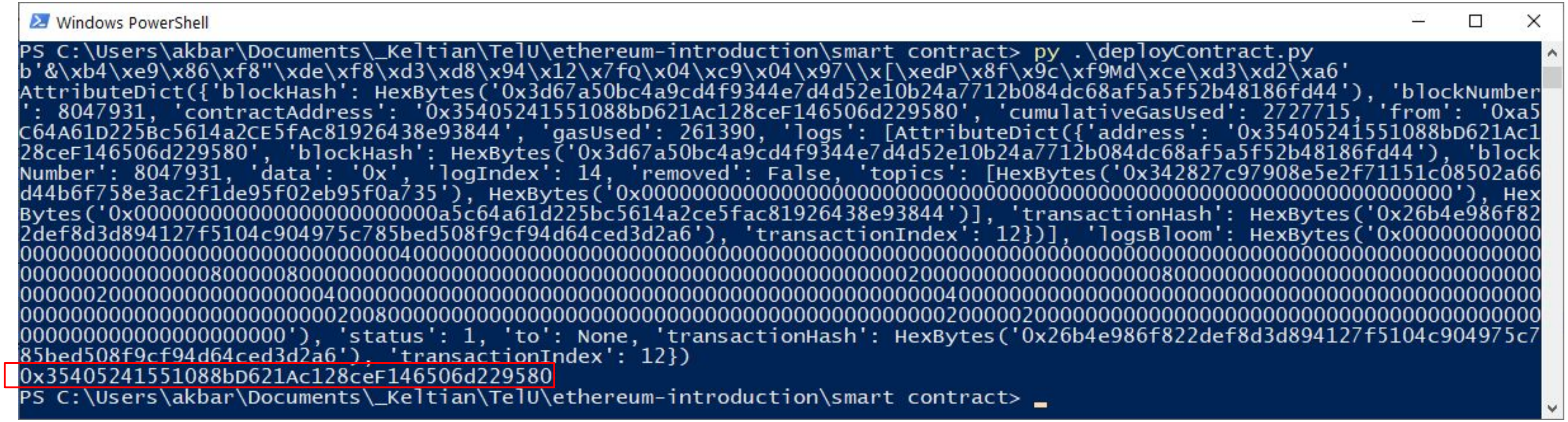
Deploy smart contract

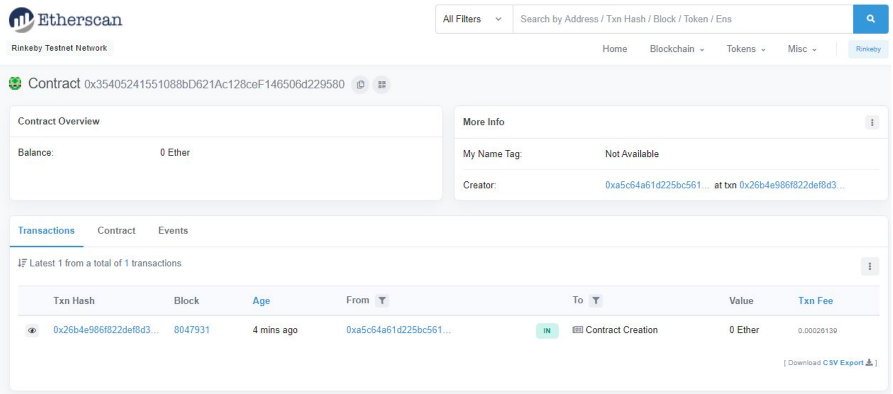
- Compile dan buat smart contract
  - Modul contract.py: compileContract()
- Kirim transaksi via Infura
  - Transaksi harus diberi digital signature sebelum dikirim
  - Susun transaksi, dengan kolom data bernilai default
  - Kolom data akan otomatis dihasilkan saat pembuatan transaksi smart contract
  - Pembuatan transaksi: <nama contract>.<nama fungsi>(<parameter fungsi>).buildTransaction(<transaksi>)
  - Deploy: fungsi constructor:
    - deploy_txn = <nama contract>.**constructor**().buildTransaction(transaction)

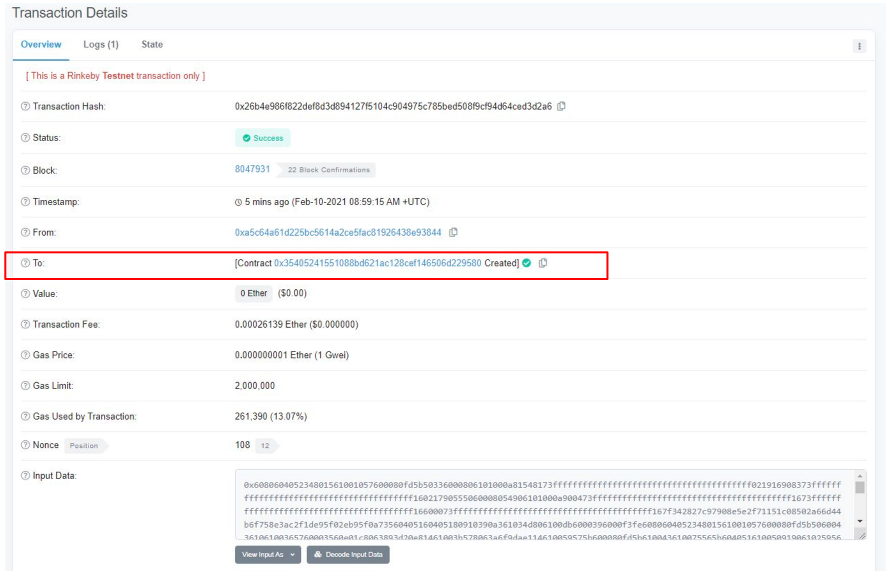
Deploy smart contract

- Compile smart contract
- Susun transaksi
- Generate transaksi
- Beri digital signature
- Kirim transaksi

```
...

myAdr = '0xa5C64A61D225Bc5614a2CE5fAc81926438e93844'
private_key = <private_key>
nonce = w3.eth.getTransactionCount(myAdr)

abi, bc = compileContract("solc-windows.exe","owner.sol")
abi = json.loads(abi)
myContract = w3.eth.contract(abi=abi, bytecode=bc)

transaction = {
    'from': myAdr,
    'value': 0,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}

deploy_txn = myContract.constructor().buildTransaction(transaction)
signed_txn = w3.eth.account.sign_transaction(deploy_txn,
private_key=private_key)
txn_hash = w3.eth.sendRawTransaction(signed_txn.rawTransaction)
print(txn_hash)
txn_receipt = w3.eth.waitForTransactionReceipt(txn_hash)

print(txn_receipt)
print(txn_receipt['contractAddress'])
```







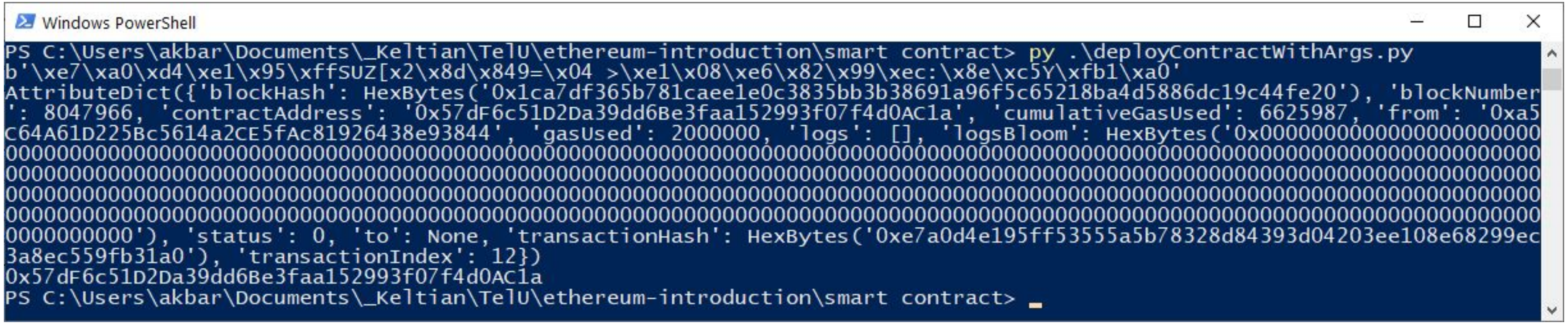
Alamat deploy:
0x35405241551088bD621Ac128ceF146506d229580

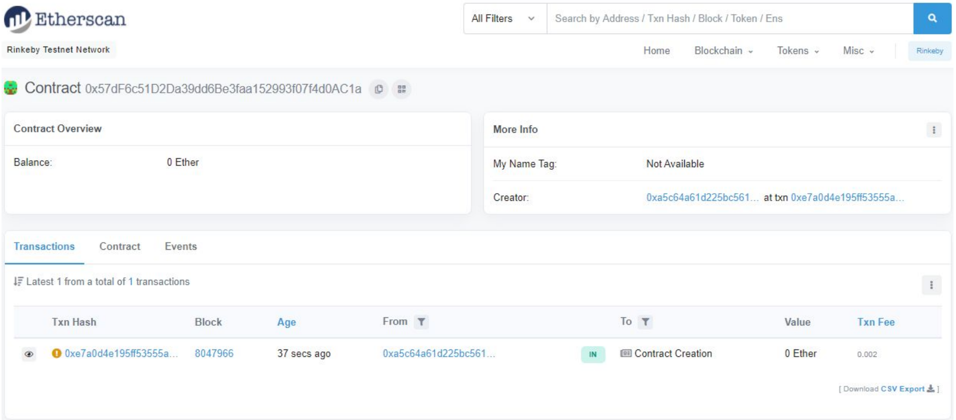
Deploy dengan parameter

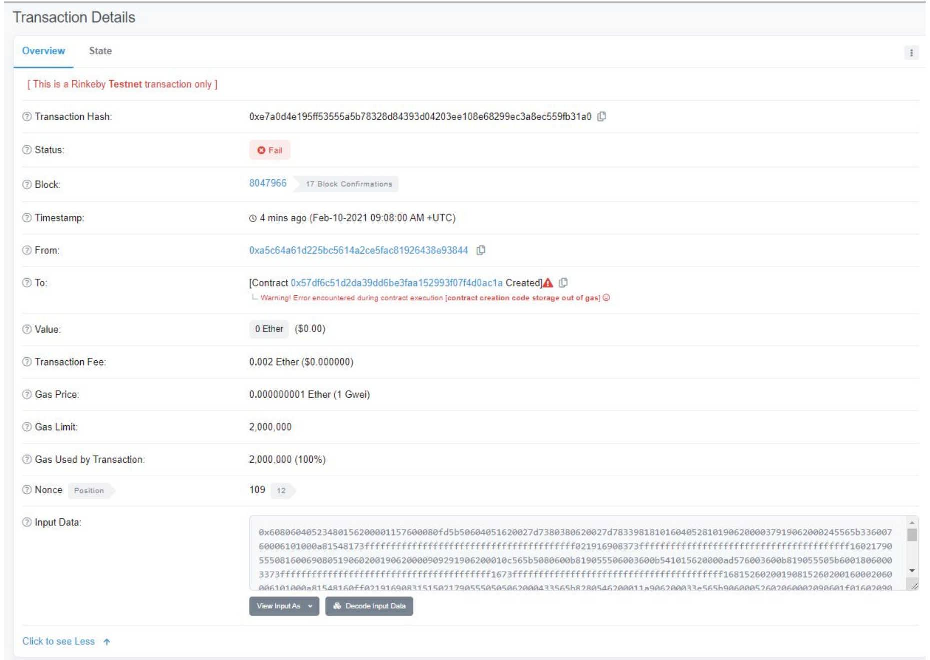
- Modul contract.py: genContract
- Parameter: <file compiler>,<file solidity>,<transaksi>,[parameter constructor])
- Smart contract: Forum.sol:
  - constructor(nama forum, nilai threshold konten)
    - Nama: “OurForum”
    - Threshold: 3 voting
- Biaya transaksi sesuai ukuran smart contract:
  - Gas: 200000 tidak mencukupi
  - Out-of-gas error
  - Naikkan Gas menjadi: 400000

```
...

transaction = {
    'from': myAdr,
    'value': 0,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}

deploy_txn = genContract("solc-windows.exe", "forum.sol",
transaction, ["OurForum",3] )

...
```

Smart contract: forum.sol







Deploy smart contract gagal karena out-of-gas
(Gas: 200000)

Windows PowerShell

```
PS C:\Users\akbar\Documents\_Keltian\TelU\ethereum-introduction\smart contract> py .\deployContractWithArgs.py
b'\xd6\xb4\x1b\xc8c\x80S\xe8\xdc v\xd7\xea\xf8\x80X\x8em_\xc8\xc0y\x01\xfa3\x0b\x10\xb0l1\xe4Z'
AttributeDict({'blockHash': HexBytes('0x968bfc79bc4374a7c2055e34489a433c09c395ae0be1a1738dd0f7f657017777'), 'blockNumber
': 8047994, 'contractAddress': '0xcbF5B41113B243A9b4EB0BF59AfBD5543A989908', 'cumulativeGasUsed': 3397008, 'from': '0xa5
C64A61D225Bc5614a2CE5fAc81926438e93844', 'gasUsed': 2109197, 'logs': [], 'logsBloom': HexBytes('0x0000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000'), 'status': 1, 'to': None, 'transactionHash': HexBytes('0xd6b41bc8638053e8dc2076d7eaf880588e6d5fc8c07901fa33
0b10b06c31e45a'), 'transactionIndex': 11})
0xcbF5B41113B243A9b4EB0BF59AfBD5543A989908
PS C:\Users\akbar\Documents\_Keltian\TelU\ethereum-introduction\smart contract>
```

Ubah gas menjadi 400000
→ gasUsed: 2109197 (>200000)
→ 52.73 %

Sisa gas yang tidak terpakai dikembalikan ke pengirim:

- Hanya ether sebesar gasUsed yang ditransfer ke miner pembuat block

Transaction Hash:	0xd6b41bc8638053e8dc2076d7eaf880588e6d5fc8c07901fa330b10b06c31e45a
Status:	Success
Block:	8047994 5 Block Confirmations
Timestamp:	1 min ago (Feb-10-2021 09:15:00 AM +UTC)
From:	0xa5c64a61d225bc5614a2ce5fac81926438e93844
To:	[Contract 0xcbf5b41113b243a9b4eb0bf59afbd5543a989908 Created]
Value:	0 Ether ($0.00)
Transaction Fee:	0.002109197 Ether ($0.000000)
Gas Price:	0.000000001 Ether (1 Gwei)
Gas Limit:	4,000,000
Gas Used by Transaction:	2,109,197 (52.73%)
Nonce Position	111 11
Input Data:	0x608060405234801562000011576000080fd5b50604051620027d7380380620027d783

Transaksi pemanggilan fungsi

- Jalin koneksi ke smart contract dengan menyertakan alamat smart contract (hasil deploy)
  - Alamat: 0x5F06b8d51Cd337d48913C365000CA3b1465dAbA0
- Format pemanggilan fungsi:
  - <nama contract>.functions.<nama fungsi>(<parameter fungsi>).buildTransaction(<transaksi>)
- Smart contract: owner.sol → fungsi pemindahan kepemilikan (changeOwner)
- Parameter fungsi: alamat penerima contract (0x1C49AA3FEAb57bc27Ad79F0De64862786766F611)
  - Pengirim transaksi = pembuat smart contract/pemilik sekarang

Smart contract: owner.sol

```
...

myContract = w3.eth.contract(address=" 0x5F06b8d51Cd337d48913C365000CA3b1465dAbA0 ", abi=abi)

transaction = {
    'from': myAdr,
    'value': 0,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}

deploy_txn =
myContract.functions.changeOwner( '0x1C49AA3FEAb57bc27Ad79F0De64862786766F611' ).buildTransaction(transaction)

...
```

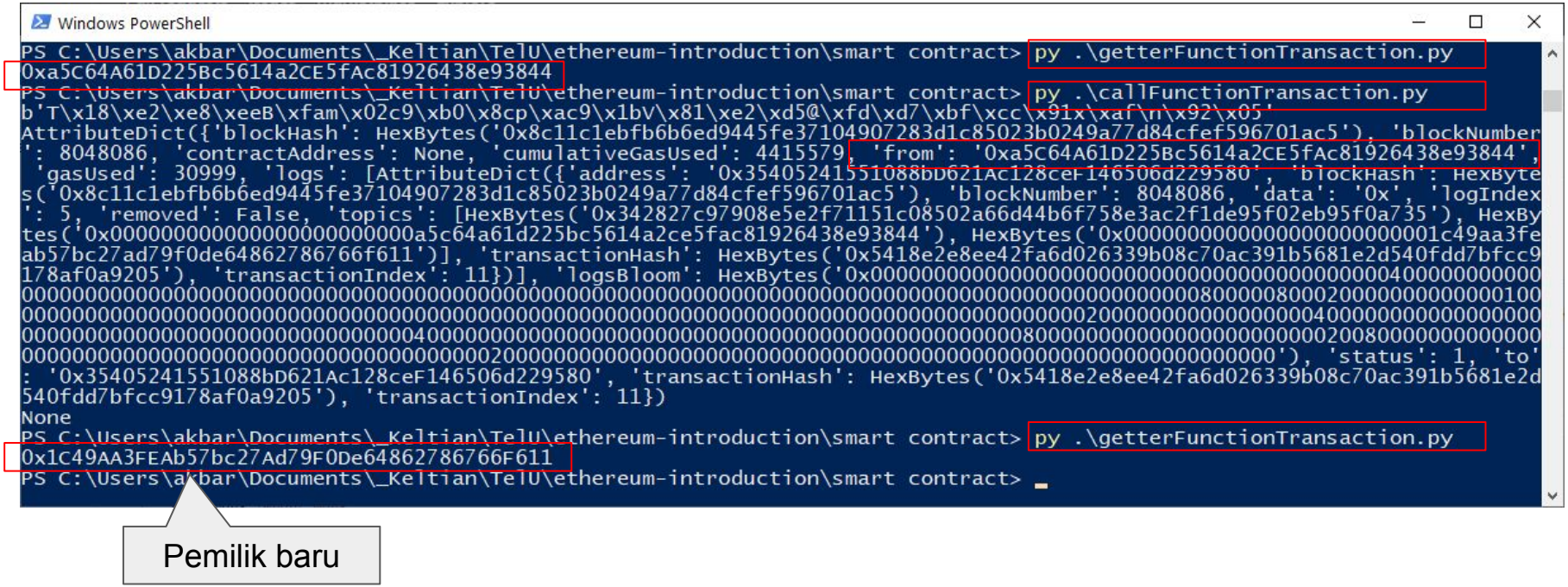
Besaran gas menyesuaikan ukuran kode fungsi yang dipanggil

Getter function

- Pemanggilan seperti halnya memanggil fungsi smart contract biasa
- Tidak perlu diberi digital signature, langsung mendapatkan hasil (tanpa dicatat di blockchain)
- Smart contract owner.sol → getOwner(): meminta informasi siapa pemilik kontrak
- <contract>.functions.<nama fungsi>(<parameter fungsi>).call()

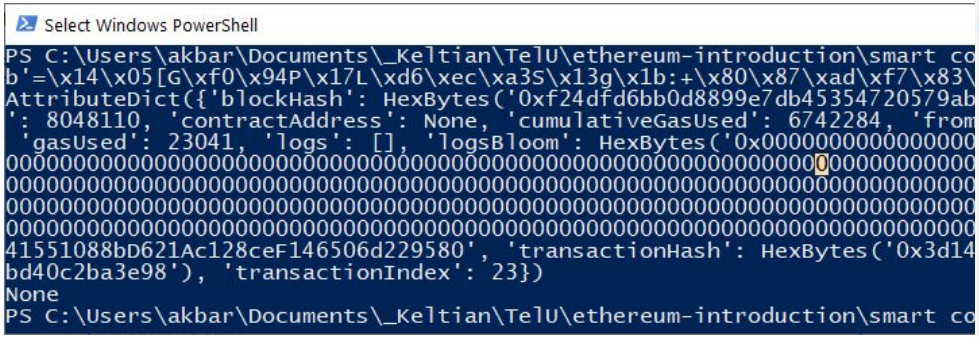
```
...

myContract = w3.eth.contract(address="0x5F06b8d51Cd337d48913C365000CA3b1465dAbA0", abi=abi)

transaction = {
    'from': myAdr,
    'value': 0,
    'gas': 2000000,
    'gasPrice': 1000000000,
    'nonce': nonce,
    'chainId': 4
}

txn_receipt = myContract.functions.getOwner().call()
print(txn_receipt)
```

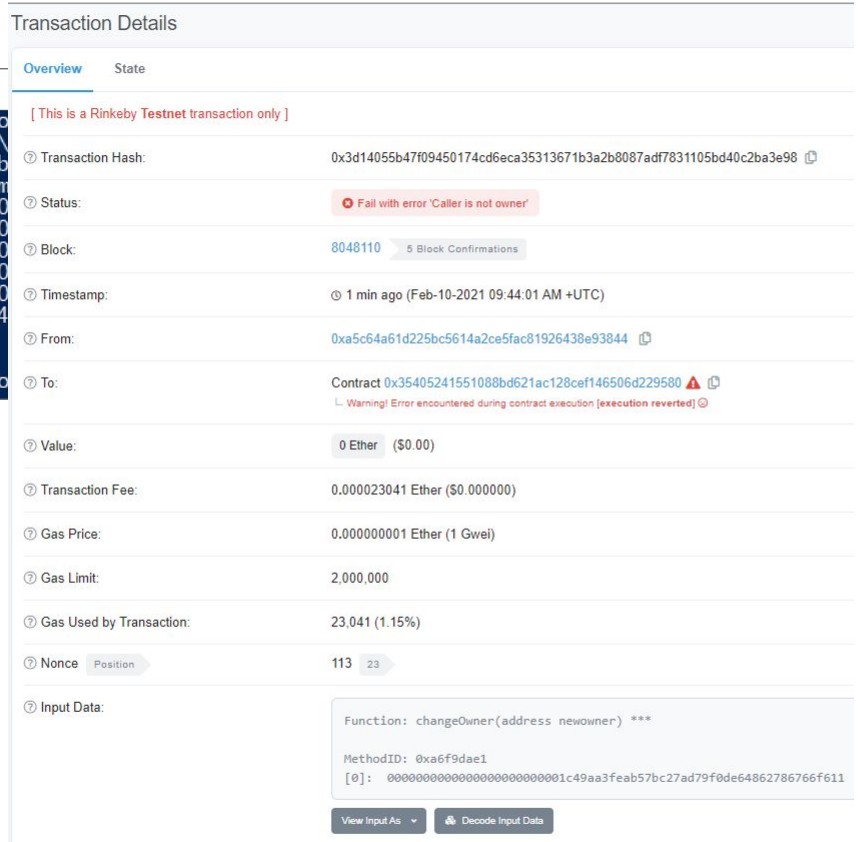
Transfer kepemilikian ke alamat: 0x1C49AA3FEAb57bc27Ad79F0De64862786766F611
Pemanggil → pemilik smart contract: 0xa5C64A61D225Bc5614a2CE5fAc81926438e93844



Kirim ulang fungsi changeOwner dengan akun pemilik lama: 0xa5C64A61D225Bc5614a2CE5fAc81926438e93844

Transaksi berhasil dicatat, tetapi fungsi gagal dieksekusi.





. . . . . .