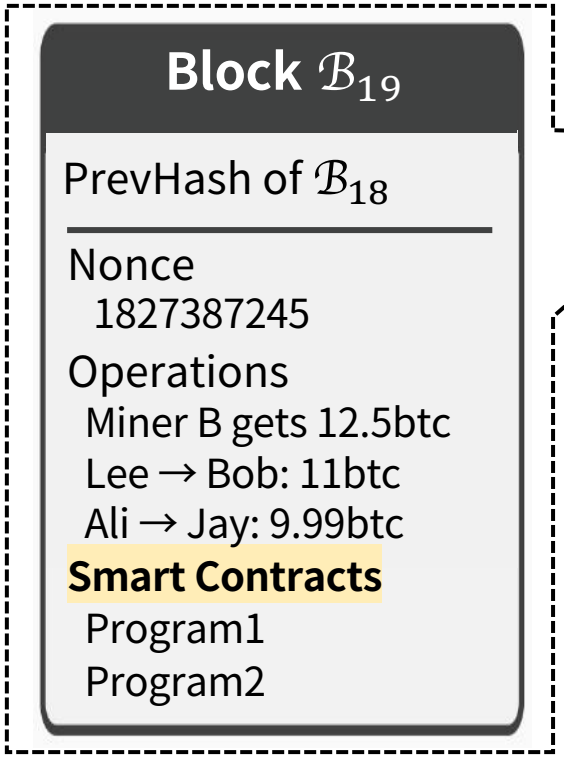
CHAPTER4 스마트 컨트랙트 실습

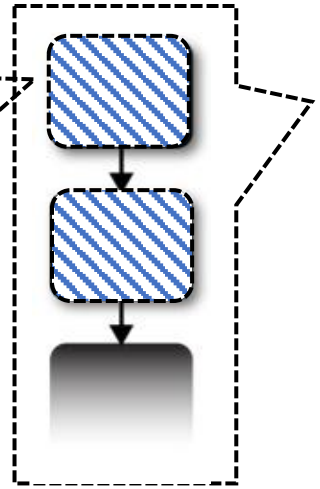
Tezos Smart Contract

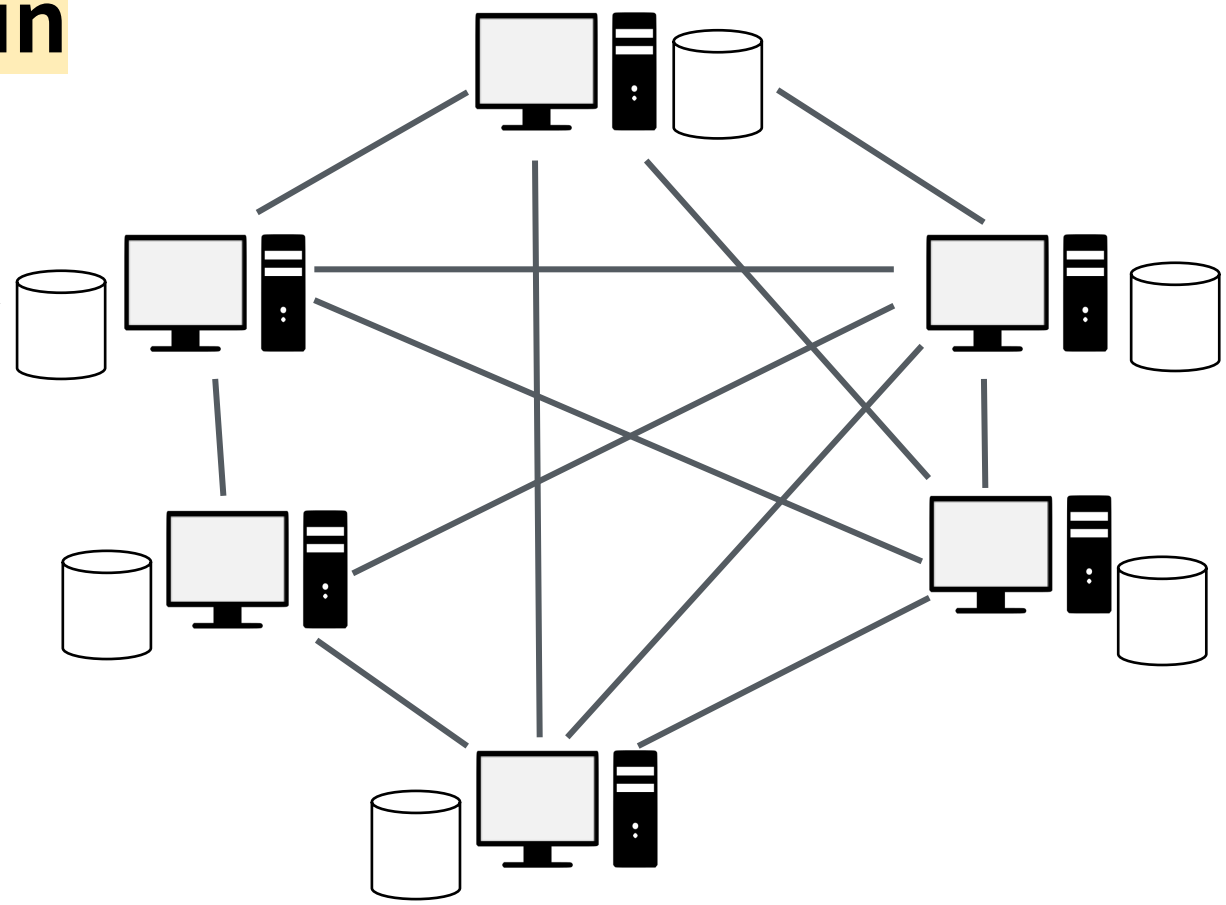
What is smart contract?

A computer program

running on top of blockchain







SMART(?) contract?

Execution

기계가 자동으로 실행함

기계는 우리가 바라는 바를 실행하지 않음

기계는 적힌 바를 실행할 뿐

Verification Technologies

1st Generation	Syntax analysis Lexical analyzer, parser Completed in 70s
2nd Generation	Type checking Completed in 90s CS' Signature achievement for the past 30yrs
3rd Generation	Static analysis Theorem proving, model checking, abstract interpretation Still being developed

Verification of smart contract?

Conventional

Teaching

- Guide lines
- Good practices
- Anti-patterns

Testing

- Manual code review
- Unit testing
- Penetration testing
- External audit
- Bug bounty

Formal Verification

Benefits

- Comprehensive
- Unambiguous
- Complete
- Language-independent

Costs

- Steep learning curve
- High entry barrier
- Human error

Age of Hype

Resources on blockchain

The same storage and computation across distributed nodes

Very expensive and Inefficient

Simple business logic

Smart contracts should be as simple as possible

Contract : x → sqrt x
App : ( ) → f 9

▶

Contract : x y → if x * x = y then x else fail
App : ( ) → f (sqrt 9) 9

Challenge

Given a program, how to separate it into off-chain and on-chain

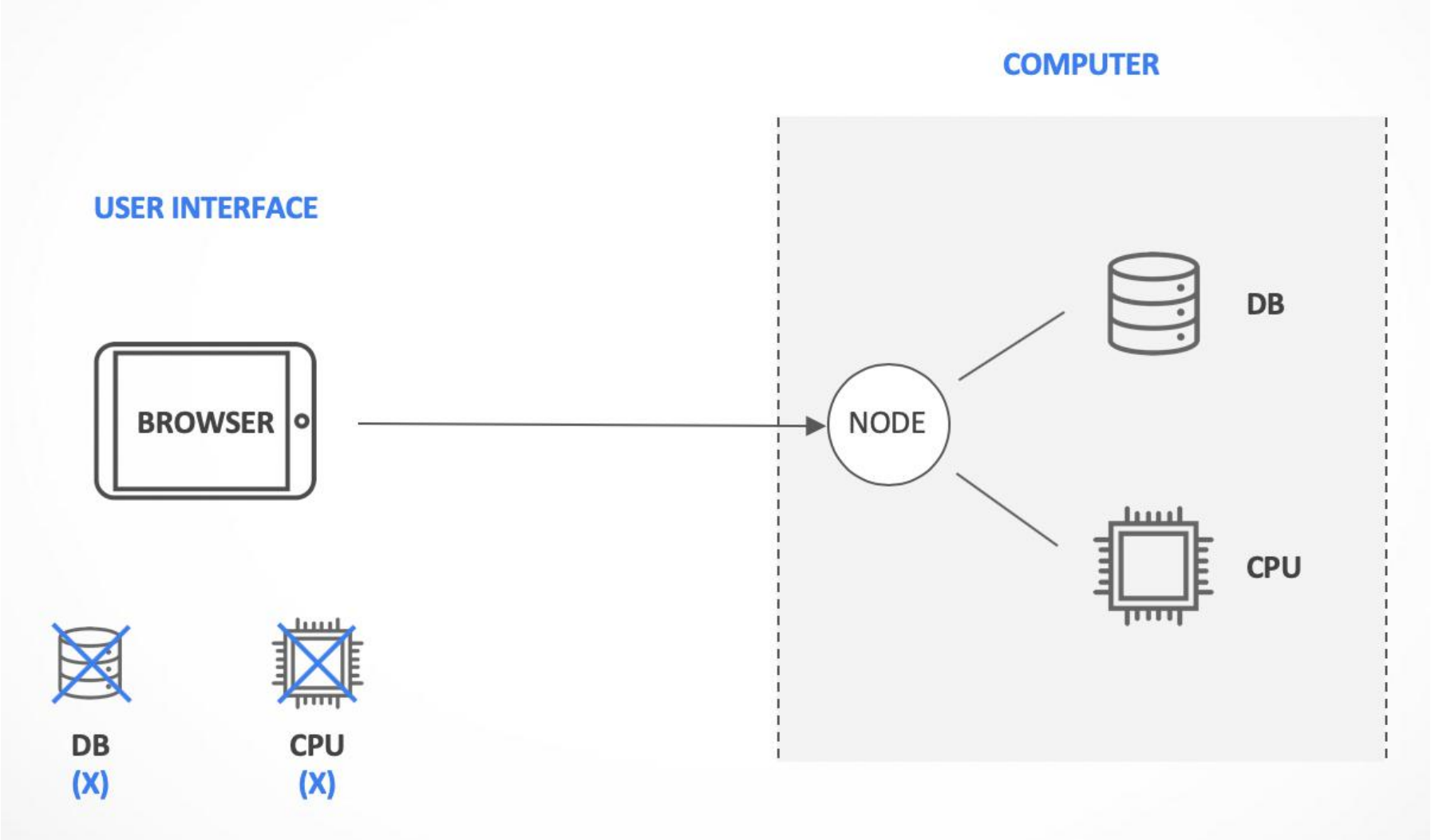
Smart contracts are **neither smart nor (legal) contract**

Smart contracts are **neither smart nor (legal) contract**

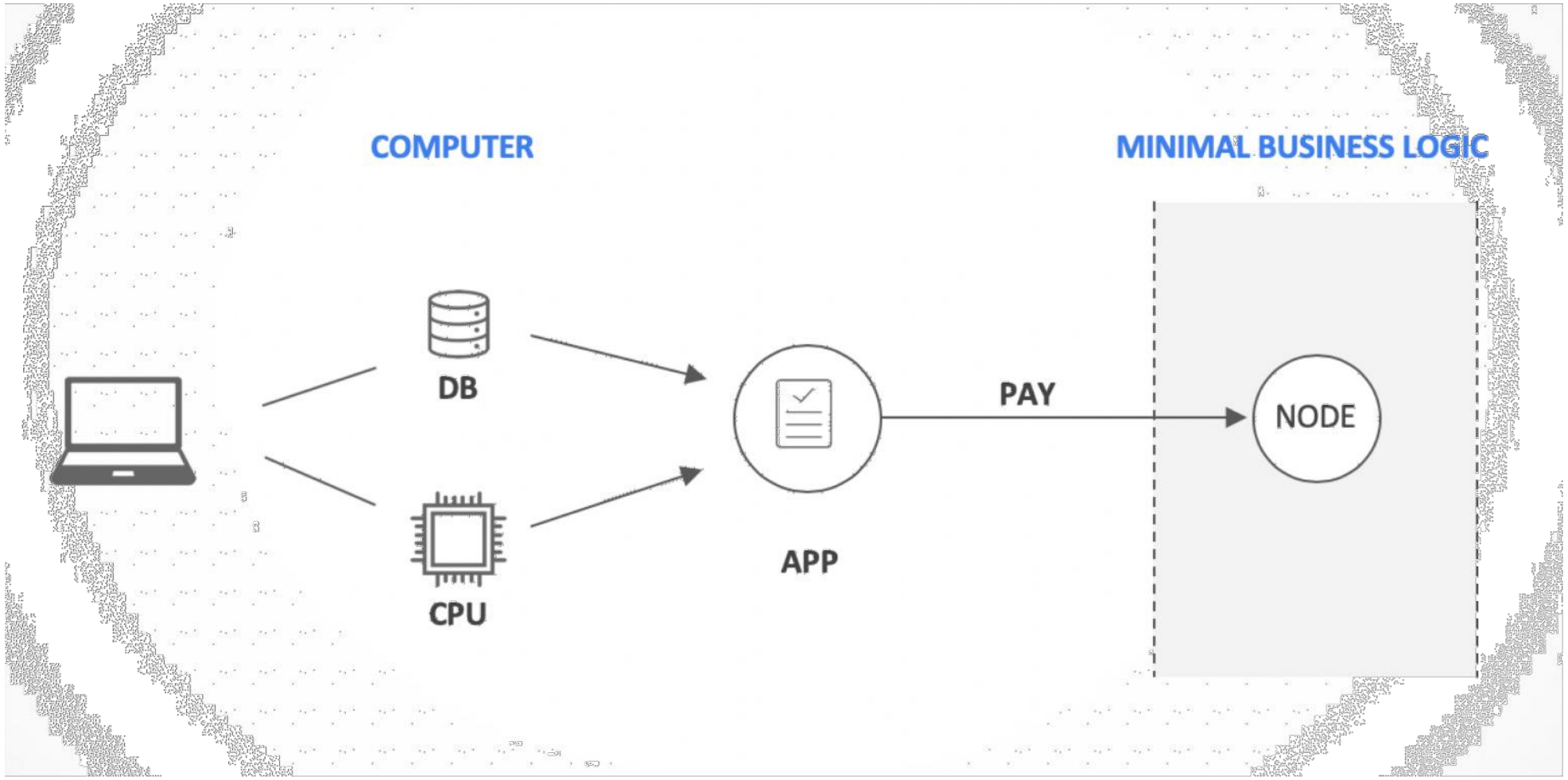
No One-Size-Fits-All

World computer	Simple business logic	Industrial-scale apps
Arbitrary programs	Automated escrows	Arbitrary programs
General purpose	Domain specific	General purpose
Gas model	Gas model	Ownership model
Very low level VM	Base language both low and high	WASM
Very hard for FV	Easy target for FV	Very hard for FV

World Computer



Automated Escrow



Bytecode vs. Michelson


```
0x6060604052341561000f57600080fd5b60405161046d38038061046d833981016040
5280805190910190506000818051610003d9291602001906100445650506100df565b
828054600181600116156101000203166002900490600052602060002090601f016020
900481019282601f106100855780516 0ff191683800117855610 0b2565b8280016001
0185558215610 0b2579182015b828111156100b2578251825591602001919060010190
610097565b506100be9291506100c2565b5090565b6100dc91905b808211156100be57
600081556001016100c8565b90565b61037f806100ee6000396000f300606060405260
0436106100565763ffffffff7c0100000000000000000000000000000000000000000000
0000000000000000600035041663954ab4b2811461005b578063a4136862146100e55780
63ef690cc014610138575b600080fd5b34156100665760008 0fd5b61006e61014b565b
604051602080825281908101838181518152602001915080519060200190808383600 0
5b838110156100aa578082015183820152602001610092565b50505050905090810190
601f1680156100d757808203805160018360200361010 00a031916815260200191505b
509250505060405180910390f35b34156100f057600080fd5b61013660046024813581
810190830135806020601f820181900481020160405190810160405281815292919060
208401838380828437509496506101f495505050505050565b005b3415610143576000
80fd5b61006e61020b565b6101536102a9565b60008054600181600116156101000203
166002900480601f016020809104026020016040519081016040528092919081815260
200182805460018160011615610100020316600290048015610 1e95780601f106101be
576101008083540402835291602001916101e9565b820191906000526020600020905b
815481529060010190602001808311610284578290036 01f1682
01915b505050505081565b602060405190810160405260008152905 65b828054600181
600116156101000203166002900490600052602060002090601f016020900481019282
601f106102fc57805160ff191683800117855610329565b828001600101855582156 1
0329579182015b828111156103295782518255916020019190600101906103 0e565b50
610335929150610339565b5090565b6101f191905b808211156103357600081556001
0161033f5600a165627a7a72305820352cec017ed93c8351ac6fbc835eda354ea6dbc9
e672ae6b60c16f29c49a5cd30029
```


```
parameter (pair (lambda int int) (list int));
return (list int);
storage unit;
code { DIP{NIL int};
    CAR;
    DUP;
    DIP{CAR; PAIR};     # Unpack data and setup accumulator
    CDR;
    LAMBDA (pair int (pair (lambda int int) (list int)))
       (pair (lambda int int) (list int))
       # Apply the lambda and add the new element to the list
       { DUP; CDAR;
        DIP{ DUP; DIP{CDAR}; DUP;
           CAR; DIP{CDDR; SWAP}; EXEC; CONS};
        PAIR};
    REDUCE; CDR; DIP{NIL int}; # First reduce
    LAMBDA (pair int (list int))
       (list int)
       {DUP; CAR; DIP{CDR}; CONS};
    REDUCE;         # Correct list order
    UNIT; SWAP; PAIR}    # Calling convention
```

Bytecode vs. Michelson

Michelson

- Compilation target
- Domain Specific Language (Business logic)
- Do not need to trust a compiler
- Designed to facilitate Formal verification
- Formally verified implementations (Coq, F*)

```
parameter (pair (lambda int int) (list int));
return (list int);
storage unit;
code { DIP{NIL int};
    CAR;
    DUP;
    DIP{CAR; PAIR};     # Unpack data and setup accumulator
    CDR;
    LAMBDA (pair int (pair (lambda int int) (list int)))
       (pair (lambda int int) (list int))
       # Apply the lambda and add the new element to the list
       { DUP; CDAR;
        DIP{ DUP; DIP{CDAR}; DUP;
           CAR; DIP{CDDR; SWAP}; EXEC; CONS};
        PAIR};
    REDUCE; CDR; DIP{NIL int}; # First reduce
    LAMBDA (pair int (list int))
       (list int)
       {DUP; CAR; DIP{CDR}; CONS};
    REDUCE;         # Correct list order
    UNIT; SWAP; PAIR}    # Calling convention
```

High level language

Haskell

General purpose
Functional programming language

Liquidity

1.0 released ('19.03)
Syntaxes of OCaml and ReasonML

Fi

Pre-alpha ('19.02)
JS-like syntax

LIGO

Public beta ('19.06)
PascaLIGO, CameLIGO
Developed for layer 2 scaling solution (Marigold)

SMARTPY

Being developed
Python-like syntax
Static analysis and automatic proofs

Contract Type

Account type

Account = contract

There are two types of accounts: **implicit**(default) account and **originated** account

Implicit accounts

A pair of pk and sk (tz1⋯, tz2⋯, tz3⋯).

Contracts with **no executable codes**.

Only implicit accounts can bake blocks due to security deposits.

A baker is self-delegated.

```
ubuntu@ip-172-31-23-117:~/tezos$ ./tezos-client rpc get /chains/main/blocks/head/context/
contracts/tz1KfYpyh36nPgdTejQpjoms2HT8tX6S7PhA | jq
{
  "manager": "tz1KfYpyh36nPgdTejQpjoms2HT8tX6S7PhA",
  "balance": "39122682497",
  "spendable": true,
  "delegate": {
    "setable": false,
    "value": "tz1KfYpyh36nPgdTejQpjoms2HT8tX6S7PhA"
  },
  "counter": "244"
}
```

Contract Type

Account type

Account = contract

There are two types of accounts: **implicit**(default) account and **originated** account

Originated accounts

Generated by an ***origination*** operation. (kt1…)

Contracts with **executable (michelson) codes**.

Only originated accounts can delegate with their own kt1 accounts.

The purpose of an origination is **to deploy a smart contract** or **to delegate**.

Contract Type

Originated accounts

```
ubuntu@ip-172-31-23-117:~/tezos$ ./tezos-client rpc get /chains/main/blocks/head/context/contracts/KT1TM
NFb5Y2wf9sNLenTiVTFBdvkDXu16DWB
{
  "manager": "tz1VKAYh4nE92ugzdPxE1gDxvuWvjzwKGsDt",
  "balance": "1000000",
  "spendable": false,
  "delegate": { "setable": false },
  "script": {
    "code": [
      {
        "prim": "parameter",
        "args": [ { "prim": "string" } ]
      },
      {
        "prim": "storage",
        "args": [ { "prim": "string" } ]
      },
      {
        "prim": "code",
        "args": [
          [
            { "prim": "CAR" },
            { "prim": "NIL", "args": [ { "prim": "operation" } ] },
            { "prim": "PAIR" }
          ]
         ]
      }
    ],
    "storage": { "string": "medium" }
  },
  "counter": "0"
}
```

Try Michelson

Memo (Over-write)

```
[%%version 1.04]
type storage = string
let%entry main (parameter : string) (_ : storage) =
  (([] : operation list), parameter)
```

```
parameter string;
storage string;
code { CAR ; NIL operation ; PAIR }
```

Micheslon lifecycle

Michelson starts with a stack of one pair (parameter, current storage) and ends with a stack of one pair (operation list, new storage)

(parameter , storage) → (op list , storage')

Try Michelson

Memo (Over-write)

```
[%%version 1.04]
type storage = string
let%entry main (parameter : string) (_ : storage) =
  (([] : operation list), parameter)
```

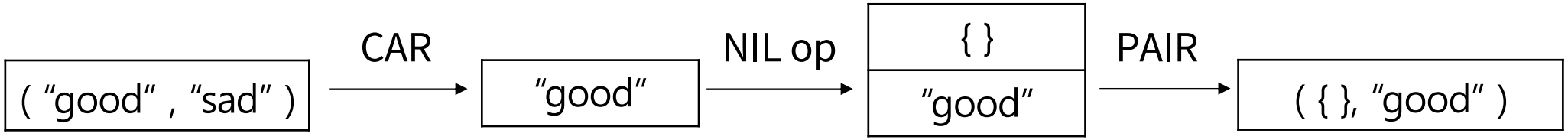
```
parameter string;
storage string;
code { CAR ; NIL operation ; PAIR }
```

Michelson operations

CAR : Access the left part of a pair. CAR / (Pair a _) : S => a : S

NIL ‘a : The empty list. NIL / S => { } : S

PAIR: Build a pair from the stack’s top two elements. PAIR / a : b : S => (Pair a b) : S



Try Michelson (ref #20)

Download resources of the camp

tezos 설치 폴더(~/tezos)로 이동

```
$ cd ~/tezos
```

```
ubuntu@ip-172-31-28-211:~/tezos$ cd ~/tezos
ubuntu@ip-172-31-28-211:~/tezos$ ls
LICENSE                docs            src                          tezos-client                 tezos-version
Makefile               dune            tests_python                 tezos-endorser-004-Pt24m4xi  vendors
README.md              dune-workspace  tezos-accuser-004-Pt24m4xi   tezos-node
active_protocol_versions emacs         tezos-admin-client           tezos-protocol-compiler
contributing.md        scripts         tezos-baker-004-Pt24m4xi     tezos-signer
```

blockchaicamp clone 후 폴더 이동

```
$ git clone https://gitlab.com/tezoskorea/blockchaincamp.git
```

```
ubuntu@ip-172-31-28-211:~/tezos$ git clone https://gitlab.com/tezoskorea/blockchaincamp.git
Cloning into 'blockchaincamp'...
remote: Enumerating objects: 61, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 61 (delta 20), reused 59 (delta 18)
Unpacking objects: 100% (61/61), done.
ubuntu@ip-172-31-28-211:~/tezos$ cd blockchaincamp/
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ ls
app.js  config.json  eztz.min.js  index.html  reference.md  simple.tz
```

Try Michelson (ref #20)

Deploy simple.tz

parameter string; storage string; code { CAR; NIL operation; PAIR }

$ ./tezos-client originate contract **simple** for **tezoskorea** transferring 0 from **tezoskorea** running ./blockchaincamp/simple.tz --init ‘“hello”’ --burn-cap 0.303 --force-low-fee

```
Node is bootstrapped, ready for injecting operations.
Estimated gas: 11314 units (will add 100 for safety)
Estimated storage: 303 bytes added (will add 20 for safety)
Enter password for encrypted key:
Operation successfully injected in the node.
Operation hash is 'ooBrsFHJGZmtmiZ8AoJyBTaVru61ba99imCvagkYcsURjwTDYSm'
Waiting for the operation to be included...
Operation found in block: BLF9VgkV6sMRzbT4bRAtacnvLEdqtg7FyhENkyeuZR1yi7f8XZo (pass: 3, offset: 1)
This sequence of operations was run:
  Manager signed operations:
    From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
    Fee to the baker: ꜩ0.001442
    Expected counter: 74450
    Gas limit: 11414
    Storage limit: 323 bytes
    Balance updates:
      tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ............. -ꜩ0.001442
      fees(tz3WXYtyDUNL91qfiCJtVUX746QpNv5i5ve5,243) ... +ꜩ0.001442
    Origination:
      From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
      For: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
```

simple: deploy할 컨트랙트에 대한 alias (닉네임)

tezoskorea: faucet에서 테조스 토큰을 옮긴 나의 주소 alias

Try Michelson (ref #20)

Deploy simple.tz

```
ubuntu@ip-172-31-28-211:~$ tezos-client originate contract simple for tezoskorea transferring 0 from tezoskorea running ./simple.tz
 -init '"hello"' --burn-cap 0.303 --force-low-fee
Node is bootstrapped, ready for injecting operations.
Estimated gas: 11314 units (will add 100 for safety)
Estimated storage: 303 bytes added (will add 20 for safety)
Enter password for encrypted key:
Operation successfully injected in the node.
Operation hash is 'ooBrsFHJGZmtmiZ8AoJyBTaVru61ba99imCvagkYcsURjwTDYSm'
Waiting for the operation to be included...
Operation found in block: BLF9VgkV6sMRzbT4bRAtacnvLEdqtg7FyhENkyeuZR1yi7f8XZo (pass: 3, offset: 1)
This sequence of operations was run:
  Manager signed operations:
    From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
    Fee to the baker: ꜩ0.001442
    Expected counter: 74450
    Gas limit: 11414
    Storage limit: 323 bytes
    Balance updates:
      tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ............. -ꜩ0.001442
      fees(tz3WXYtyDUNL91qfiCJtVUX746QpNv5i5ve5,243) ... +ꜩ0.001442
    Origination:
      From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
      For: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
      Credit: ꜩ0
      Script:
        { parameter string ;
          storage string ;
          code { CAR ; NIL operation ; PAIR } }
        Initial storage: "hello"
        No delegate for this contract
        This origination was successfully applied
        Originated contracts:
          KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
        Storage size: 46 bytes
        Paid storage size diff: 46 bytes
        Consumed gas: 11314
        Balance updates:
          tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ... -ꜩ0.046
          tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ... -ꜩ0.257

New contract KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH originated.
```

Try Michelson (ref #20)

Deploy simple.tz

```
    Origination:
      From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
      For: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
      Credit: ꜩ0
      Script:
        { parameter string ;
          storage string ;
          code { CAR ; NIL operation ; PAIR } }
        Initial storage: "hello"
        No delegate for this contract
        This origination was successfully applied
        Originated contracts:
          KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
        Storage size: 46 bytes
        Paid storage size diff: 46 bytes
        Consumed gas: 11314
        Balance updates:
          tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ... -ꜩ0.046
          tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ... -ꜩ0.257

New contract KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH originated.
```

Storage cost

0.001 ꜩ per bytes

0.46 ꜩ for 46 bytes (41 bytes for the code, 5 bytes for storage “hello”)

0.257 ꜩ for orgination

Try Michelson (ref #20)

Check the contract in local

```
$ ./tezos-client list known contracts
```

```
ubuntu@ip-172-31-28-211:~/tezos$ ./tezos-client list known contracts
simple: KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
tezoskorea: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
myWallet: tz1ZpRoQUksG3uPe2yThjst5MJUXnj6329vN
john: tz1LsDttshWsAAPHZqJGYuSZfrkGMcSpZ25B
kim: tz1M7sFiTqDffL5Nj4znQcXixcVwZEbW7Y1z
```

Try Michelson (ref #20)

Check the contract in the blockchain

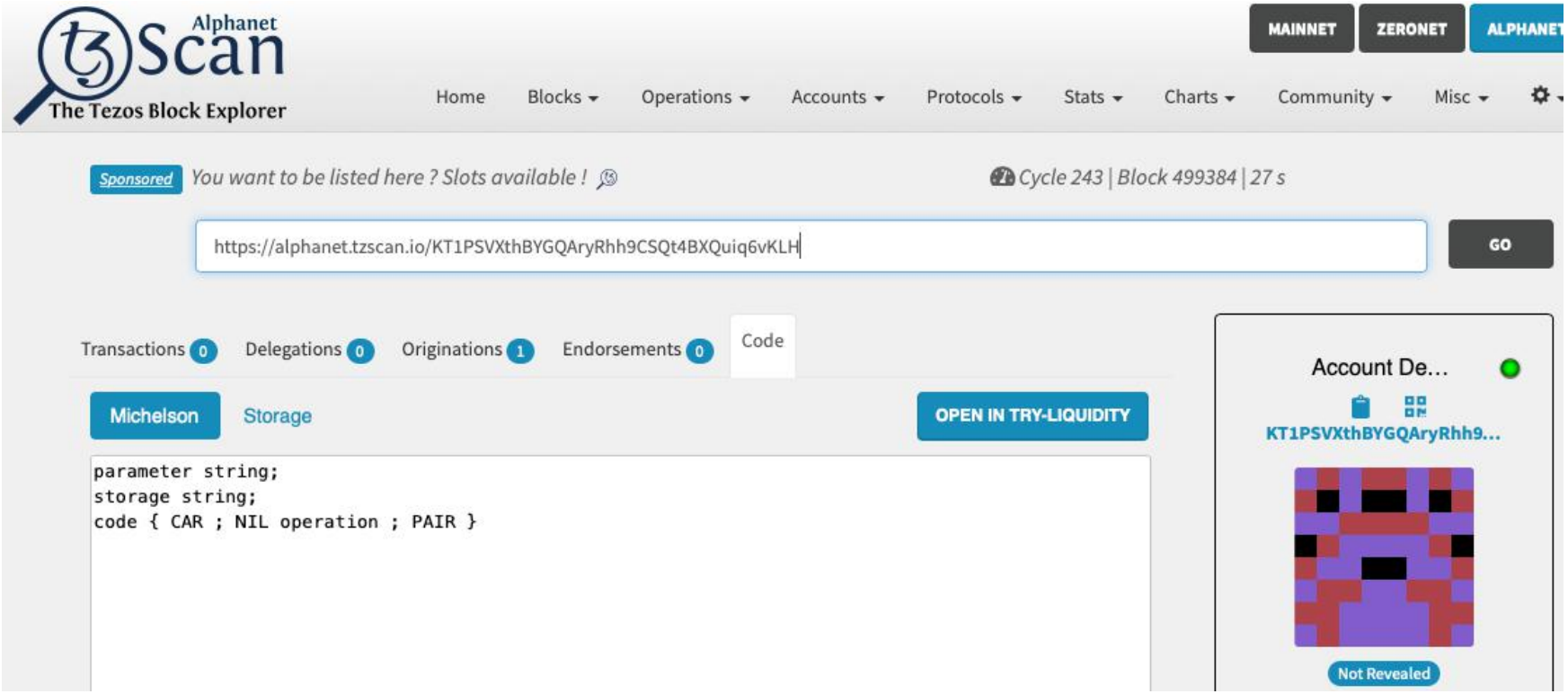
“hello”

$ ./tezos-client rpc get /chains/main/blocks/head/context/contracts/**KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH**

```
ubuntu@ip-172-31-28-211:~$ tezos-client rpc get /chains/main/blocks/head/context/contracts
/KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
{ "manager": "tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD", "balance": "0",
  "spendable": false, "delegate": { "setable": false },
  "script":
    { "code":
        [ { "prim": "parameter", "args": [ { "prim": "string" } ] },
          { "prim": "storage", "args": [ { "prim": "string" } ] },
          { "prim": "code",
            "args":
              [ [ { "prim": "CAR" },
                  { "prim": "NIL", "args": [ { "prim": "operation" } ] },
                  { "prim": "PAIR" } ] ] } ],
      "storage": { "string": "hello" } }, "counter": "0" }
```

Try Michelson (ref #20)

Check the contract in a block explorer



Try Michelson (ref #20)

Interaction with the contract

$ ./tezos-client transfer 0 from **tezoskorea** to **simple** --arg '"world"'

```
ubuntu@ip-172-31-28-211:~$ tezos-client transfer 0 from tezoskorea to KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
--arg '"world"'
Node is bootstrapped, ready for injecting operations.
Estimated gas: 11705 units (will add 100 for safety)
Estimated storage: no bytes added
Enter password for encrypted key:
Operation successfully injected in the node.
Operation hash is 'oo18eHngcrvoEAWtsPcJdG2o5ppH8bUmK4Ggohc82sLAk9nQvPU'
Waiting for the operation to be included...
Operation found in block: BMXkZKB8GUwbKb1PiHabsGPcguZR6imBQqo9kfE1meSome1xDwg (pass: 3, offset: 0)
This sequence of operations was run:
  Manager signed operations:
    From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
    Fee to the baker: ꜩ0.001446
    Expected counter: 74451
    Gas limit: 11805
    Storage limit: 0 bytes
    Balance updates:
      tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD ............. -ꜩ0.001446
      fees(tz3gN8NTLNLJg5KRsUU47NHNVHbdhcFXjjaB,287) ... +ꜩ0.001446
    Transaction:
      Amount: ꜩ0
      From: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
      To: KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
```

Try Michelson (ref #20)

Check the contract in the blockchain

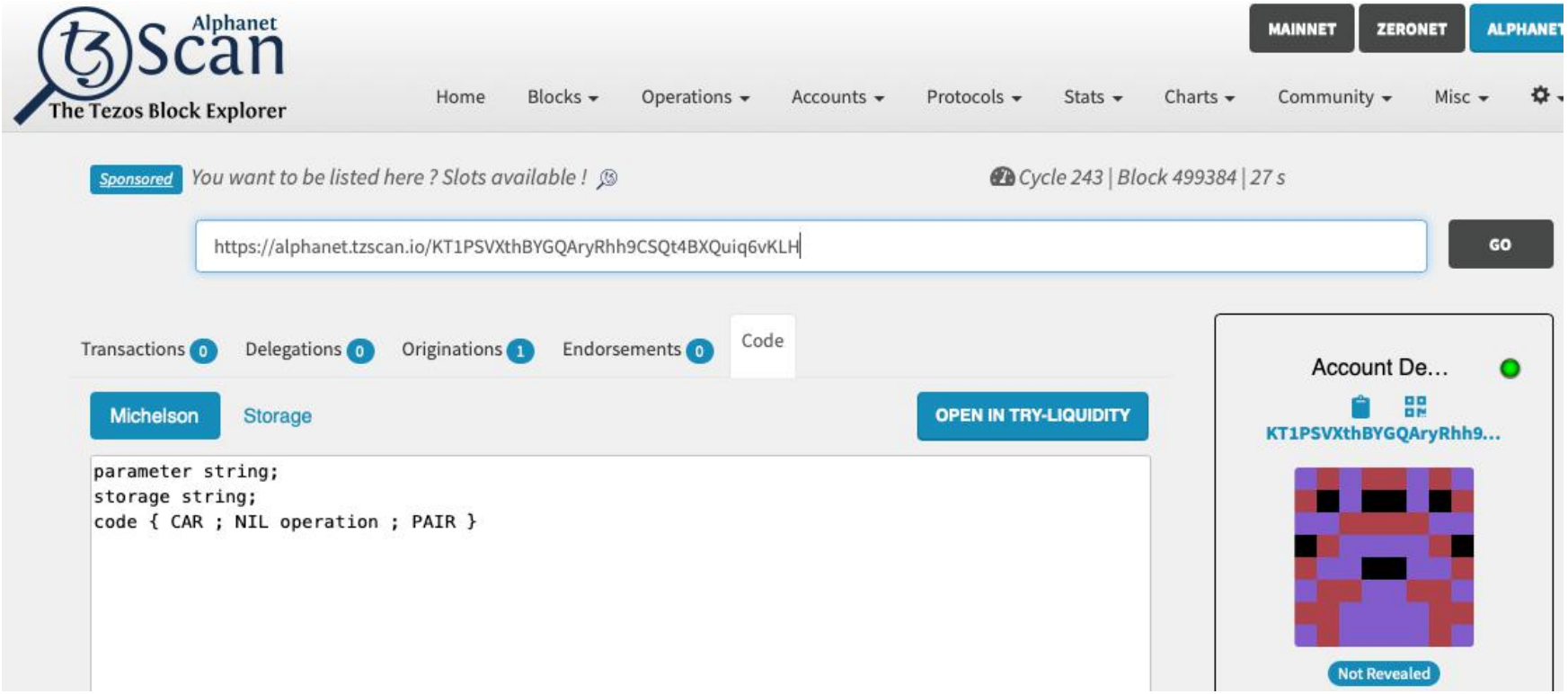
“world”

$ ./tezos-client rpc get /chains/main/blocks/head/context/contracts/**KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH**

```
ubuntu@ip-172-31-28-211:~$ tezos-client rpc get /chains/main/blocks/head/context/contracts/KT1PSVXthBYGQAry
Rhh9CSQt4BXQuiq6vKLH
{ "manager": "tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD", "balance": "0",
  "spendable": false, "delegate": { "setable": false },
  "script":
    { "code":
        [ { "prim": "parameter", "args": [ { "prim": "string" } ] },
          { "prim": "storage", "args": [ { "prim": "string" } ] },
          { "prim": "code",
            "args":
              [ [ { "prim": "CAR" },
                  { "prim": "NIL", "args": [ { "prim": "operation" } ] },
                  { "prim": "PAIR" } ] ] } ],
      "storage": { "string": "world" } }, "counter": "0" }
```

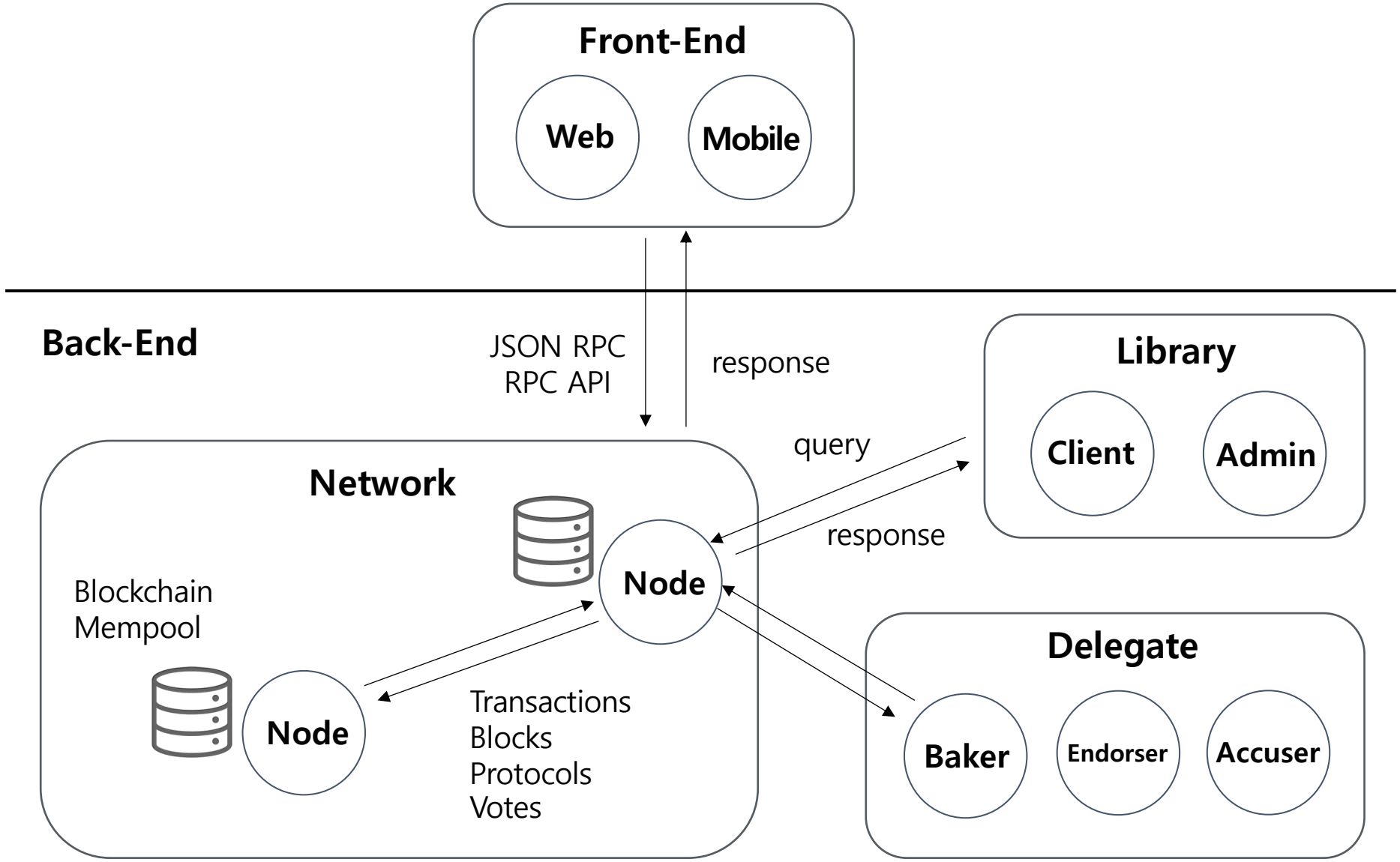
Try Michelson (ref #20)

Check the contract in a block explorer



DApp (Front-end)

Data flow

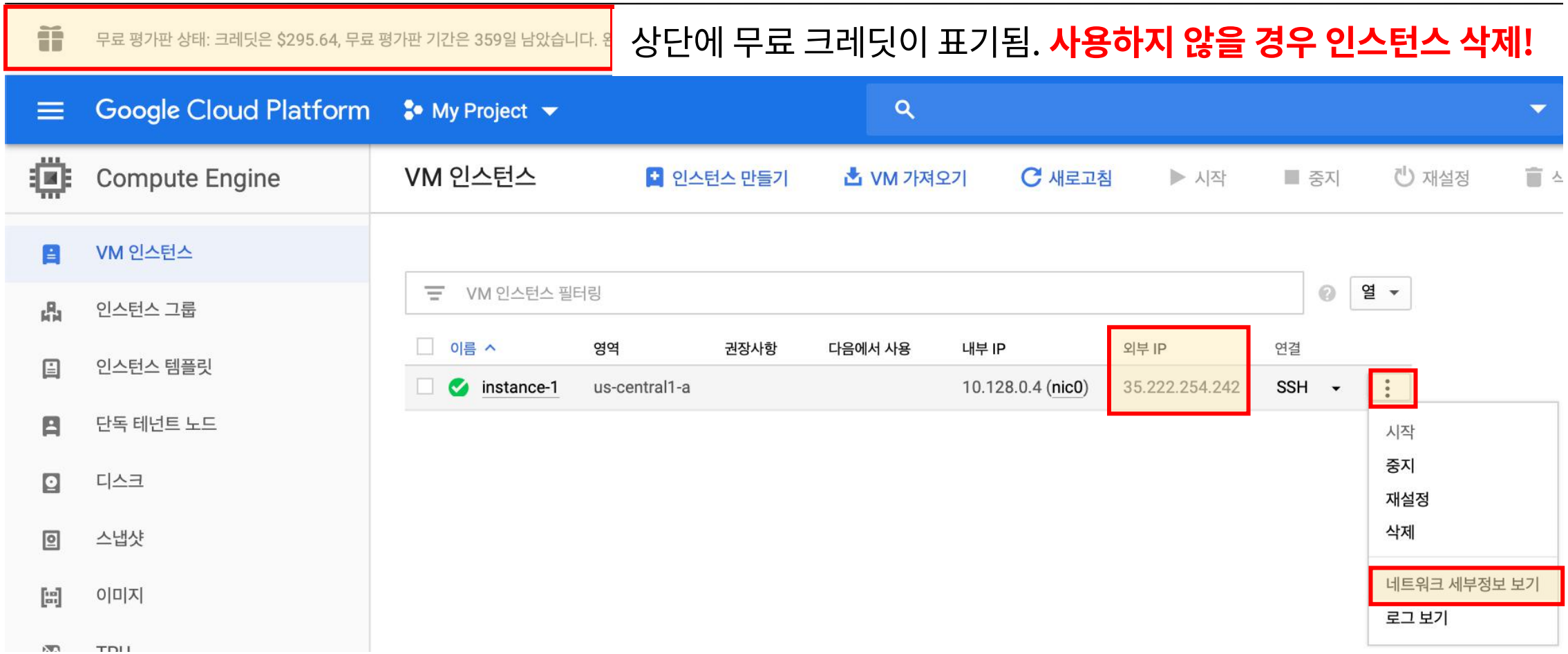

Front-End
Web
Mobile
Back-End
JSON RPC
RPC API
response
Library
Client
Admin
query
response
Network
Blockchain
Mempool
Node
Node
Transactions
Blocks
Protocols
Votes
Delegate
Baker
Endorser
Accuser

클라우드 네트워크 설정

클라우드 콘솔창으로 이동 (https://console.cloud.google.com)

외부 IP 확인 후 저장

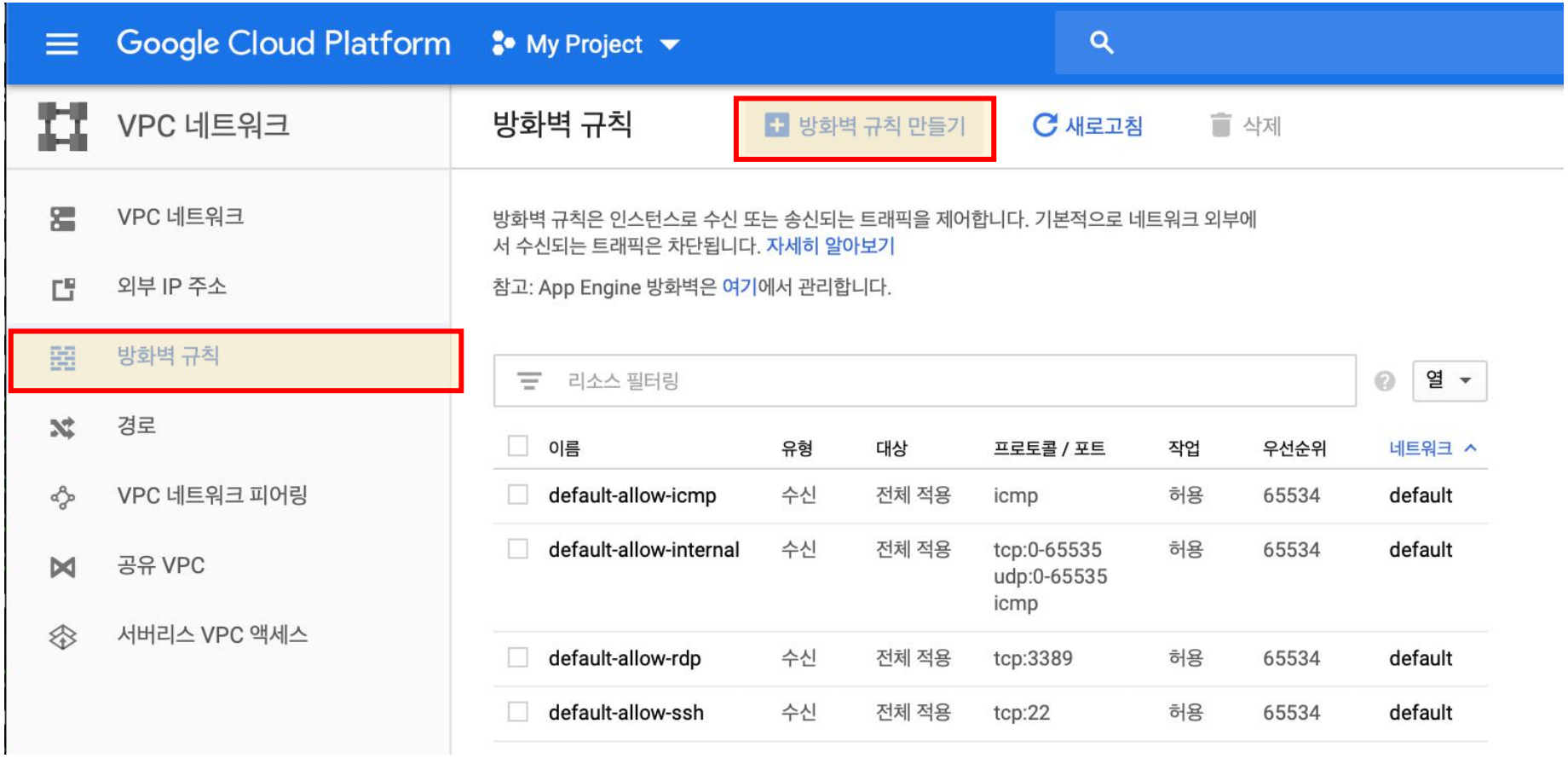
네트워크 세부정보 보기 클릭



상단에 무료 크레딧이 표기됨. **사용하지 않을 경우 인스턴스 삭제!**

클라우드 네트워크 설정

방화벽 규칙 만들기 클릭



클라우드 네트워크 설정

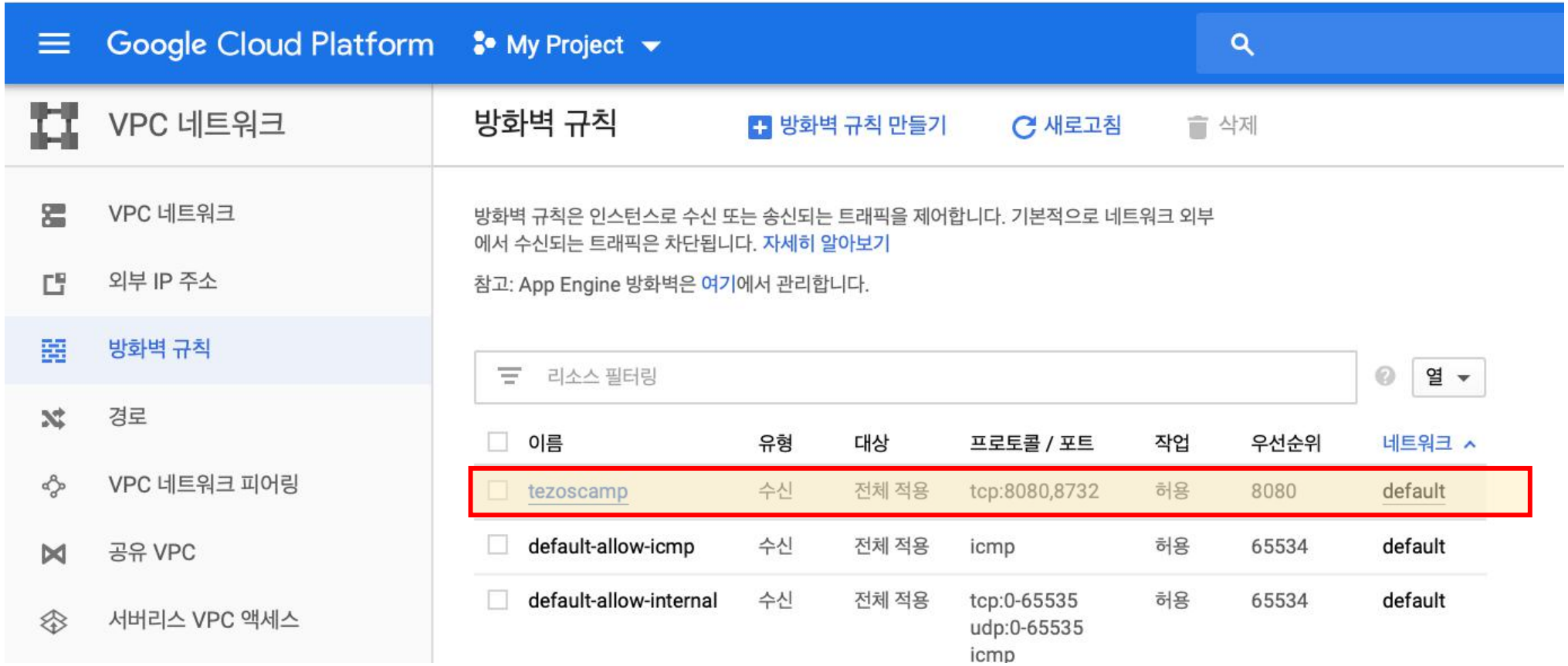
이름 tezoscamp
우선순위 8080
트래픽 방향 수신
대상 네트워크의 모든 인스턴스
소스 IP 범위 0.0.0.0/0
프로토콜 및 포트 tcp 8080, 8732

만들기 클릭

클라우드 네트워크 설정

새로운 방화벽 규칙 tezoscamp 생성 확인

tcp:8080,8732



클라우드 네트워크 설정 (ref #21)

blockchaincamp 폴더로 이동

config.json 파일 **~/.tezos-node** 로 이동

```
$ cd ~/tezos/blockchaincamp
$ cp ./config.json ~/.tezos-node/config.json
```

```
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ cd ~/tezos/blockchaincamp/
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ ls
app.js        extractKeys-linux-x64  index.html    simple.tz
config.json   eztz.min.js            reference.md
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ cat config.json
{ "rpc": { "cors-origin": ["*"], "cors-headers": ["*"] }, "p2p": { "listen-addr": "[::]:9732" } }
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ cp config.json ~/.tezos-node/config.json
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ cat ~/.tezos-node/config.json
{ "rpc": { "cors-origin": ["*"], "cors-headers": ["*"] }, "p2p": { "listen-addr": "[::]:9732" } }
```

클라우드 네트워크 설정 (ref #21)

노드 실행 중인 스크린으로 이동

노드 중지 (Ctrl + c)

노드 재실행 (**--rpc-addr 0.0.0.0:8732**)

```
$ cd ~/tezos
$ ./tezos-node run --rpc-addr 0.0.0.0:8732
```

```
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ screen -list
There are screens on:
        1609.node       (08/25/19 09:16:29)     (Detached)
4 Sockets in /run/screen/S-ubuntu.
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ screen -r node
```

```
Aug 26 12:18:19 - prevalidator.NetXgtSLGNJvN.Pt24m4xiPbLD_1: Worker terminated [NetXgtSLGNJvN.Pt2
4m4xiPbLD]
Aug 26 12:18:19 - validation_process.sequential: Shutting down...
^CAug 26 12:18:20 - node.main: Shutting down the RPC server...
Aug 26 12:18:20 - node.main: Received the INT signal, triggering shutdown.
Aug 26 12:18:20 - node.main: BYE (-6)
ubuntu@ip-172-31-28-211:~$ ^C
ubuntu@ip-172-31-28-211:~$ tezos-node run --rpc-addr 0.0.0.0:8732
Aug 26 12:18:28 - node.main: Starting the Tezos node...
Aug 26 12:18:28 - node.main: No local peer discovery.
Aug 26 12:18:28 - node.main: Peer's global id: idsoQXMNqMSJMDW8XQeFcGVDy4giSt
```

클라우드 네트워크 설정 (ref #21)

파이썬 http 서버 실행

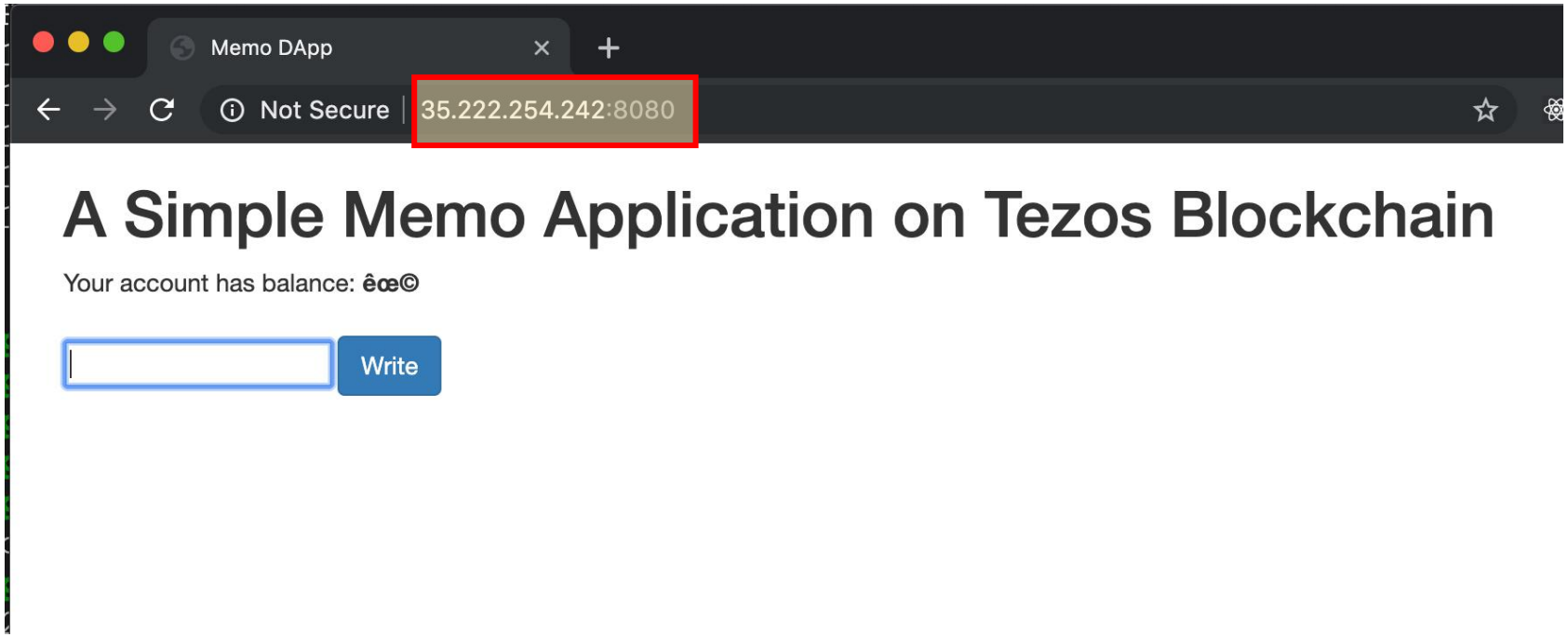
```
$ cd ~/tezos/blockchaincamp
$ sudo apt-get install python3
$ python3 -m http.server 8080
```

```
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ cd ~/tezos/blockchaincamp/
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ sudo apt-get install python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.6.7-1~18.04).
0 upgraded, 0 newly installed, 0 to remove and 33 not upgraded.
ubuntu@ip-172-31-28-211:~/tezos/blockchaincamp$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

클라우드 네트워크 설정 (ref #21)

웹 브라우저로 서버 접속 (외부 IP)

35.222.254.242:8080



eztz Library (ref #22)

Connections

rpc.tezrpc.me

local node

any node accessible

eztz.node.setProvider(url)

”http://**구글클라우드외부IP**:8732”

eztz - Javascript API library for Tezos

build failing | codecov 32%

This library is compatible with the Tezos blockchain, implementing communication with the JSON RPC API and providing key generation, signing, verification, and contract interaction. eztz.js is used by numerous projects, including TezBox and Bakechain.

You can checkout our Documentation, or follow installation below.

By default, eztz will connect to https://rpc.tezrpc.me - public Tezos infrastructure provided by TezTech. You can switch this to use your own local node, or a node of your choosing, via eztz.node.setProvider(url).

Installation

In browser, just include dist/eztz.min.js and you're good to go.

NPM plugin in development

Building

Rebuild bundle using the following code (requires webpack):

```
npm run-script build
```

Usage

Include the eztz.js file and use the eztz object directly:

```
<script src="./eztz.min.js"></script>
<script>
    eztz.rpc.getBalance("tz1LSAycAVcNdYnXCy18bwVksXci8gUC2YpA").then(function(res){
        alert("Your balance is " + res);
    }).catch(function(e){
        console.log(e);
    });
</script>
```

eztz Library (ref #22)

4가지 모듈

상황에 맞게 적절하게 이용

The purpose of eztz.js is to allow developers to easily integrate the Tezos blockchain within web and node-based apps. Inspired by ethereum's web3 library, eztz contains a very broad range of Tezos related functions. eztz can communicate with any Tezos node that exposes the RPC API, and provides the ability to generate keys, sign/verify messages, interact with contracts and more.

Checkout the readme for Installation and Usage instructions.

API Reference

The eztz library is split into 4 main parts:

- eztz.crypto - responsible for key generation, message signing and verification
- eztz.node - responsible for communication and connection with Tezos nodes
- eztz.rpc - a direct representation of the Tezos node JSON RPC API
- eztz.contract - responsible for interacting with and deploying contracts

There are also 2 additional helper parts:

- eztz.utility - a number of utility functions
- eztz.prefix - various Tezos specific prefixes to be used with key base58check encode/decode
- eztz.watermark - an object containing the various watermarks used by Tezos for marking operations
- eztz.library - an object containing all of the root libraries used by eztz

Front-end (ref #23)

키 추출*

복사 붙여 넣기 시, **줄바꿈 주의**

```
$ cd ~/tezos
$ ./tezos-client show address tezoskorea -S
```

```
mcwithimp@instance-1:~/tezos$ ./tezos-client show address tezoskorea -S
Hash: tz1LpLiNNqMTfPJpf52H558Lmqx6Xomaor1S
Public Key: edpku47Gsf34xh3mJ5bnwWAmUp72UsFsgHyaVrFBeJhvHV1De2cBV6
Secret Key: encrypted:edesk1ZBoCwpuPGhhmYoECCiCquymaZ6k5x2PQtVMSsdnT4C6skNGt2ev7e5xXDDppFW6Y4
9z6Xvj4PnsJbVSm9R
```

```
edesk1ZBoCwpuPGhhmYoECCiCquymaZ6k5x2PQtVMSsdnT4C6skN
Gt2ev7e5xXDDppFW6Y4
9z6Xvj4PnsJbVSm9R
```

ssh 창에서 복붙할 경우 줄바꿈 일어나므로

```
edesk1ZBoCwpuPGhhmYoECCiCquymaZ6k5x2PQtVMSsdnT4C6skN
Gt2ev7e5xXDDppFW6Y49z6Xvj4PnsJbVSm9R
```

edtior(atom, notepad)에 붙여넣기 한 후,
한 줄로 만들 것

* 키 추출은 오로지 Dapp의 원리를 학습하기 위한 임시방편입니다.메인넷에서는 이런 식으로 시크릿 키를 노출되도록 코드를 구현하지 않습니다. 사용자의 입력을 암호화된 형태로 받는 것은 이번 캠프의 범위를 벗어나므로 생략합니다.

Front-end (ref #23)

키 복호화

결과로 나온 pkh가 내 주소와 같은 지 확인

결과 전체 복사

```
$ cd ~/tezos/blockchaincamp
$ ./extractKeys-linux-x64
```

```
mcwithimp@instance-1:~/tezos$ cd ~/tezos/blockchaincamp/
mcwithimp@instance-1:~/tezos/blockchaincamp$ ./extractKeys-linux-x64
Enter secret_key edesk1ZBoCwpuPGhhmYoECCiCquymaZ6k5x2PQtVMSsdnT4C6skNGt2ev7e5xXDDppFW6Y49z6Xvj
4PnsJbVSm9R
Enter Please enter your password ********

{ sk:
  'edskRrYMqSJK5VeuF7JJ8xY53uj9iPq6jbKzQwcCehydrfTP6H9B6x7UPdQidfwvx51bNDkRq6AXCZpfx3sY7XR4Cve
XhQYsEU',
  pk: 'edpku47Gsf34xh3mJ5bnwWAmUp72UsFsqHyaVrFBeJhvHV1De2cBV6',
  pkh: 'tz1LpLiNNqMTfPJpf52H558Lmqx6Xomaor1S' }
```

```
mcwithimp@instance-1:~/tezos$ ./tezos-client show address tezoskorea -S
Hash: tz1LpLiNNqMTfPJpf52H558Lmqx6Xomaor1S
Public Key: edpku47Gsf34xh3mJ5bnwWAmUp72UsFsqHyaVrFBeJhvHV1De2cBV6
Secret Key: encrypted:edesk1ZBoCwpuPGhhmYoECCiCquymaZ6k5x2PQtVMSsdnT4C6skNGt2ev7e5xXDDppFW6Y4
9z6Xvj4PnsJbVSm9R
```

Front-end (ref #23)

app.js

복호화 키 복붙 (sk 줄바꿈 주의)

```
mcwithimp@instance-1:~/tezos/blockchaincamp$ ./extractKeys-linux-x64
Enter secret_key  edesk1ZBoCwpuPGhhmYoECCiCquymaZ6k5x2PQtVMSsdnT4C6skNGt2ev7e5xXDDppFW6Y49z6Xvj
4PnsJbVSm9R
Enter Please enter your password  ********

{ sk:
   'edskRrYMqSJK5VeuF7JJ8xY53uj9iPq6jbKzQwcCehydrfTP6H9B6x7UPdQidfwvx51bNDkRq6AXCZpfx3sY7XR4Cve
XhQYsEU',
  pk: 'edpku47Gsf34xh3mJ5bnwWAmUp72UsFsqHyaVrFBeJhvHV1De2cBV6',
  pkh: 'tz1LpLiNNqMTfPJpf52H558Lmqx6Xomaor1S' }
```

```
var keys ={ sk:
   'edskRrYMqSJK5VeuF7JJ8xY53uj9iPq6jbKzQwcCehydrfTP6H9B6x7UPdQidfwvx51bNDkRq6AXCZpfx3sY7XR4Cve$
  pk: 'edpku47Gsf34xh3mJ5bnwWAmUp72UsFsqHyaVrFBeJhvHV1De2cBV6',
  pkh: 'tz1LpLiNNqMTfPJpf52H558Lmqx6Xomaor1S' }
var account;
```

DISCLAIMER

키 추출은 오로지 Dapp의 원리를 학습하기 위한 임시방편입니다.
메인넷에서는 이런 식으로 시크릿 키를 노출되도록 코드를 구현하지 않습니다.
사용자의 입력을 암호화된 형태로 받는 것은 이번 캠프의 범위를 벗어나므로
생략합니다.

Front-end (ref #23)

app.js

line 9 contractAddress : simple의 주소

line 10 eztz.node.setProvider: 인스턴스 외부 IP

```
$ cd ~/tezos
$ ./tezos-client list known contracts
```

```
ubuntu@ip-172-31-28-211:~/tezos$ ./tezos-client list known contracts
simple: KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH
tezoskorea: tz1gNwQLtamC46Ac1oiC3CJXV9yfzYWXuVXD
myWallet: tz1ZpRoQUksG3uPe2yThjst5MJUXnj6329vN
```

이름	영역	권장사항	다음에서 사용	내부 IP	외부 IP	연결
instance-1	us-central1-a			10.128.0.4 (nic0)	35.222.254.242	SSH

```
8   function loadData() {
9     contractAddress = "KT1PSVXthBYGQAryRhh9CSQt4BXQuiq6vKLH";
10    eztz.node.setProvider("http://35.222.254.242:8732");
11    account = keys.pkh;
12    console.log(account);
```

Front-end (ref #23)

Memo Dapp 완성

서버 재실행

```
$ cd ~/tezos/blockchaincamp
$ python3 -m http.server 8080
```

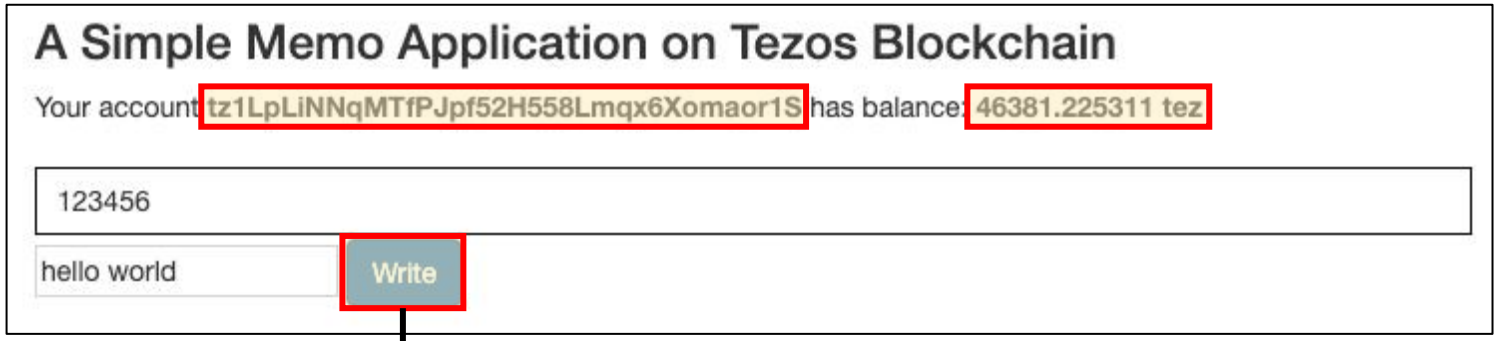
```
mcwithimp@instance-1:~/tezos/blockchaincamp$ screen -list
There are screens on:
        16670.http      (08/25/19 12:53:19)     (Detached)
        14445.node      (08/25/19 10:39:54)     (Detached)
2 Sockets in /run/screen/S-mcwithimp.
```

```
mcwithimp@instance-1:~/blockchaincamp$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
^C
Keyboard interrupt received, exiting.
mcwithimp@instance-1:~/blockchaincamp$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

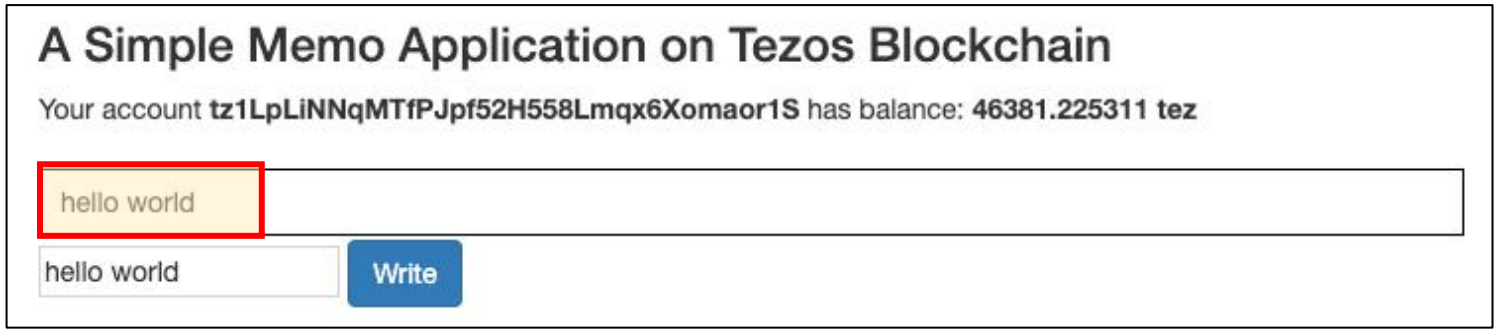
Front-end (ref #23)

Memo Dapp 완성

브라우저 hard refresh (Ctrl + Shift + F5)



약 1분 후 업데이트

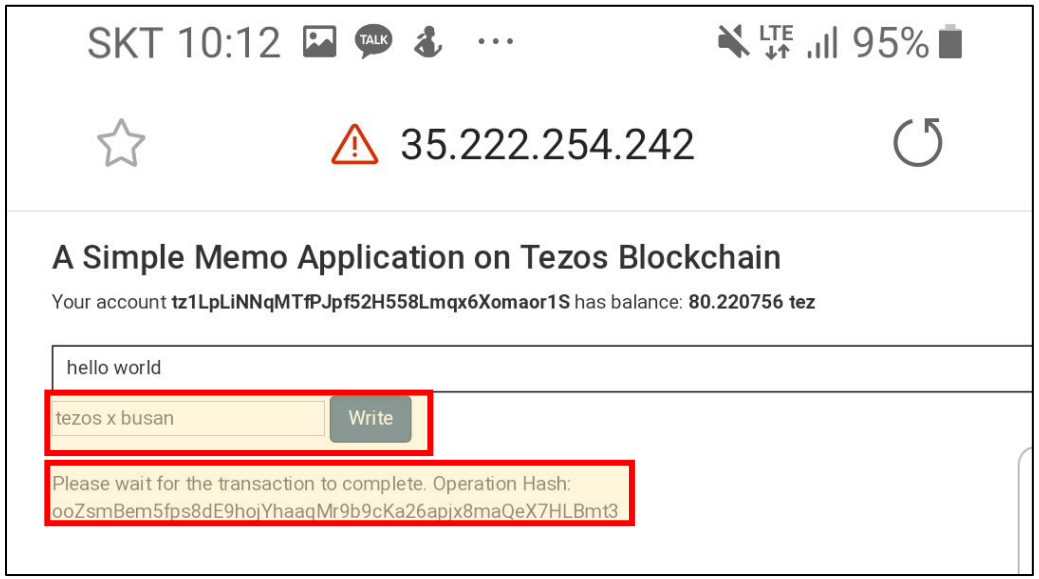


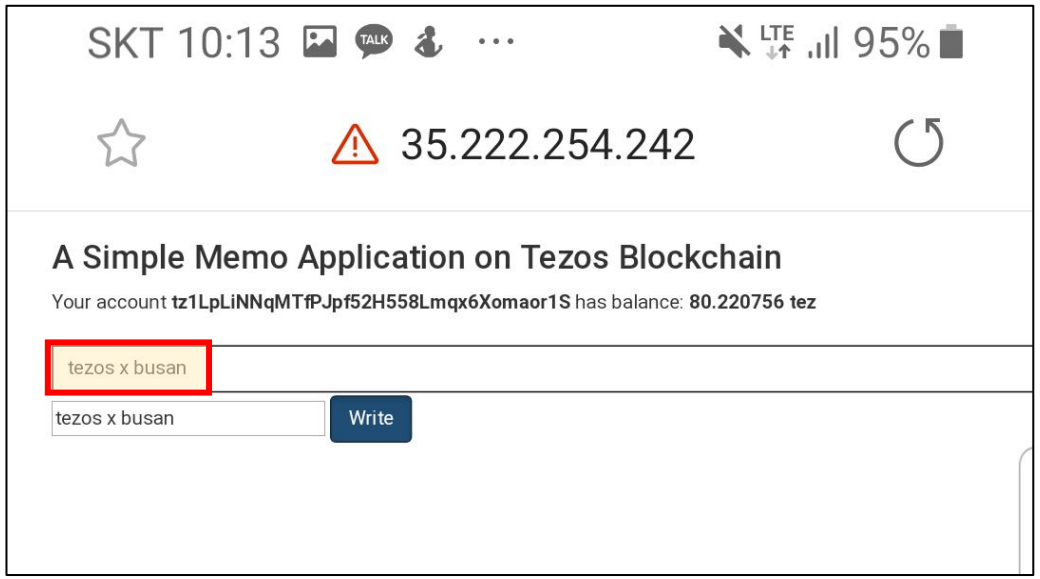
Front-end (ref #23)

Memo Dapp 완성

모바일 브라우저 (**클라우드 외부 IP:8080**)

ex. 35.222.254.242:8080





Lectures

Tezos study resources (ref #24)

컴퓨터 과학이 여는 세계 (Youtube, 책)

Khan Academy

Blockchain in Berkely

Mastering Bitcoin

Mastering Ethereum

SKKRYPTO Study Note

Blockchain Camp Seoul

Tezos Capstone

B9Lab Tezos 101

Tezos Online Class (Coming Soon)

Tezos Map

Reference sites (ref #25)

Tezos Foundation

For news of tezos ecosystem

Tezos Foundation Biannual Update (2019.08)

The recent report from Tezos Foundation

Tezos Gitlab Repo

Tezos source

Tezos Dev Docu

Tezos Stack Exchange

Questions and answers

Tezos Developer Portal

Everything you need to know when you develop something on Tezos