

KSCN 성능 테스트 - 최종정리

Created By	☞ 우석 장
Created	@2023년 9월 28일 오후 4:57
Last Edited Time	@2025년 8월 17일 오후 7:04
Last Edited By	☞ 우석 장

사전 준비

1. 테스트용 월렛 생성

Install ethereumjs-wallet

2. 테스트 계정에 KLAY 전송

계정 잠금 해제

트랜잭션을 보내고 잔액 확인하기

3. ERC-20 Token 배포

Remix Desktop IDE 설치

블록체인 노드 RPC접근 허용

ERC-20 Smart Contract

SOLIDITY COMPILER

DEPLOY & RUN TRANSACTIONS

MINT Token to Test Account

Check Token Balance

토큰 전송 대행 서버 준비

1. load-API-agent 서버정보

2. 테스트 용 Wallet 주소 및 SC 설정

3. 토큰 전송 대행 서버 실행

4. Jmeter 설정

Jmeter 설치시 유의사항

Jmeter API-master 설정

Jmeter API-master 설치

Jmeter load-API-slave설치

성능 테스트 구성

테스트 결과

블록원장 사이즈 확인

Trouble shootings

Jmeter 설정

Remote Agent 설정

사전 준비

1. 테스트용 월렛 생성

로컬PC에서 node.js 설치하여 월렛 계정 생성

Install ethereumjs-wallet

```
npm install ethereumjs-wallet --save
```

Create an index.js file and use the following code.

```
const ethWallet = require('ethereumjs-wallet');

for (let index = 0; index < 10; index++) {
  let addressData = ethWallet.default.generate();
  console.log(`Private key = ${addressData.getPrivateKeyString()}`);
  console.log(`Address = ${addressData.getAddressString()}`);
}
```

실제로는 10개의 키쌍이 생성. 이 중에서 8개만 사용

```
% node index.js
#fromAddress 또는 ownerAddress로 사용, 테스트 위해 토큰 민팅할 주소들
Private key = 0x340ef912cc86832243229891662cf935d350781708ebbc6
ea4f36a6b7acb5579
Address = 0xad5d12965b1a3b2e1739984ba977008d75d706a1
Private key = 0x652f530479d31c439a065feb12e05ec0b2052bfa7159d562
fe19f342362ad387
Address = 0x76dd095398315355dc7883121d01558e15d9906b
Private key = 0x90222628d21f2e9fa98b3339ad36da8876f11ddb17fe6f68d
b652573d9f7e028
Address = 0x082a28590b7aed26d718d383c539e8969bdb139d
Private key = 0x1585a0ab19e9734f1caf03cdaf121ba2b7dc859a3d5cecd160
ea40453390688a
```

```
Address = 0xf7facb39d88d9ddaf991431fae8aa6ebf433d6c0
```

toAddress 테스트시 토큰을 받을 주소

```
Private key = 0x01bb17ed79262b40b0b1e3038621bd982a5660042c6aee0  
144726142653282b5
```

```
Address = 0x6f1d7941d5dcdcc1957565c1a59b91ce89ee4d07
```

```
Private key = 0x2f822ff3121814e93a19949402884ffa27d854299f16b2d9c  
0e9965c453f744
```

```
Address = 0x00d5b3c727fae3b53153c0ffc3e8c938215abd53
```

```
Private key = 0xade5f394548344b70c41e0c7fe26981d1c129f0d4f8b1bb4f  
d313a0baa37fbbb
```

```
Address = 0x323c377c54791b27e05bc242410bf034cea3ab7e
```

```
Private key = 0x905d65e0bd35cda8f822f3371a524746feb60e8f2b2d5ad1  
50c841319647215d
```

```
Address = 0xffee6d510f783c38b885c03ae416230ca19f4576
```

2. 테스트 계정에 KLAY 전송

계정 잠금 해제

#1번 노드에 접속하여 실행. 최초에 key import 진행

```
$ kscn account import --datadir ~/klaytn/data ~/klaytn/homi-output/keys_t  
est/testkey1
```

Your new account is locked with a password. Please give a password. Do not forget this password.

Passphrase: #패스워드 : et#\$dgghDgsd

Repeat passphrase:

Address: {b8dd4f3368545a25c0994e1844e1a3bb81a5a928}

Your account is imported at /root/klaytn/data/keystore/UTC--2023-09-28T08-25-15.442240332Z--b8dd4f3368545a25c0994e1844e1a3bb81a5a928

#1번 노드에 접속하여 실행. 이미 key를 import 하였다면 명령어로 확인

```
# ls /root/klaytn/data/keystore
```

```
UTC--2023-09-28T08-25-15.442240332Z--b8dd4f3368545a2c0994e184  
4e1a3bb81a5a928
```

```
#1번 노드에 접속하여 실행. import한 계정 unlock 진행
$ kscn attach --datadir ~/klaytn/data
> personal.unlockAccount("b8dd4f3368545a25c0994e1844e1a3bb81a5a928")
Unlock account b8dd4f3368545a25c0994e1844e1a3bb81a5a928
Passphrase:
true
```

트랜잭션을 보내고 잔액 확인하기

```
#테스트 할 계정으로 각각 coin을 보내준다.
```

```
> klay.sendTransaction({from: "b8dd4f3368545a25c0994e1844e1a3bb81a  
5a928", to: "ad5d12965b1a3b2e1739984ba977008d75d706a1", value: 1000  
000000000000000000000000})  
"0xfce2193116e9cb4d9b63b39770fb49a400c365dc4219a1262a8e87a3f34  
485b4"  
> klay.getBalance("ad5d12965b1a3b2e1739984ba977008d75d706a1")  
1000000000000
```

```
> klay.sendTransaction({from: "b8dd4f3368545a25c0994e1844e1a3bb81a5a928", to: "76dd095398315355dc7883121d01558e15d9906b", value: 1000000000000000000000000})
> klay.sendTransaction({from: "b8dd4f3368545a25c0994e1844e1a3bb81a5a928", to: "082a28590b7aed26d718d383c539e8969bdb139d", value: 10000000000000000000000000000000})
> klay.sendTransaction({from: "b8dd4f3368545a25c0994e1844e1a3bb81a5a928", to: "f7facb39d88d9ddaf991431fae8aa6ebf433d6c0", value: 10000000000000000000000000000000})
> klay.getBalance("f7facb39d88d9ddaf991431fae8aa6ebf433d6c0")
1000000000000
```

#메타마스크 계정으로도 (필요시)

```
> klay.sendTransaction({from: "b8dd4f3368545a25c0994e1844e1a3bb81a5a928", to: "ED01EF69a53d0ceE4357859d9e11420d2CF57941", value: 100
```

```
000000000000000000000000})  
0xED01EF69a53d0ceE4357859d9e11420d2CF57941
```

3. ERC-20 Token 배포

Remix Desktop IDE 설치

<https://github.com/ethereum/remix-desktop/releases>

블록체인 노드 RPC접근 허용

```
# 1번 노드에서 RPC 통신위한 8551번 포트 오픈
sudo firewall-cmd --permanent --zone=public --add-port=8551/tcp
sudo firewall-cmd --reload
```

ERC-20 Smart Contract

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol";
import "@openzeppelin/contracts/security/Pausable.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyToken is ERC20, ERC20Burnable, Pausable, Ownable {
    constructor() ERC20("MyToken", "MTK") {}

    function pause() public onlyOwner {
        _pause();
    }

    function unpause() public onlyOwner {
        _unpause();
    }
}
```

```

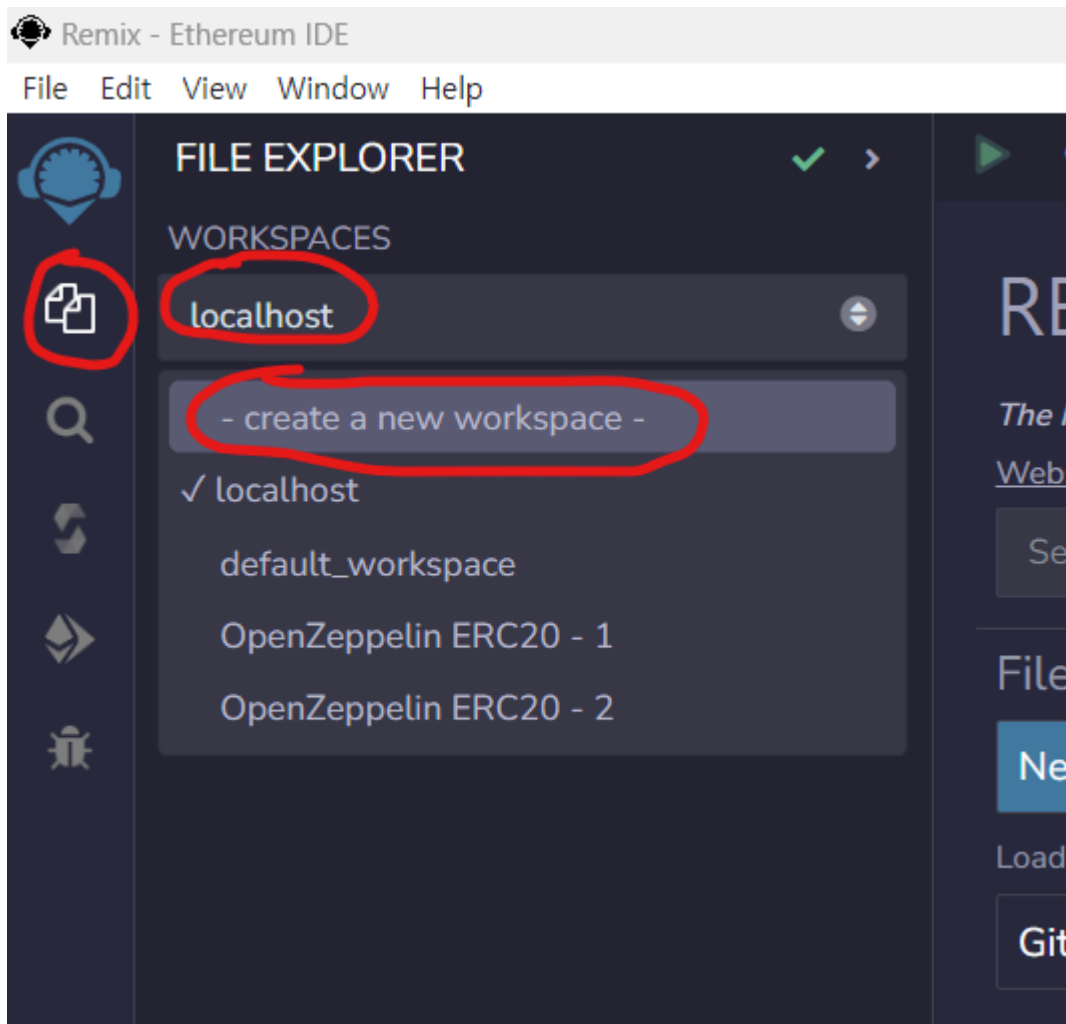
    }

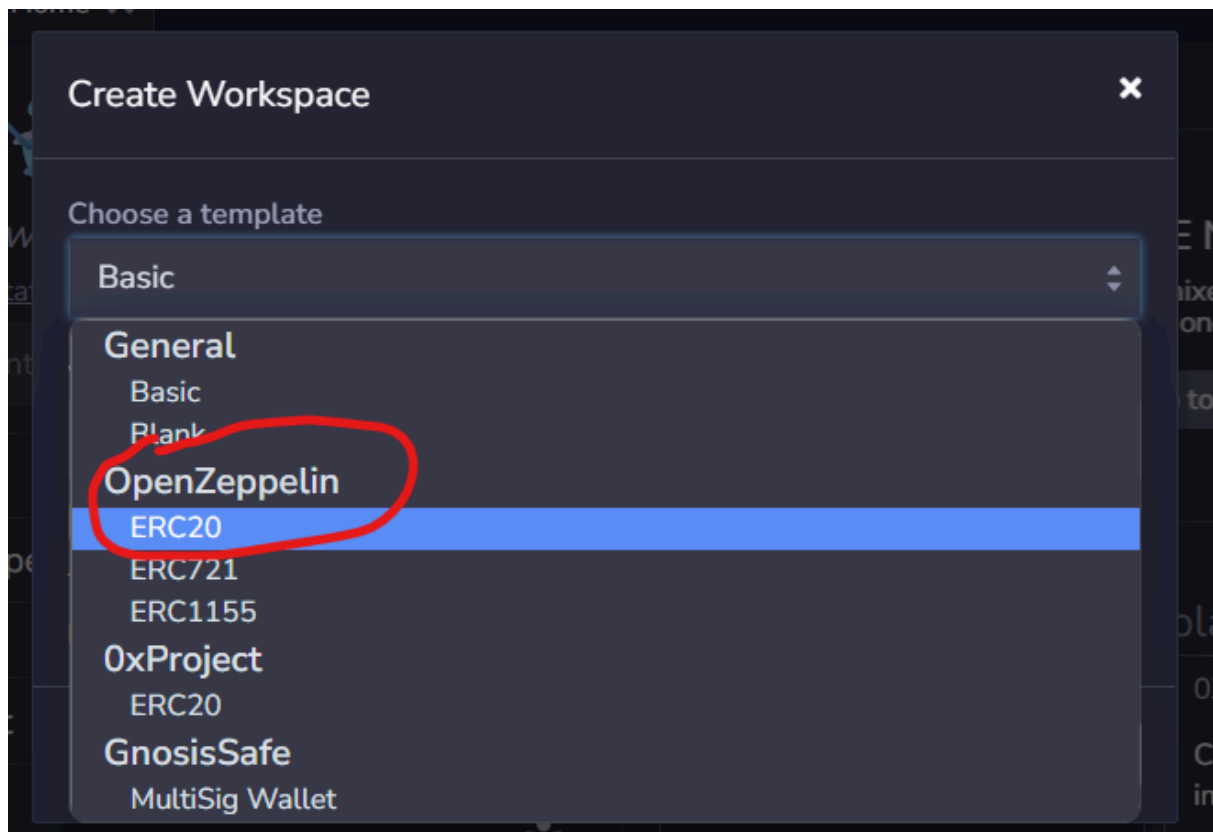
    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }

    function _beforeTokenTransfer(address from, address to, uint256 amount)
    internal
    whenNotPaused
    override
    {
        super._beforeTokenTransfer(from, to, amount);
    }
}

```

또는 ERC20 토큰 컨트랙트 템플릿 사용





Create Workspace

Choose a template

ERC20

Customize template

Features

☒ Mintable

☒ Burnable

☒ Pausable

Upgradeability

☐ Transparent

☐ UUPS

Workspace name

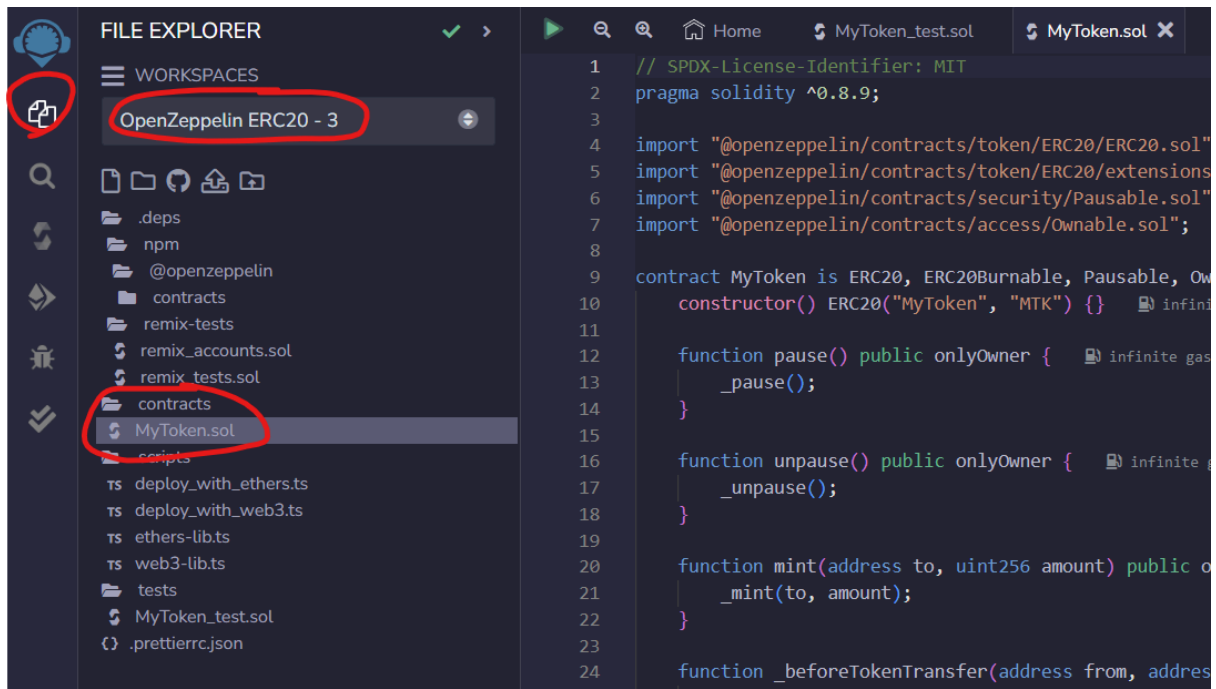
OpenZeppelin ERC20 - 3

☐ Initialize workspace as a new git repository

To use Git features, add username and email to the Github section of the Settings panel.

OK

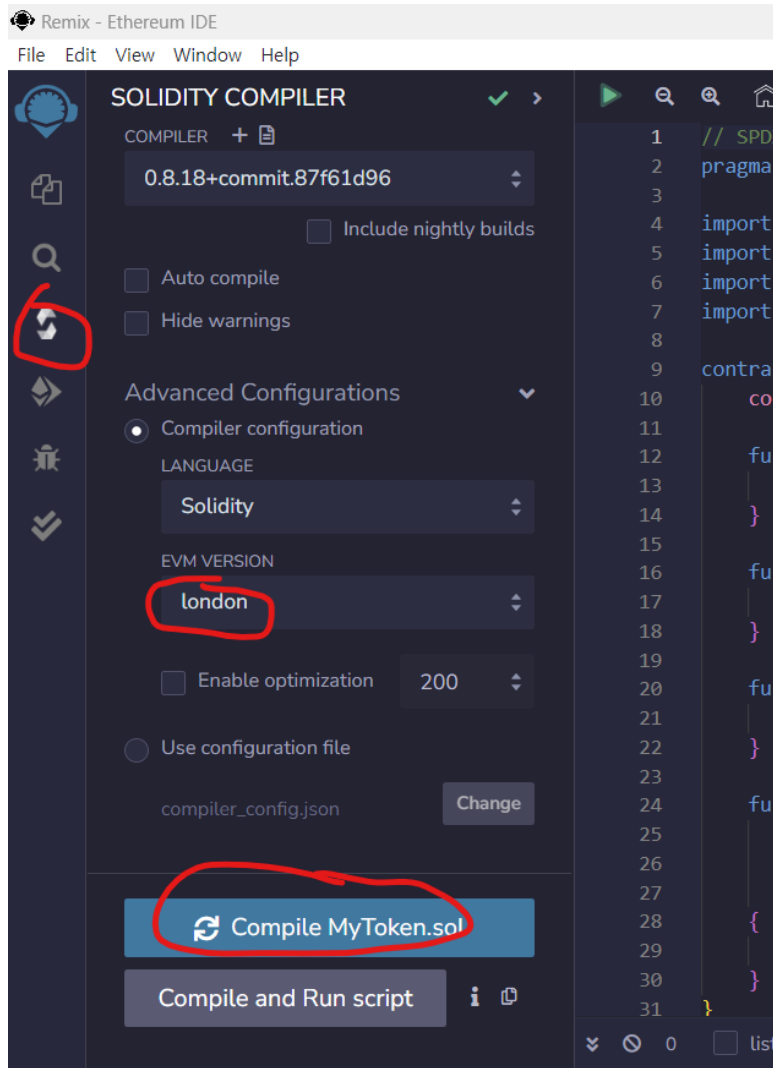
Cancel



생성한 ERC20 token 코드 확인

SOLIDITY COMPILER

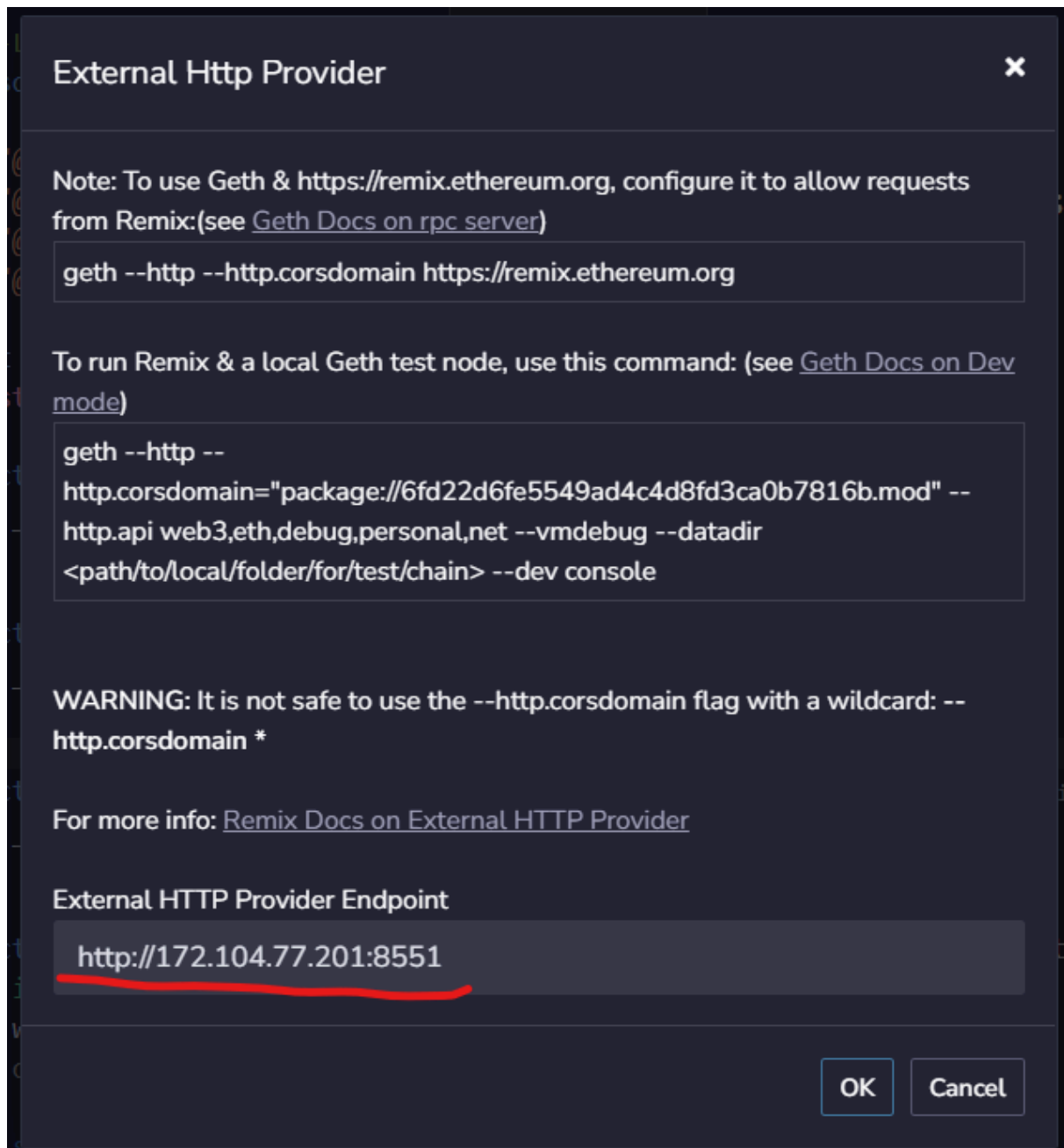
EVM VERSION은 london을 선택합니다.



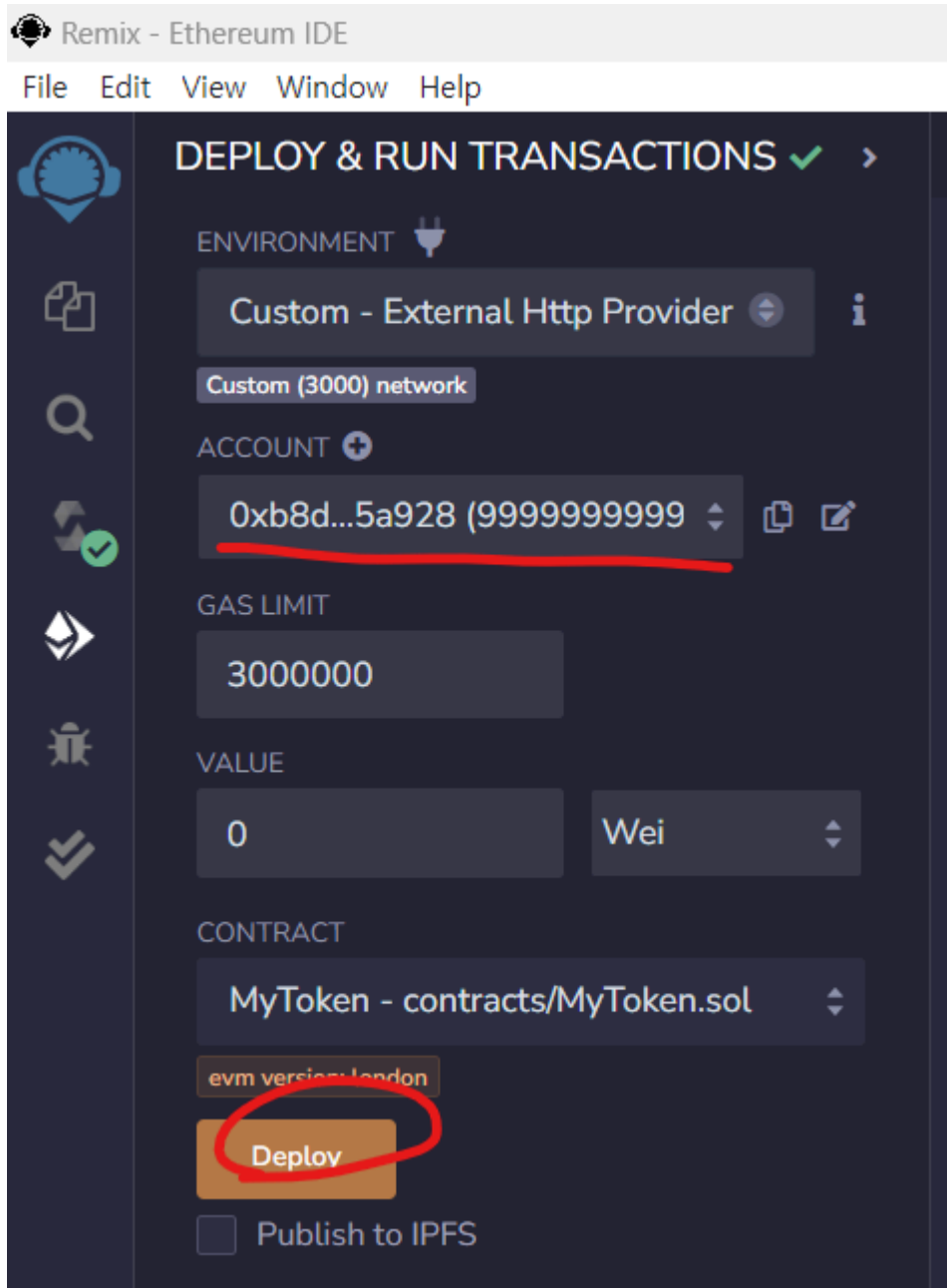
DEPLOY & RUN TRANSACTIONS

ENVIRONMENT → Custom External Http Provider : <http://172.104.77.201:8551> 입력

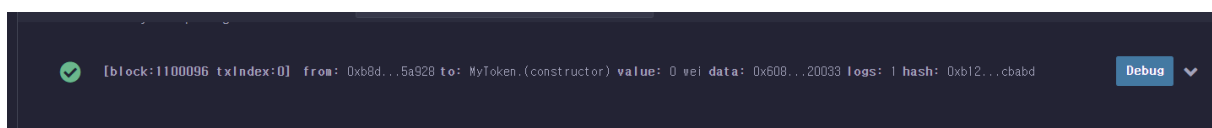
```
# 배포전에 1번 노드에 접속하여 사용블록체인 계정의 잠금해제를 진행
ssh root@172.104.77.201
$ kscn attach --datadir ~/klaytn/data
> personal.unlockAccount("b8dd4f3368545a25c0994e1844e1a3bb81a5a928")
Unlock account b8dd4f3368545a25c0994e1844e1a3bb81a5a928
Passphrase: #패스워드 : et#$dgghDgsd
true
```



변경된 민팅에 사용할 지갑주소 확인 후 Deploy



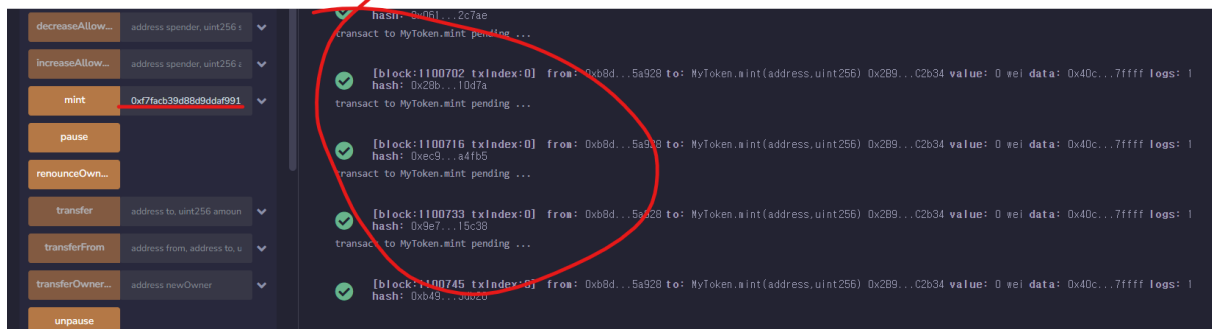
정상배포 여부 터미널 메시지 확인



스마트 컨트랙트 주소 복사 : 0x2B92Aa2c97DeD29834F3437Dd11aE8aa1b7C2b34
0x75094C8998535d2023FE573c509Cd97963650C84 (다시)

MINT Token to Test Account

전송 테스트 진행할 계정 4개에 토큰 민팅 진행



민팅 이전에 1번 노드에 접속하여 사용블록체인 계정의 잠금해제를 진행

```
ssh root@172.104.77.201
```

```
$ kscn attach --datadir ~/klaytn/data
```

```
> personal.unlockAccount("b8dd4f3368545a25c0994e1844e1a3bb81a5a928")
```

Unlock account b8dd4f3368545a25c0994e1844e1a3bb81a5a928

Passphrase: #패스워드 : et#\$dgghDgsd

true

keystore 계정에 민팅

```
0xb8dd4f3368545a25c0994e1844e1a3bb81a5a928,999999999999999999
```

Metamask 계정에 민팅, 갯수 확인용

```
0x36F5A9A81591D66857b91E4FB903d7E67e89e4C3,999999999999999999
```

토큰전송 from계정에 민팅

```
0xad5d12965b1a3b2e1739984ba977008d75d706a1,999999999999999999
```

```
0x76dd095398315355dc7883121d01558e15d9906b,999999999999999999
```

```
0x082a28590b7aed26d718d383c539e8969bdb139d,999999999999999999
```

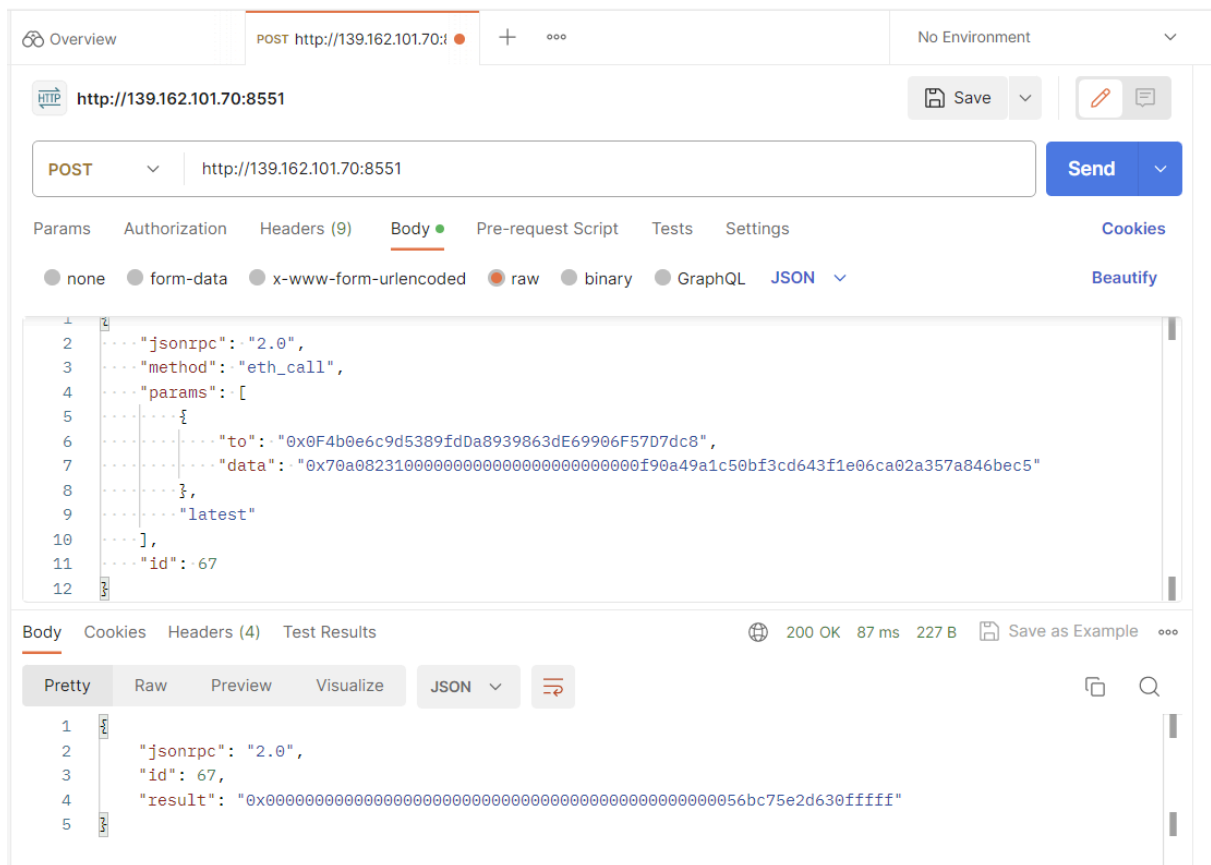
```
0xf7facb39d88d9ddaf991431fae8aa6ebf433d6c0,999999999999999999
999999999999
```

Check Token Balance

테스트 계정으로 민팅한 토큰의 밸런스를 확인 합니다.

[illegible]

- 포스트맨 또는 커맨드 창에서도 확인 가능



토큰 전송 대행 서버 준비

API-master설정 (동일 도메인).

1. load-API-agent 서버정보

#centOS7, shared 2CPU, Linode 4GB 플랜

#mid-API-agent1: ssh root@172.104.66.212 / 192.168.150.247 / et#\$dgghDg
sd

#mid-API-agent2: ssh root@172.105.198.45 / 192.168.150.234 / et#\$dgghDg
sd


```
#mid-API-agent3: ssh root@172.105.198.64 / 192.168.136.86 / et#$dgghDgs  
d
```

load-api-server.zip

2. 테스트 용 Wallet 주소 및 SC 설정

첨부 압축 파일의 app.service.ts 파일의 아래 항목을 API 서버 별로 다르게 설정합니다.

```
# toAddress(받을주소) / tokenAddress(컨트렉주소) / ownerAddress(보내는주소) / ownerPri(프라이빗키)  
# tokoenAddress 부분 업데이트 할것  
  
# rpc url은 mid-KSCN의 각 노드 주소  
  
# chainID 변경이 있는 경우 수정  
# chainId: +BigInteger.from(1000).toHexString(),  
# → chainId: +BigInteger.from(1027).toHexString(),  
  
# 계속적인 트래픽 발생에 따라 가스비 증가로 인해 수정  
# 기존 gasPrice: BigInteger.from(250000000000).toHexString(),  
# 수정 gasPrice: BigInteger.from(25000000000000).toHexString(),  
  
# .env  
  
# 하기 설정 부분은 하드코딩에서 환경변수 (.env)를 통해 설정토록 변경됨 (2023.10.3)  
  
#load-api-server-1, rpc url : http://172.104.77.201:8551 #내부IP를 넣으면 더 빠를듯  
private toAddress = '0x6f1d7941d5dcdcc1957565c1a59b91ce89ee4d07';  
private tokenAddress = '0x75094C8998535d2023FE573c509Cd97963650C84';  
private ownerAddress = '0xad5d12965b1a3b2e1739984ba977008d75d706'
```

```

a1';
private ownerPrivateKey =
    '340ef912cc86832243229891662cf935d350781708ebbc6ea4f36a6b7a
cb5579';
private url = 'http://172.104.77.201:8551';

#load-api-server-2, rpc url : http://172.105.237.141:8551
private toAddress = '0x6f1d7941d5dcdcc1957565c1a59b91ce89ee4d07';
private tokenAddress = '0x75094C8998535d2023FE573c509Cd97963650
C84';
private ownerAddress = '0x76dd095398315355dc7883121d01558e15d990
6b';
private ownerPrivateKey =
    '652f530479d31c439a065feb12e05ec0b2052bfa7159d562fe19f342362
ad387';
private url = 'http://172.104.77.201:8551';

#load-api-server-3, rpc url : http://172.104.66.43:8551
private toAddress = '0x6f1d7941d5dcdcc1957565c1a59b91ce89ee4d07';
private tokenAddress = '0x75094C8998535d2023FE573c509Cd97963650
C84';
private ownerAddress = '0x082a28590b7aed26d718d383c539e8969bdb13
9d';
private ownerPrivateKey =
    '90222628d21f2e9fa98b3339ad36da8876f11ddb17fe6f68db652573d9f7
e028';
private url = 'http://172.104.77.201:8551';

#load-api-server-4, rpc url : http://172.104.78.15:8551
private toAddress = '0xffee6d510f783c38b885c03ae416230ca19f4576';
private tokenAddress = '0x75094C8998535d2023FE573c509Cd97963650
C84';
private ownerAddress = '0xf7facb39d88d9ddaf991431fae8aa6ebf433d6c
0';
private ownerPrivateKey =
    '1585a0ab19e9734f1caf03cdaf121ba2b7dc859a3d5cecd160ea40453390
688a';
private url = 'http://172.104.78.15:8551';

```

- 각 노드로 복사

```
#윈도우 로컬에서 각 load-API 서버로 복사. 패스워드 : et#$dgghDgsd
```

```
scp -r load-api-server/ root@172.104.66.212:~/load-api-server/
scp -r load-api-server/ root@172.105.198.45:~/load-api-server/
scp -r load-api-server/ root@172.105.198.64:~/load-api-server/
scp -r load-api-server/ root@Load-API-IP:~/load-api-server/
```

```
# 이후 각 서버에서 .env.1 .env.2 .env.3 환경변수를 등록하고 실행
$ export $(xargs < .env.1)
$ export $(xargs < .env.2)
$ export $(xargs < .env.3)
```

3. 토큰 전송 대행 서버 실행

```
# mid-API-agent1: ssh root@172.104.66.212 / 192.168.150.247 / et#$dgghDgsd
# mid-API-agent2: ssh root@172.105.198.45 / 192.168.150.234 / et#$dgghDgsd
# mid-API-agent3: ssh root@172.105.198.64 / 192.168.136.86 / et#$dgghDgsd
```

API_agent_node1 서버셋업

```
ssh root@172.104.66.212 / error during JRMP connection establishment
```

```
#sudo firewall-cmd --permanent --zone=public --add-port=8551/tcp
sudo firewall-cmd --permanent --zone=public --add-port=3000/tcp
sudo firewall-cmd --permanent --zone=public --add-port=4000/tcp
sudo firewall-cmd --permanent --zone=public --add-port=1099/tcp
sudo firewall-cmd --reload
nmap -p 22,30000,8551,3000,1099,4000 172.104.66.212
nmap -p 22,30000,8551,3000,1099,4000 172.105.198.45
...
firewall-cmd --list-all
```

```
#열링포트 확인
netstat -tnlp
#명령어가 안되면, yum install net-tools

# epel repository 추가
$ yum install epel-release

# npm 과 nodejs 설치
$ yum install -y npm nodejs

# 버전 확인
$ node -v
$ npm -v

# 패키지 설치 및 실행 /load-api-server-x 폴더에서 실행하면 됨
$ npm install
$ npm run start

# 세션 끊겨도 계속 실행
$ nohup npm run start &

# 세션 연결 중에만 계속 실행
npm run start &

# 종료
ps auxf | grep node // node를 실행중인 프로세스 검색
kill -9 프로세스ID // 위에서 찾은 ID를 이용해 프로세스 종료

# epel repository 추가
$ yum install epel-release

# npm 과 nodejs 설치
$ yum install -y npm nodejs

# 버전 확인
$ node -v
$ npm -v
```

패키지 설치 및 실행 /load-api-server-x 폴더에서 실행하면 됨

\$ npm install

\$ npm run start

세션 끊어도 계속 실행

\$ nohup npm run start &

세션 연결 중에만 계속 실행

npm run start &

종료

ps auxf | grep node // node를 실행중인 프로세스 검색

kill -9 프로세스ID // 위에서 찾은 ID를 이용해 프로세스 종료

실행순서

1. 대행서버 실행

- load-api-server 폴더에서 npm run start &

2. apache-jmeter-5.6.2/bin 폴더에서 ./jmeter-server &

```
Complete!
[root@172-104-66-212 ~]# node -v
v16.18.1
[root@172-104-66-212 ~]# npm -v
8.19.2
[root@172-104-66-212 ~]#
```

```
Complete!
[root@139-162-120-37 load-api-server-1]# npm run start

> api-server@0.0.1 start
> nest start

[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [NestFactory] Starting Nest application...
[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [InstanceLoader] HttpModule dependencies initialized +37ms
[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [InstanceLoader] AppModule dependencies initialized +94ms
[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [RoutesResolver] AppController {/}: +12ms
[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [RouterExplorer] Mapped {/, GET} route +4ms
[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [RouterExplorer] Mapped {/sendTx, GET} route +0ms
[Nest] 21247 - 09/20/2023, 11:04:11 AM LOG [NestApplication] Nest application successfully started +3ms
```

```
[root@139-162-120-37 bin]# ./jmeter-server &WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Using local port: 1099
lsCreated remote object: UnicastServerRef2 [liveRef: [endpoint:[139.162.120.37:1099](local),objID:[5ccd0eba:18ac1a5b5bb:-7fff, 4251027571360393844]]]
ls
ApacheJMeter.jar      heapdump.cmd      jmeter-server      mirror-server      stoptest.sh
BeanShellAssertion.bshrc heapdump.sh        jmeter-server.bat  mirror-server.cmd  system.properties
BeanShellFunction.bshrc jaas.conf          jmeter-server.log  mirror-server.sh    templates
BeanShellListeners.bshrc jmeter            jmeter.sh          reportgenerator.properties threaddump.cmd
BeanShellSampler.bshrc  jmeter.bat         jmeter-t.cmd       report-template      threaddump.sh
create-rmi-keystore.bat jmeter.log         jmeterw.cmd        saveservice.properties upgrade.properties
create-rmi-keystore.sh  jmeter-n.cmd       Klaytn_RPC.jmx     shutdown.cmd         user.properties
examples               jmeter-n-r.cmd     krb5.conf           shutdown.sh          utility.groovy
nc.parameters          jmeter.properties log4j2.xml          stoptest.cmd
[root@139-162-120-37 bin]# curl --location 'http://localhost:3000/sendTx'
insufficient funds of the sender for value
insufficient funds of the sender for value
insufficient funds of the sender for value [root@139-162-120-37 bin]#
```

api서버 app.service.ts에서 chainID 변경 1000 → 1027

curl --location 'http://localhost:3000/sendTx'

4. Jmeter 설정

Jmeter 설치시 유의사항

Master(Client) - Slave(Server) 모두 같은 subnet에 존재해야 한다!

Jmeter API-master 설정

우분투 GUI환경 구축하여 진행

- ubuntu

#Ubuntu 20.04 설치방법 정리

#API서버정보 Linode dedicated 8cpu, 16GB | Ubuntu 20.04 LTS

ssh root@172.104.81.230 / 0x76dd095398315355dc7883121d01558e15d9906b

#xrdp 설치

sudo apt update

sudo apt install --no-install-recommends xrdp

#서비스 실행상태 확인

systemctl status xrdp

원격접속위함 포트 3389 오픈, jmeter RMI 포트 1099 오픈

```
sudo ufw allow 3389/tcp
sudo ufw allow 1099/tcp
#sudo ufw allow 8551/tcp
#sudo apt install ufw
#sudo ufw status verbose
#sudo ufw app list
```

```
# xorgxrdp 설치
sudo apt install xorgxrdp
```

```
# xfce 설치
sudo apt install xfce4
설치중에 의존성 프로그램 목록에서 lightdm 선택
```

```
# xrdp를 xfce4의 GUI로 연결
echo xfce4-session > ~/.xsession
```

```
# xserver-xorg-input-all 설치, 키보드와 마우스
sudo apt install xserver-xorg-input-all
```

```
# xrdp서비스 재시작
sudo service xrdp restart
```

#이후 윈도우 원격접속 하여 사용

<https://www.bearpooh.com/98>

아래는 CentOS7 에서 설정, 그냥 참고. 잘 안됨

```
#CentOS GUI 설치방법 정리
#CentOS 설치방법 정리
yum update
yum group list
sudo yum groupinstall "GNOME Desktop" "Graphical Administration Tools"
```

#구동환경확인

systemctl get-default

#multi-user: 텍스트모드 적용

#graphical: 그래픽모드

#graphical 모드로 변환

systemctl set-default graphical.target

#원격데스크톱 설치 - EPEL저장소를 추가

yum install epel-release

#xrdp와 tigervnc-server설치

yum install xrdp tigervnc-server

#방화벽오픈 - 원격데스크톱 3389번

sudo firewall-cmd --permanent --zone=public --add-port=3389/tcp

sudo firewall-cmd --permanent --zone=public --add-port=8551/tcp

sudo firewall-cmd --permanent --zone=public --add-port=3000/tcp

sudo firewall-cmd --permanent --zone=public --add-port=4000/tcp

sudo firewall-cmd --permanent --zone=public --add-port=1099/tcp

sudo firewall-cmd --reload

#xrdp 서비스 시작

systemctl start xrdp

#재부팅 후 xrdp자동시작

systemctl enable xrdp

#이후 윈도우에서 원격데스크톱 으로 연결

#자바 JDK설치 : <https://www.java.com/en/download/>

#윈도우 : <https://dlcdn.apache.org/jmeter/binaries/apache-jmeter-5.6.2.zip>

#리눅스 : <https://dlcdn.apache.org/jmeter/binaries/apache-jmeter-5.6.2.tgz>

#윈도우 리눅스 버전을 맞춰야


```
yum list java*jdk-devel
yum install -y java-11-openjdk-devel.x86_64
yum install -y java-1.8.0-openjdk-devel.x86_64

mkdir -p ~/jmeter562
cd ~/jmeter562

wget -d https://dlcdn.apache.org/jmeter/binaries/apache-jmeter-5.6.2.tgz
tar -zxvf apache-jmeter-5.6.2.tgz

# Jmeter 실행 파일이 있는 경로로 이동
cd ~/jmeter562/apache-jmeter-5.6.2/bin

# jmeter 실행
./jmeter
```

Jmeter API-master 설치

- Java 1.8.0 버전 설치
- Jmeter 5.6.2버전 설치 및 jmeter-plugins-manager 설치

```
#자바 JDK설치 : https://www.java.com/en/download/
#윈도우 : https://dlcdn.apache.org/jmeter/binaries/apache-jmeter-5.6.2.zip
#리눅스 : https://dlcdn.apache.org/jmeter/binaries/apache-jmeter-5.6.2.tgz
#윈도우 리눅스 버전을 맞춰야

#java 1.8.0_382 설치
sudo apt install openjdk-8-jdk
java -version

wget -d https://dlcdn.apache.org/jmeter/binaries/apache-jmeter-5.6.2.tgz
tar -zxvf apache-jmeter-5.6.2.tgz

# Jmeter 실행 파일이 있는 경로로 이동
cd ~/apache-jmeter-5.6.2/bin
vim jmeter.properties
```

```
#-----
# Remote hosts and RMI configuration
#-----

# Remote Hosts - load-api-server들을 같은 subnet에 만들고 콤마로 분리하여 IP
# 넣어준다.
remote_hosts=192.168.x.x
#remote_hosts=localhost:1099,localhost:2010
...
server_port=1099
client.rmi.localport=1099
server.rmi.localport=1099
server.rmi.ssl.disable=true
...

nmap -p 22,30000,8551,3000,1099,4000 192.168.187.223

# jmeter 실행
./jmeter

# https://jmeter-plugins.org/downloads/all/ 에서 plugin 다운로드 후
scp -r jmeter-plugins-manager-1.9.jar root@172.104.81.230:~/apache-jmete
r-5.6.2/lib/ext/
scp -r Klaytn_RPC.jmx root@172.104.81.230:~/apache-jmeter-5.6.2/
```

Klaytn_RPC.jmx

Jmeter load-API-slave설치

Apache JMeter

We recommend you use a mirror to download our release builds, but you must verify the integrity of the downloaded files using signatures downloaded from our main

 https://jmeter.apache.org/download_jmeter.cgi

```
#자바 JDK설치 : https://www.java.com/en/download/  
#윈도우 : https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.2.zip  
#리눅스 : https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.2.tgz  
#윈도우 리눅스 버전을 맞춰야
```

```
yum list java*jdk-devel  
#yum install -y java-11-openjdk-devel.x86_64  
yum install -y java-1.8.0-openjdk-devel.x86_64
```

```
#mkdir -p ~/jmeter562  
#cd ~/jmeter562
```

```
wget -d https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.2.tgz  
tar -zxvf apache-jmeter-5.6.2.tgz
```

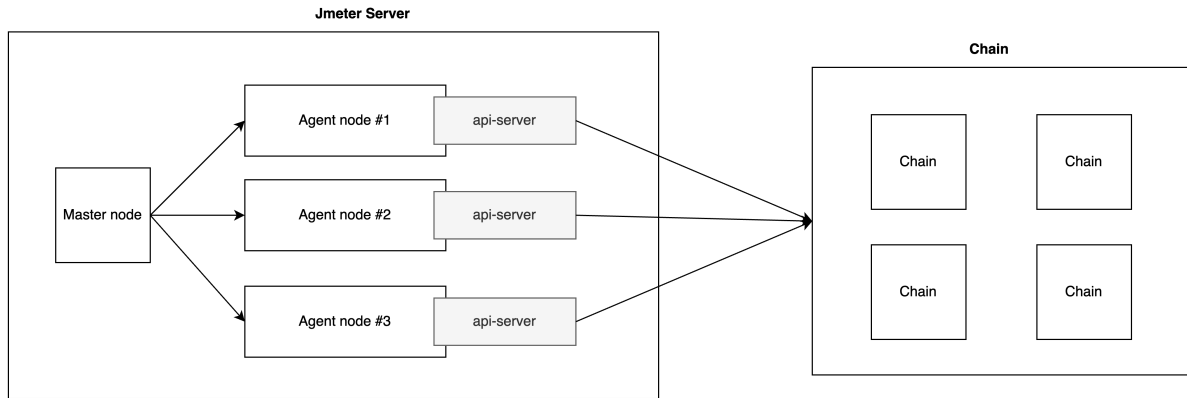
```
# Jmeter 실행 파일이 있는 경로로 이동  
cd ~/apache-jmeter-5.6.2/bin
```

```
vim jmeter.properties  
server_port=1099  
client.rmi.localport=1099  
server.rmi.port=1099  
server.rmi.localport=1099  
server.rmi.ssl.disable=true
```

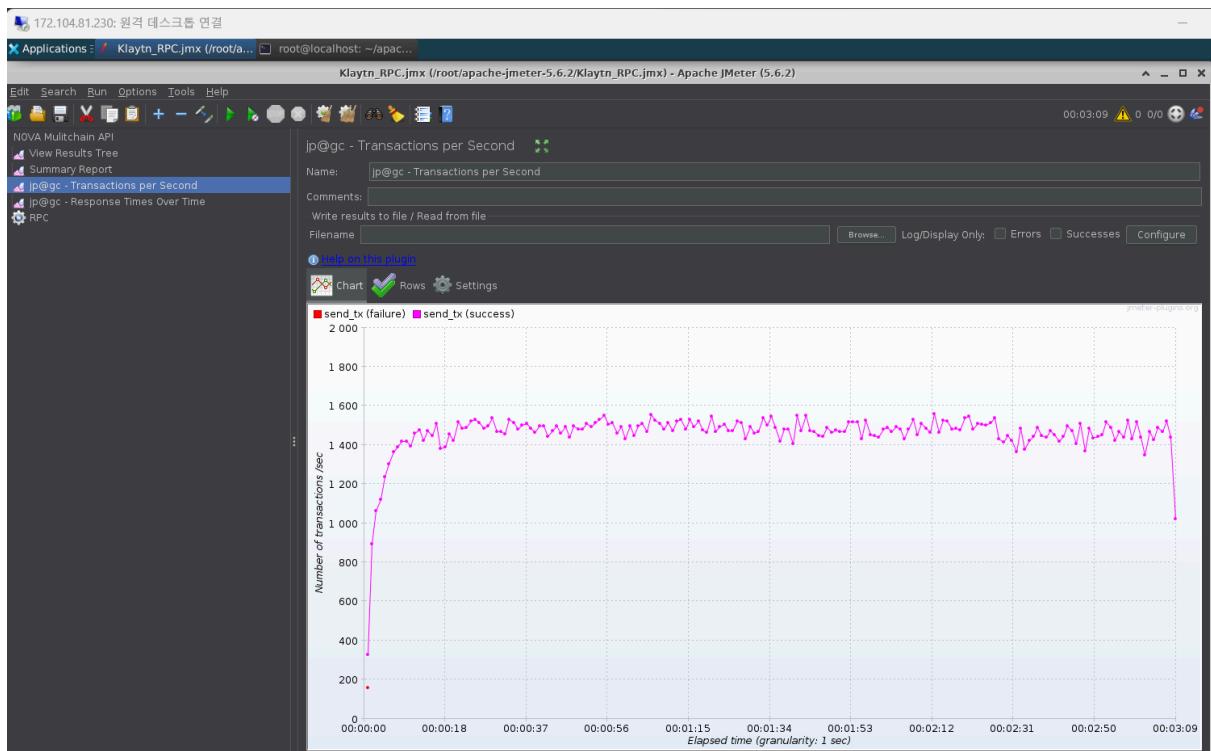
```
# jmeter 실행
```

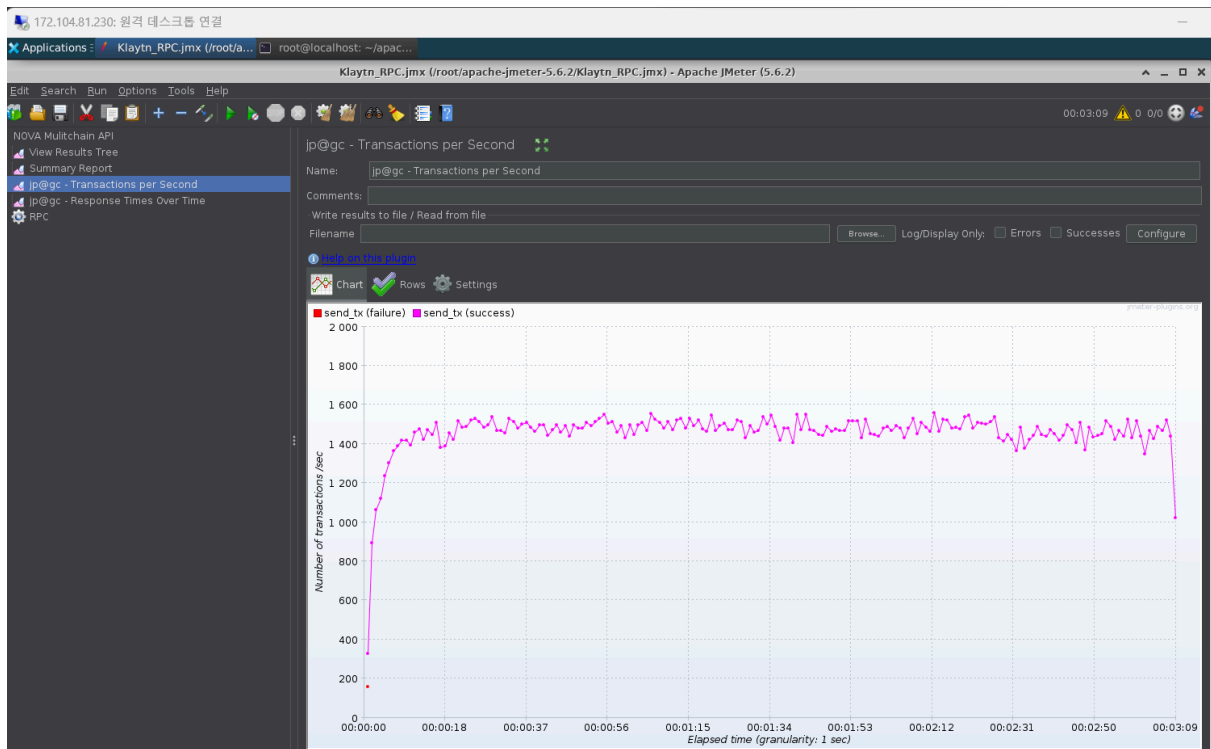
```
./jmeter-server &
```

성능 테스트 구성



테스트 결과





블록원장 사이즈 확인

\$ du -sm ~/klaytn/data/klay/chaindata

```
tmpfs          7.8G      0  7.8G    0% /sys/fs/cgroup
/dev/sda       315G    4.5G  294G    2% /
tmpfs          1.6G      0  1.6G    0% /run/user/0
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1559    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1561    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1562    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1563    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1565    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1566    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1567    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]# du -sm ~/klaytn/data/klay/chaindata
1569    /root/klaytn/data/klay/chaindata
[root@172-104-77-201 ~]#
```

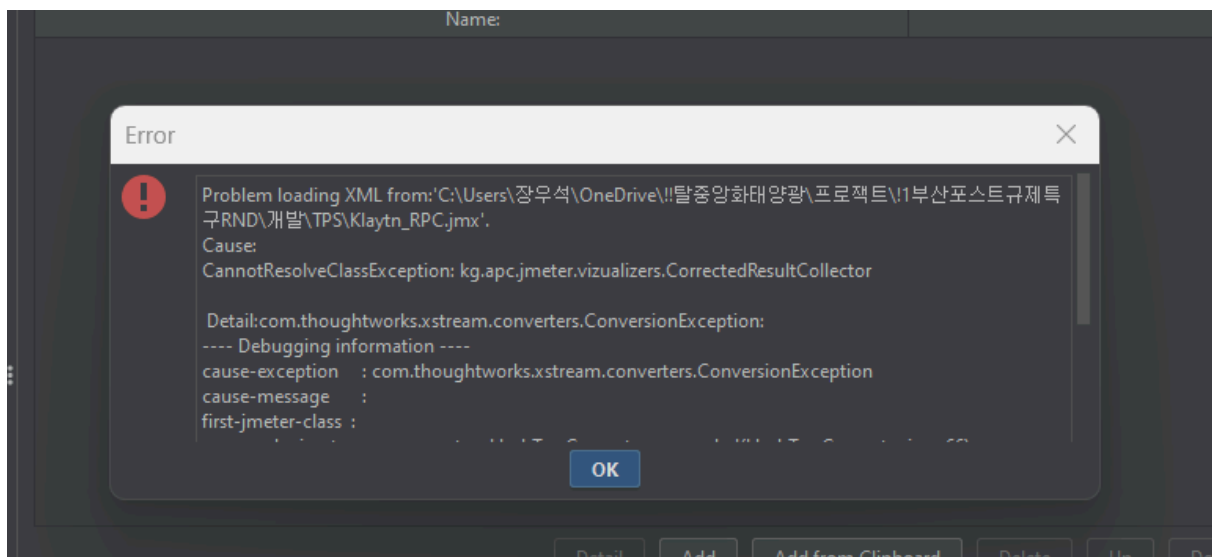
```
echo "$(date '+%Y-%m-%d %H:%M:%S') - $(du -sm  
~/klaytn/data/klay/chaindata | awk '{print $1}')
```

Trouble shootings

자바 버전을 server(리눅스)-client(마스터PC, 윈도우) 모두 같은 버전으로 세팅,
1.8.0_381

```
PS C:\Users\장우석> java -version  
java version "1.8.0_381"  
Java(TM) SE Runtime Environment (build 1.8.0_381-b09)  
Java HotSpot(TM) 64-Bit Server VM (build 25.381-b09, mixed mode)  
PS C:\Users\장우석>
```

Jmeter 설정



1. plugins-manager.jar 를 다운받는다.

<https://jmeter-plugins.org/downloads/all/>

2. 위 파일을 {jmeter 설치 폴더}/lib/ext 아래에 옮겨두기

3. jmeter 리스타트


4. 재시도(.jmx 파일 open)

→ 해결완료

Remote Agent 설정

Jmeter remote start (원격pc로 jmeter테스트하기)

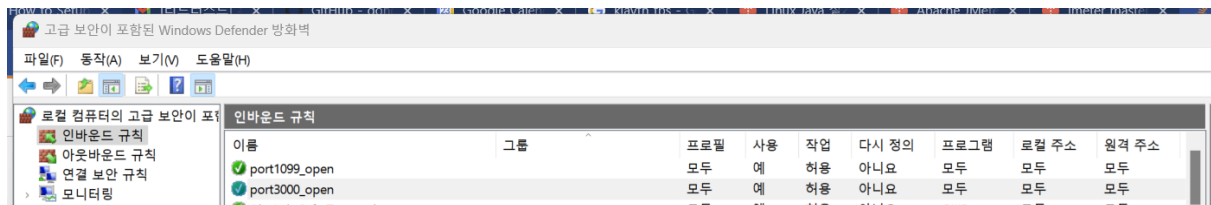
로컬PC로 Jmeter스트레스테스트를 하게 되면, 로컬PC의 성능이나 동시작업중인것에 따라 스트레스 테스트가 제대로 나오지 않을수 있습니다. 고로, 원격PC를 통해서 하게 되는데 원격PC로 테스트플랜파일

 <https://stormaa.tistory.com/75>

comma delimited
.111.122.133,
-localhost:1099

```
# Remote Hosts - comma delimited  
remote_hosts=139.162.120.37  
#remote_hosts=localhost:1099,localhost:2010
```

MasterPC 방화벽 포트오픈



설정1 (현재)

<https://velog.io/@leejh9022/Jmeter-설치-Jmeter-이용한-부하테스트>

위 블로그에 따르면, Jmeter server (slave) 설정에서 server.rmi.localport=4000

설정테스트2

<https://suricata.tistory.com/44>

기존 (slave)	블로그세팅 (slave)
remote_hosts=127.0.0.1 server_port=1099	remote_hosts=127.0.0.1 server_port=1099

#client.rmi.localport=0 server.rmi.port=1099 server.rmi.localport=4000	client.rmi.localport=1099 server.rmi.port=1099 server.rmi.localport=1099
기존 (master)	블로그세팅 (master)
#server_port=1099 #client.rmi.localport=0 #server.rmi.localport=4000	server_port=1099 client.rmi.localport=1099 server.rmi.localport=1099