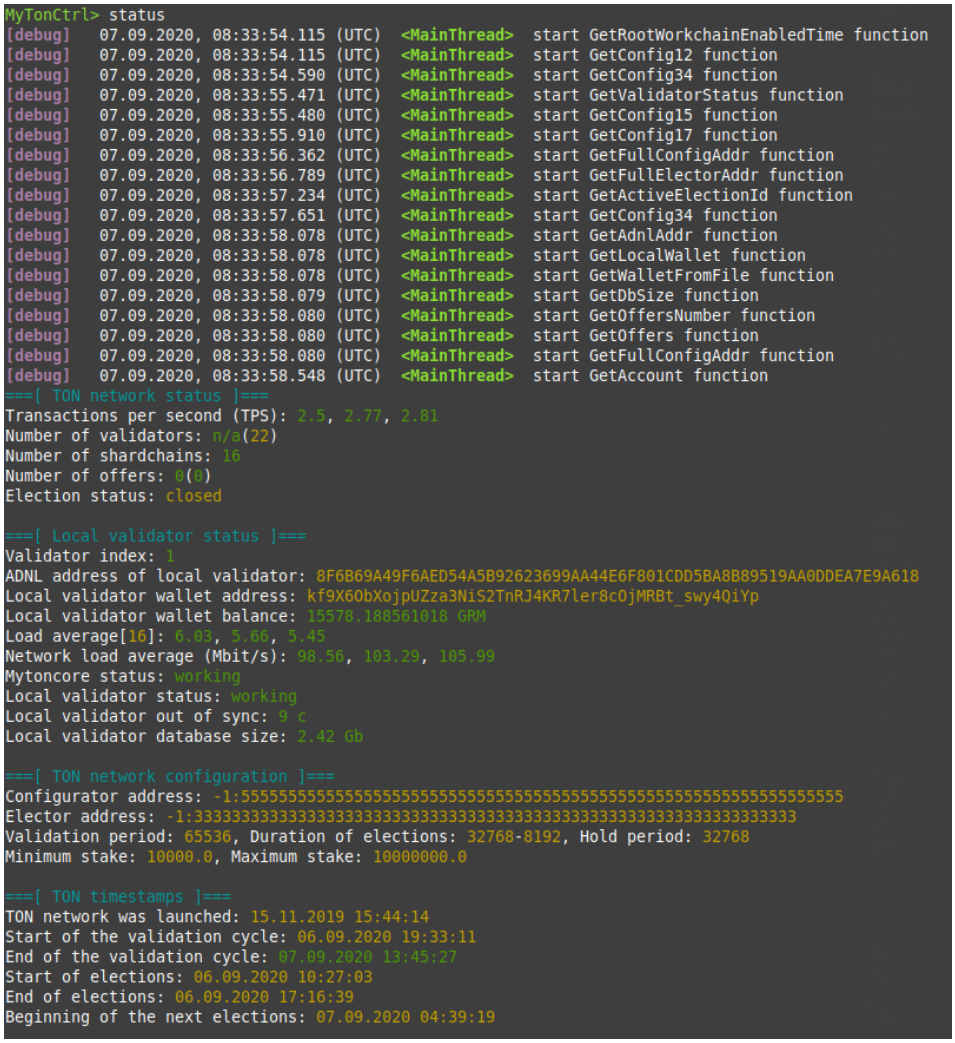
1) MyTonCtrl이란 무엇입니까?

MyTonCtrl은 fift, lite-client 및 validator-engine-console에 대한 편리한 래퍼 역할을 하는 콘솔 애플리케이션입니다. 이는 Linux 운영 체제에서 지갑, 도메인 및 유효성 검사기 관리 작업을 간소화하기 위해 특별히 개발되었습니다.



기능성

TON 네트워크 상태 표시
로컬 지갑 관리

- 로컬 지갑 생성
- 로컬 지갑 활성화
- 지역 지갑 표시
- 파일(.pk)에서 지갑 가져오기
- 지갑 주소를 파일(.addr)에 저장

- 로컬 지갑 삭제

계정 상태 표시

- 계좌 잔액 표시
- 계정 내역 표시
- 북마크에서 계정 상태 표시

지갑으로 자금 이체

- 일정금액 이체
- 전액 이체(모두)
- 지갑 비활성화로 전체 금액 이체(alld)
- 북마크에서 지갑으로 자금 이체
- 자체 삭제 지갑 체인을 통해 지갑으로 자금 이체

북마크 관리

- 북마크에 계정 추가
- 북마크 표시
- 북마크 삭제

제안 관리

- 쿠폰 표시
- 제안에 투표하세요
- 이전에 투표된 제안에 대한 자동 투표

도메인 관리

- 새 도메인 임대
- 임대한 도메인 표시
- 도메인 상태 표시
- 도메인 삭제
- 자동 도메인 갱신

유효성 검사기 제어

- 검증인 선출에 참여
- 베팅 반환 + 보상
- 비정상 종료 시 자동 시작 유효성 검사기(systemd)
- https://toncenter.com으로 검증인 통계 보내기

테스트된 운영 체제 목록

운영 체제	상태
우분투 **16.04 LTS (Xenial Xerus)**	오류**: TON** 컴파일 오류
우분투 **18.04 LTS(**바이오닉 비버**)**	**OK**
우분투 **20.04 LTS(**포칼 포사**)**	**OK**
우분투 **22.04 LTS(**제미	**OK**

젤리피쉬)	
데비안 **8**	오류**: libgsl-dev** 패키지를 찾을 수 없습니다**.**
데비안 **9**	오류**: TON** 컴파일 오류
데비안 **10**	**OK**

설치

설치 스크립트 개요

- toninstaller.sh: TON 및 mytonctrl 소스를 /usr/src/ton 및/usr/src/mytonctrl 폴더에 복제하고 소스에서 프로그램을 컴파일하여 /usr/bin/에 씁니다.
- mytoninstaller.py: 유효성 검사기와 mytonctrl을 구성합니다. 유효성 검사기 연결 키를 생성합니다.

설치 모드

설치 모드에는 라이트와 전체의 두 가지가 있습니다. 둘 다 TON 구성 요소를 컴파일하고 설치합니다. 그러나 라이트 버전은 노드/검증기를 구성하거나 실행하지 않습니다.

우분투 설치

- 원하는 설치 모드에서 install.sh 스크립트를 다운로드하고 실행합니다. 설치하는 동안 스크립트는 수퍼유저 비밀번호를 묻는 메시지를 여러 번 표시합니다. wget https://raw.githubusercontent.com/ton-blockchain/mytonctrl/master/scripts/install.sh sudo bash install.sh -m <mode>
- 완료. 이제 mytonctrl 콘솔을 실행해 볼 수 있습니다. mytonctrl

데비안 설치

- 원하는 설치 모드에서 install.sh 스크립트를 다운로드하고 실행합니다. 설치하는 동안 스크립트는 수퍼유저 비밀번호를 묻는 메시지를 여러 번 표시합니다. wget https://raw.githubusercontent.com/ton-blockchain/mytonctrl/master/scripts/install.sh su root -c 'bash install.sh -m <mode>'
- 완료. 이제 mytonctrl 콘솔을 실행해 볼 수 있습니다. mytonctrl

MyTonCtrl 문서

이 저장소에는 언어별로 분류된 다음과 같은 MyTonCtrl 기술 문서가 포함되어 있습니다. 아래 링크를 클릭하시면 관심 있는 문서로 이동하실 수 있습니다.

	자주하는 질문	지갑 가져오기	우분투 매뉴얼
영어(EN)	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/en/FAQ.md	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/en/import-wallets.md	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/en/manual-ubuntu.md
러시아어(RU)	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/ru/FAQ.md	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/ru/import-wallets.md	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/ru/manual-ubuntu.md
중국어 번체	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/zh_TW/FAQ.md	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/zh_TW/import-wallets.md	https://github.com/ton-blockchain/mytonctrl/blob/master/docs/zh_TW/manual-ubuntu.md

원격 측정

기본적으로 mytonctrl은 유효성 검사기 통계를 https://toncenter.com 서버로 보냅니다. 네트워크 이상 징후를 파악하고, 개발자에게 신속한 피드백을 제공하는 것이 필요합니다. 설치 중에 원격 측정을 비활성화하려면 -t 플래그를 사용하십시오.

```
sudo bash install.sh -m <mode> -t
```

설치 후 원격 측정을 비활성화하려면 다음을 수행합니다.

```
MyTonCtrl> set sendTelemetry false
```

MyTonCtrl 설치 프로그램 모드

웹 관리 패널

브라우저를 통해 노드/검증기를 제어하려면 추가 모듈을 설치해야 합니다: mytonctrl -> 설치 프로그램 -> JR 활성화

다음으로, 연결을 위한 비밀번호를 생성해야 합니다: mytonctrl -> installer -> setwebpass

준비가 된. 이제 https://tonadmin.org 사이트로 이동하여 자격 증명으로 로그인할 수 있습니다. 자식: https://github.com/igroman787/mtc-jsonrpc

toncenter의 로컬 복사본

서버에 로컬 https://toncenter.com 복사본을 설정하려면 추가 모듈을 설치하십시오: mytonctrl ->installer -> PT 활성화

준비가 된. toncenter의 로컬 복사본은 http://<server-ip-address>:8000 git: https://github.com/igroman787/pytonv3 에서 사용할 수 있습니다.

유용한 링크

- https://docs.ton.org/

2) mytonctrl(v0.2, OS Ubuntu)을 사용하여 유효성 검사기가 되는 방법

mytonctrl을 사용하여 유효성 검사기가 되는 단계는 다음과 같습니다. 이 예는 Ubuntu 운영 체제에 적용됩니다.

1. mytonctrl을 설치합니다**:**

- 설치 스크립트를 다운로드합니다. 루트가 아닌 로컬 사용자 계정으로 도구를 설치하는 것이 좋습니다. 이 예에서는 로컬 사용자 계정이 사용됩니다. wget

https://raw.githubusercontent.com/ton-blockchain/mytonctrl/master/scripts/install.sh

- 관리자로 설치 스크립트를 실행합니다. wget https://raw.githubusercontent.com/ton-blockchain/mytonctrl/master/scripts/install.sh

2. 운용성 테스트 실시**:**

- 1단계에서 설치에 사용한 로컬 사용자 계정에서 mytonctrl을 실행합니다. mytonctrl
- mytonctrl 상태, 특히 다음을 확인하십시오.
- mytoncore 상태: 녹색이어야 합니다.

- 로컬 검증인 상태: 또한 녹색이어야 합니다.

- 로컬 검증인이 동기화되지 않음: 처음에는 큰 숫자가 표시됩니다. 새로 생성된 검증인이 다른 검증인과 연결되면 그 수는 약 250,000개가 됩니다. 동기화가 진행됨에 따라 이

숫자는 감소합니다. 20 미만으로 떨어지면 유효성 검사기가 동기화됩니다.

```
MyTonCtrl> status
[debug]   07.09.2020, 08:33:54.115 (UTC)  <MainThread>  start GetRootWorkchainEnabledTime function
[debug]   07.09.2020, 08:33:54.115 (UTC)  <MainThread>  start GetConfig12 function
[debug]   07.09.2020, 08:33:54.590 (UTC)  <MainThread>  start GetConfig34 function
[debug]   07.09.2020, 08:33:55.471 (UTC)  <MainThread>  start GetValidatorStatus function
[debug]   07.09.2020, 08:33:55.480 (UTC)  <MainThread>  start GetConfig15 function
[debug]   07.09.2020, 08:33:55.910 (UTC)  <MainThread>  start GetConfig17 function
[debug]   07.09.2020, 08:33:56.362 (UTC)  <MainThread>  start GetFullConfigAddr function
[debug]   07.09.2020, 08:33:56.789 (UTC)  <MainThread>  start GetFullElectorAddr function
[debug]   07.09.2020, 08:33:57.234 (UTC)  <MainThread>  start GetActiveElectionId function
[debug]   07.09.2020, 08:33:57.651 (UTC)  <MainThread>  start GetConfig34 function
[debug]   07.09.2020, 08:33:58.078 (UTC)  <MainThread>  start GetAdnlAddr function
[debug]   07.09.2020, 08:33:58.078 (UTC)  <MainThread>  start GetLocalWallet function
[debug]   07.09.2020, 08:33:58.078 (UTC)  <MainThread>  start GetWalletFromFile function
[debug]   07.09.2020, 08:33:58.079 (UTC)  <MainThread>  start GetDbSize function
[debug]   07.09.2020, 08:33:58.080 (UTC)  <MainThread>  start GetOffersNumber function
[debug]   07.09.2020, 08:33:58.080 (UTC)  <MainThread>  start GetOffers function
[debug]   07.09.2020, 08:33:58.080 (UTC)  <MainThread>  start GetFullConfigAddr function
[debug]   07.09.2020, 08:33:58.548 (UTC)  <MainThread>  start GetAccount function
===[ TON network status ]===
Transactions per second (TPS): 2.5, 2.77, 2.81
Number of validators: n/a(22)
Number of shardchains: 16
Number of offers: 0(0)
Election status: closed

===[ Local validator status ]===
Validator index: 1
ADNL address of local validator: 8F6B69A49F6AED54A5B92623699AA44E6F801CDD5BA8B89519AA0DDEA7E9A618
Local validator wallet address: kf9X6ObXojpUZza3NiS2TnRJ4KR7ler8cOjMRBt_swy4QiYp
Local validator wallet balance: 15578.188561018 GRM
Load average[16]: 6.03, 5.66, 5.45
Network load average (Mbit/s): 98.56, 103.29, 105.99
Mytoncore status: working
Local validator status: working
Local validator out of sync: 9 c
Local validator database size: 2.42 Gb

===[ TON network configuration ]===
Configurator address: -1:5555555555555555555555555555555555555555555555555555555555555555
Elector address: -1:3333333333333333333333333333333333333333333333333333333333333333
Validation period: 65536, Duration of elections: 32768-8192, Hold period: 32768
Minimum stake: 10000.0, Maximum stake: 10000000.0

===[ TON timestamps ]===
TON network was launched: 15.11.2019 15:44:14
Start of the validation cycle: 06.09.2020 19:33:11
End of the validation cycle: 07.09.2020 13:45:27
Start of elections: 06.09.2020 10:27:03
End of elections: 06.09.2020 17:16:39
Beginning of the next elections: 07.09.2020 04:39:19
```

3. 사용 가능한 지갑 목록 보기

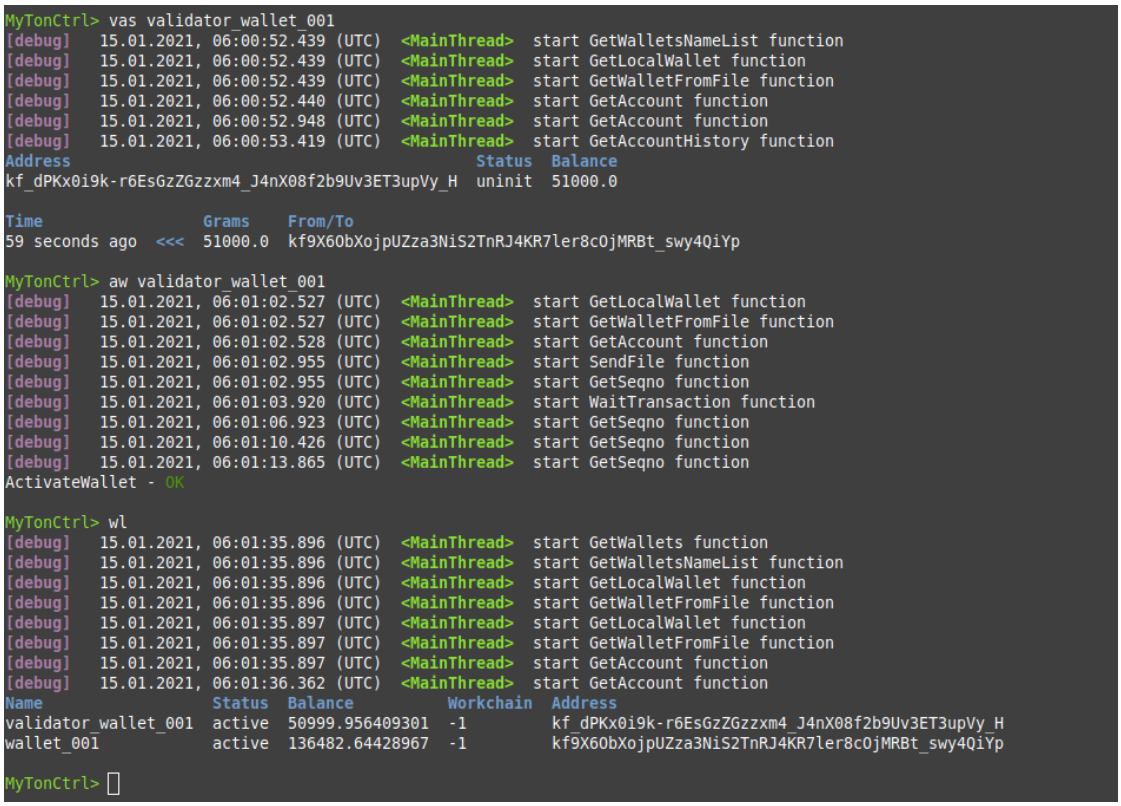
사용 가능한 지갑 목록을 확인하세요. 예를 들어 mytonctrl을 설치하는 동안 validator_wallet_001 지갑이 생성됩니다.

```
user-user{~}:mytonctrl
[debug]   15.01.2021, 05:56:04.961 (UTC)  <Logging>     Start WritingLogFile thread.
[debug]   15.01.2021, 05:56:04.961 (UTC)  <SelfTesting> Start SelfTesting thread.
[debug]   15.01.2021, 05:56:04.961 (UTC)  <LocdbSaving> Start LocaldbSaving thread.
[info]    15.01.2021, 05:56:04.961 (UTC)  <MainThread>  Start program '/usr/src/mytonctrl/mytonctrl.py'
Welcome to the console. Enter 'help' to display the help menu.
MyTonCtrl> wl
[debug]   15.01.2021, 05:56:08.080 (UTC)  <MainThread>  start GetWallets function
[debug]   15.01.2021, 05:56:08.080 (UTC)  <MainThread>  start GetWalletsNameList function
[debug]   15.01.2021, 05:56:08.080 (UTC)  <MainThread>  start GetLocalWallet function
[debug]   15.01.2021, 05:56:08.081 (UTC)  <MainThread>  start GetWalletFromFile function
[debug]   15.01.2021, 05:56:08.081 (UTC)  <MainThread>  start GetLocalWallet function
[debug]   15.01.2021, 05:56:08.081 (UTC)  <MainThread>  start GetWalletFromFile function
[debug]   15.01.2021, 05:56:08.081 (UTC)  <MainThread>  start GetAccount function
[debug]   15.01.2021, 05:56:08.561 (UTC)  <MainThread>  start GetAccount function
Name                Status  Balance           Workchain  Address
validator_wallet_001  empty   0                 -1         0f_dPKx0i9k-r6EsGzZGzzxm4_J4nX08f2b9Uv3ET3upV3IC
wallet_001            active  187482.697291926  -1         kf9X6ObXojpUZza3NiS2TnRJ4KR7ler8cOjMRBt_swy4QiYp

MyTonCtrl>
```

4. 필요한 만큼의 코인을 지갑으로 보내고 활성화하세요**.**

한 번의 선거 라운드에 참여하는 데 필요한 최소 코인 금액을 확인하려면 tonmon.xyz > Participant stakes.

- vas 명령을 사용하여 전송 기록을 표시합니다.
- aw 명령을 사용하여 지갑을 활성화합니다.



5. 이제 유효성 검사기가 준비되었습니다

mytoncore는 자동으로 선거에 참여합니다. 지갑 잔액을 두 부분으로 나누어 선거에 참여하기 위한 지분으로 사용합니다. 스테이크 크기를 수동으로 설정할 수도 있습니다.

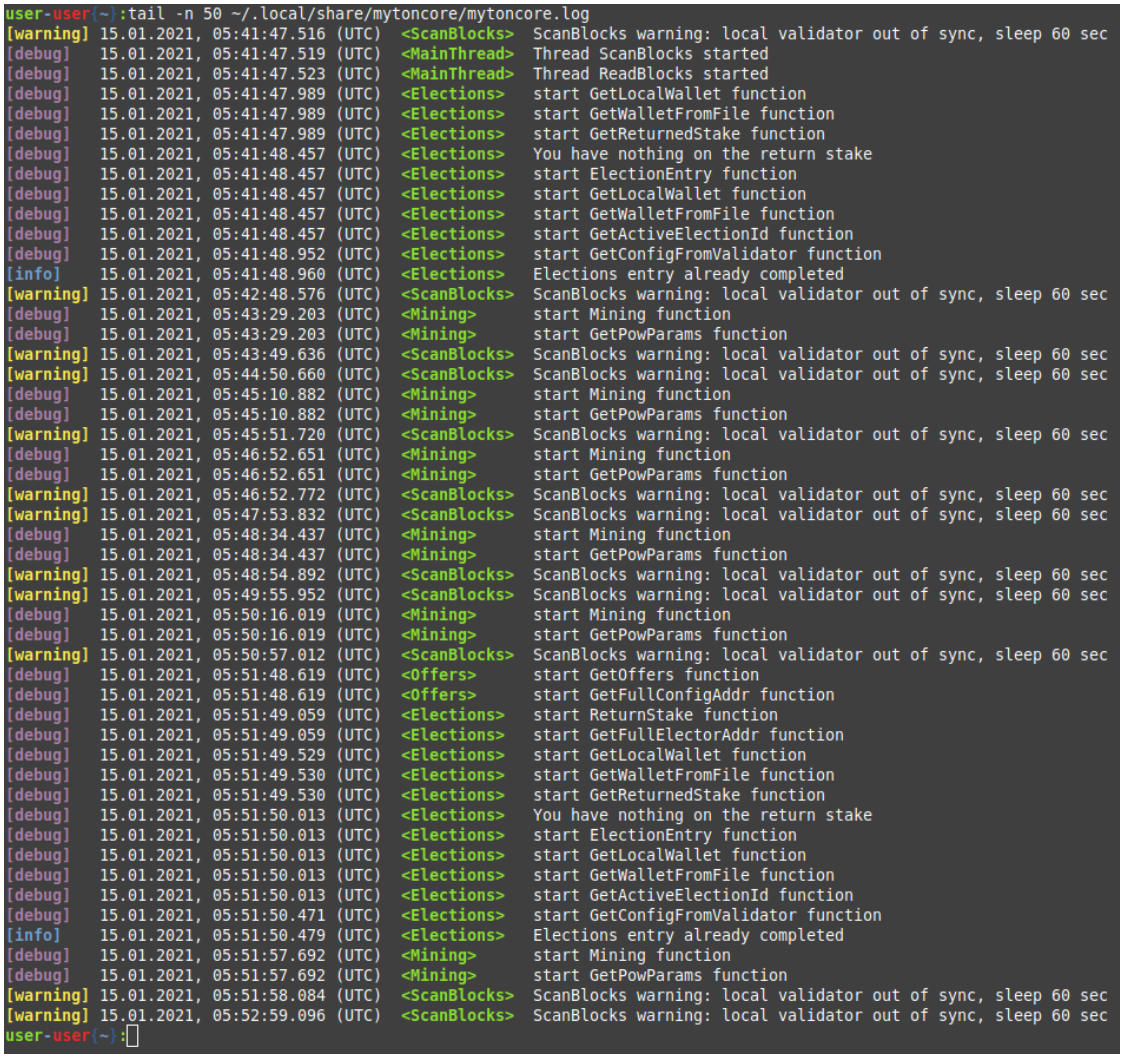
스테이크 50000 설정 - 스테이크 크기를 50,000코인으로 설정합니다. 베팅이 수락되고 우리 노드가 검증자가 되면 베팅은 두 번째 선거에서만 철회될 수 있습니다(유권자의 규칙에 따라).

```
user-user(~):mytonctrl
[debug]   15.01.2021, 06:17:25.851 (UTC)  <Logging>     Start WritingLogFile thread.
[debug]   15.01.2021, 06:17:25.852 (UTC)  <SelfTesting> Start SelfTesting thread.
[debug]   15.01.2021, 06:17:25.852 (UTC)  <LocdbSaving> Start LocaldbSaving thread.
[info]    15.01.2021, 06:17:25.852 (UTC)  <MainThread>  Start program '/usr/src/mytonctrl/mytonctrl.py'
Welcome to the console. Enter 'help' to display the help menu.
MyTonCtrl> set stake 50000
SetSettings - OK

MyTonCtrl> get stake
50000

MyTonCtrl>
```

언제든지 도움을 요청할 수도 있습니다.

```
MyTonCtrl> help
help        Print help text
clear       Clear console
exit        Exit from application
update      Подтянуть обновление mytonctrl
upgrade     Подтянуть исходный код и перекомпилировать компоненты TON
installer   Запустить установщик модулей TON
status      Показать статус TON
seqno       Получить seqno кошелька
nw          Создать новый локальный кошелек
aw          Активировать локальный кошелек
wl          Показать локальные кошельки
iw          Импортировать кошелек из файла
swa         Сохранить адрес кошелька в файл
dw          Удалить локальный кошелек
vas         Показать статус аккаунта
vah         Показать историю аккаунта
mg          Перевод средств на кошелек
nb          Добавить аккаунт в закладки
bl          Показать закладки
db          Удалить закладку
nd          Арендовать новый домен
dl          Показать арендованные домены
vds         Показать статус домена
dd          Удалить домен
ol          Показать действующие предложения
vo          Голосовать за предложение
el          Показать действующие выборы
ve          Голосовать в выборах
vl          Показать действующие валидаторы
get         Посмотреть настройки
set         Задать настройки
test        Test
pt          PrintTest

MyTonCtrl>
```

mytoncrl 로그를 확인하려면 다음을 엽니다.
~/.local/share/mytoncore/mytoncore.log 로컬 사용자의 경우 또는
/usr/local/bin/mytoncore/mytoncore.log 루트용.



3) 추천자 풀

후보 풀 모드에서 유효성 검사기 실행

1. 유효성 검사기용 하드웨어를 설정하세요. vCPU 8개, 64GB 메모리, 1TB SSD, 고정 IP 주소, 1Gb/s 인터넷 속도가 필요합니다. 네트워크 안정성을 유지하려면 검증인 노드를 단일 데이터 센터에 집중시키기보다는 전세계 다양한 지리적 위치에 분산시키는 것이 좋습니다. 당신이 사용할 수있는 https://status.toncenter.com 다양한 위치의 부하를 평가합니다. 지도는 유럽, 특히 핀란드, 독일, 파리에서 데이터 센터 활용도가 높다는 것을 나타냅니다. 따라서 Hetzner, OVH와 같은 공급자를 사용하는 것은 권장되지 않습니다. (*하드웨어가 위의 사양과 일치하거나 이를 초과하는지 확인하세요. 부족한*

*하드웨어에서 유효성 검사기를 실행하면 네트워크에 부정적인 영향을 미치고 처벌을 받을 수 있습니다. **2021**년 **5**월부터 **Hetzner**는 서버에서의 채굴을 금지했으며, 이러한 금지에는 **PoW** 및 **PoS** 알고리즘이 모두 포함됩니다. 일반 노드를 설치하는 것조차 서비스 약관을 위반하는 것으로 간주될 수 있습니다. 권장 공급자에는 **Amazon, DigitalOcean, Linode, Alibaba Cloud, Latitude**가 있습니다*).

2. 가이드에 설명된 대로 mytonctrl을 설치하고 동기화합니다 https://github.com/ton-blockchain/mytonctrl/blob/master/docs/en/manual-ubuntu.md — 1, 2, 3단계만 따르십시오. 이것도 참고하시면 됩니다 https://ton.org/docs/#/nodes/run-node 추가 도움이 필요한 경우.
3. wl 목록에 표시된 검증인 지갑 주소로 1TON을 전송합니다.
4. aw 명령을 사용하여 유효성 검사기 지갑을 활성화하세요.
5. 두 개의 풀을 만듭니다(짝수 및 홀수 검증 라운드용).

```
new_pool p1 0 1 1000 300000
new_pool p2 0 1 1001 300000
```

어디:

- p1은 풀 이름입니다.
- 0%는 검증인의 보상 지분입니다(예: 40%에 40을 사용).
- 1은 풀의 최대 추천자 수입니다(<= 40이어야 함).
- 1000 TON은 최소 검증인 스테이크입니다(1K TON 이상이어야 함).
- 300000 TON은 최소 지정자 지분입니다(10K TON 이상이어야 함). ((*!*) 풀 구성은 동일할 필요가 없으며, 한 풀의 최소 지분에 ***1***을 추가하여 다르게 만들 수 있습니다. *(!)* https://tonmon.xyz/ 를 사용하여 현재 최소 검증인 지분을 결정하세요*)*.

6. 풀 주소를 표시하려면 pools_list를 입력하세요.

```
pools_list
Name  Status  Balance  Address
p1    empty   0        0f98YhXA9wnr0d5XRXT-I2yH54nyQzn0tuAYC4FunT780qIT
p2    empty   0        0f9qtmnzs2-PumMisKDmv6KNjNfOMDQG70mQdp-BcAhnV5jL
```

7. 각 풀에 1TON을 보내고 풀을 활성화합니다.

```
mg validator_wallet_001 0f98YhXA9wnr0d5XRXT-I2yH54nyQzn0tuAYC4FunT780qIT 1   mg
validator_wallet_001 0f9qtmnzs2-PumMisKDmv6KNjNfOMDQG70mQdp-BcAhnV5jL 1
activate_pool p1
activate_pool p2
```

8. 풀을 표시하려면 pools_list를 입력하세요.

```
pools_list
Name  Status  Balance      Address
p1    active  0.731199733  kf98YhXA9wnr0d5XRXT-I2yH54nyQzn0tuAYC4FunT780v_W
p2    active  0.731199806  kf9qtmnzs2-PumMisKDmv6KNjNfOMDQG70mQdp-BcAhnV8UO
```

9. https://tonscan.org/nominator/<address_of_pool>". 링크를 통해 각 풀을 열고 풀 구성을 확인하세요.

10. 각 풀에 검증인 입금을 진행하세요.

```
deposit_to_pool validator_wallet_001 <address_of_pool_1> 1005
deposit_to_pool validator_wallet_001 <address_of_pool_2> 1005
```

이 명령에서 1005 TON은 입금액입니다. 보증금 처리를 위해 풀에서 1TON을 차감합니다.

11. 각 풀에 지명자 입금을 진행하세요:
- 풀 링크(9단계)를 방문하여 ADD STAKE를 클릭하세요. 다음 명령을 사용하여 mytonctrl을 사용하여 입금할 수도 있습니다.

```
mg nominator_wallet_001 <address_of_pool_1> 300001 -C d
mg nominator_wallet_001 <address_of_pool_2> 300001 -C d
```

- ☐ ***(!)*** 노미네이터 지갑은 베이스체인***(***워크체인 ***0)***에서 초기화되어야 합니다**.**
- ☐ ***(!)*** 검증인 지갑과 노미네이터 지갑은 별도로 보관해야 한다는 점 꼭 기억해주세요***!*** 모든 시스템 거래의 처리를 보장하기 위해 검증자 지갑은 검증자 노드와 함께 서버에 저장되어야 합니다**.** 한편**,** 노미네이터 지갑은 콜드 암호화폐 지갑에 보관되어야 합니다**.**
- ☐ 지명자 예치금을 인출하려면 ***w***라는 코멘트가 포함된 트랜잭션을 풀 주소로 보냅니다***(***트랜잭션을 처리하려면 ***1TON***을 첨부하세요***). mytonctrl***을 사용하여 이 작업을 수행할 수도 있습니다**.**

12. 풀 모드 활성화:

```
set usePool true
set stake null
```

13. 귀하의 풀에 입금할 후보를 초대하십시오. 검증 참여는 자동으로 시작됩니다.

- ☐ ***(!)*** 검증인 지갑에 운영비로 최소 ***200TON***/월이 있어야 합니다**.**

풀 구성

자신에게 빌려주려는 경우 new_pool p1 0 1 1000 300000(최대 1명의 추천자, 0% 검증인 지분)을 사용하세요.

수많은 추천자를 위한 풀을 생성하는 경우 다음과 같은 것을 사용할 수 있습니다: new_pool p1 40 40 10000 10000 (최대 40명의 추천자, 40% 검증인 지분, 최소 참가자 스테이크 10K TON).

일반 검증인을 후보 풀 모드로 전환

- 선거 참여를 중단하려면 set 스테이크 0을 입력하세요.
- 유권자로부터 두 지분이 모두 반환될 때까지 기다리십시오.
- 4단계부터 "노미네이터 풀 모드에서 검증인 실행" 단계를 진행하세요.

4) 지갑 가져오기

MyTonCtrl은 wallet-v1, wallet-v3, lockup-wallet 등 다양한 유형의 지갑형 계약을 지원합니다. 종종 이는 이러한 계약과 상호 작용하는 간단한 방법을 제공합니다.

개인 키를 사용하여 가져오기

개인키에 접근할 수 있다면 쉽게 지갑을 가져올 수 있습니다. 콘솔에 다음 명령을 입력합니다.
iw <wallet-addr> <wallet-secret-key>
여기서 <wallet-secret-key>는 base64 형식의 개인 키입니다.

니모닉 문구를 사용하여 가져오기

니모닉 문구(... 동안 문신과 같은 24개 단어의 시퀀스)가 있는 경우 다음 단계를 따르세요.

- Node.js를 설치합니다.
- mnemonic2key를 복제하고 설치합니다. git clone https://github.com/ton-blockchain/mnemonic2key.git cd mnemonic2key npm install
- 다음 명령을 실행하여 word1, word2...를 니모닉 문구로 바꾸고 주소를 지갑 계약 주소로 바꾸세요. node index.js word1 word2 ... word24 [address]
- 스크립트는 wallet.pk 및 wallet.addr을 생성합니다. import_wallet.pk 및 import_wallet.addr로 이름을 바꾸세요.
- 두 파일을 모두 ~/.local/share/mytoncore/wallets/ 디렉터리에 복사합니다.
- mytonctrl 콘솔을 열고 wl 명령을 사용하여 지갑을 나열하십시오.
- 지갑을 가져왔고 올바른 잔액이 표시되는지 확인하세요.
- 이제 mg 명령을 사용하여 자금을 보낼 수 있습니다. 도움말 문서를 보려면 mg를 입력하세요.

명령을 실행할 때 자리 표시자(< > 안의 단어)를 실제 값으로 바꾸는 것을 잊지 마십시오.

5) 자주 묻는 질문

1) 이전 블록에 따라 데이터를 쿼리하는 **API** 메서드가 있나요**?** 예를 들어**,** 이전 블록 번호를 지정하고 지갑의 **TON** 잔액을 쿼리합니다**.**

다른 예는 이전 블록에 따라 **jetton** 계약 균형 방법과 같은 **getter**를 실행하는 것입니다**.**

이를 위해 아카이브 노드가 필요합니까**?**

답변

- 톤 풀 노드(https://github.com/ton-blockchain/ton/blob/master/validator/validator.h#L95)의 기본 구성은 1시간 동안 상태를 저장하는 것입니다. 블록 관련 데이터는 1주일 동안 저장됩니다. 이 기간이 지나면 둘 다 아카이브 디렉토리로 이동됩니다. 아카이브 디렉토리는 1년 후에 삭제됩니다. 이 구성은 변경될 수 있으며 예를 들어 보관 노드는 훨씬 더 큰 값으로 노드를 배포하므로 사용자는 언제든지 기록에서 상태를 쿼리할 수 있습니다. 이전 상태(1시간 이상 전)를 쿼리하려면 아카이브 노드를 사용해야 합니다. TON APIv4를 사용하면 jetton 잔액을 쿼리하기 위해 method=get_wallet_data와 함께 사용할 수 있는 GET /block/<seqno>/<address>/run/<method>/<args?>를 쿼리할 수 있습니다. 예를 들어 Toncenter에서는 이전 상태에서 메서드를 쿼리하는 것이 불가능합니다. 그러나 archive=true와 함께 getTransactions를 사용하면 모든 트랜잭션을 가져오고 각 상태의 정확한 금액을 계산할 수 있습니다. 비아카이브(기본) 노드의 트랜잭션은 1주일 동안 저장됩니다.

2) The Open Network 거래에 대한 가스 수수료는 어떻게 계산되나요**?**

답변

- 첫째, 모든 TON 사용자는 커미션이 여러 요인에 따라 달라진다는 점을 명심해야 합니다. 문서에 따르면 TON의 커미션은 다음 공식으로 계산됩니다. 거래 수수료 = 저장 수수료 + infwd 수수료 + 계산 수수료 + 조치 수수료 + outfwd 수수료 저장 수수료는 블록체인에 스마트 계약을 저장하기 위해 지불하는 금액입니다. 실제로 스마트 계약이 블록체인에 저장되는 매초마다 비용을 지불하게 됩니다. TON 지갑은 거래를 받거나 보낼 때마다 임대료를 청구하는 스마트 계약이기도 합니다. infwdfees는 블록체인 외부에서 메시지를 가져오는 데 대한 요금입니다. 메시지를 처리하기 전에 메시지는 최종 샤드체인의 검증자에게 전달되어야 합니다. 거래를 할 때마다 이를 처리할 검증자에게 전달되어야 합니다. 예를 들어 Tonkeeper와 같은 지갑 앱을 사용하여 수행하는 각 거래는 먼저 검증 노드 간에 전달되어야 합니다. 계산 수수료는 가상 머신에서 코드를 실행하기 위해 지불하는 금액입니다. 가상 머신은 전송된 메시지로부터 입력 매개변수를 수신하고 스마트 계약 코드를 실행합니다. 더 많은 명령이 실행될수록 더 많은 커미션이 발생합니다. 예를 들어, 지갑(스마트 계약)으로 거래를 보낼 때마다 지갑 계약의 코드가 실행됩니다.
- actionfees는 스마트 계약 코드 실행 후 수신된 작업 목록을 처리하는 데 드는 수수료입니다. 이 프로세스를 통해 나가는 메시지는 다른 스마트 계약이나 블록체인에서 수행되는 기타 가시적 작업으로 전송됩니다.
- outfwdfees – 오프체인 서비스(예: 로그) 및 외부 블록체인과 상호 작용하기 위해 TON 블록체인 외부로 메시지를 보내는 데 대한 요금을 나타냅니다. 요금은 최종 샤드체인의 검증자에게 전달됩니다. 구현되지 않았기 때문에 사용되지 않습니다.

- 모든 수수료는 특정 가스량으로 지정되고 고정되지만 가스 가격 자체는 고정되어 있지 않습니다(현재 1 가스 단위의 가격은 1000나노톤입니다). TON의 다른 많은 매개변수와 마찬가지로 가스 요금도 구성 가능*하며 메인넷에서 이루어진 특별 투표를 통해 변경될 수 있습니다. 매개변수를 변경하려면 검증인 투표의 3분의 2를 얻어야 합니다. 언젠가 가스가 1000배 또는 그 이상 증가할 수 있다는 의미입니까? 기술적으로는 그렇습니다. 그러나 실제로는 그렇지 않습니다. 검증인은 거래 처리에 대해 소액의 수수료를 받으며, 수수료를 높이면 거래 수가 줄어들어 검증 프로세스의 이점이 줄어듭니다. 그렇기 때문에 수수료를 올려도 소용이 없습니다. TON의 수수료는 거래 실행 시간, 계정 상태, 메시지 내용 및 크기, 블록체인의 네트워크 설정, 그리고 거래가 전송될 때까지 계산할 수 없는 많은 변수에 따라 금액이 달라지기 때문에 미리 계산하기가 어렵습니다. 이것이 바로 NFT 마켓플레이스가 만일을 대비해 추가 금액의 TON을 가져갔다가 나중에 반환하는 이유입니다. TON에서는 스마트 계약 실행과 사용된 스토리지(바이트*초)에 대한 비용을 모두 지불한다는 점을 명심하는 것이 중요합니다. 이는 TON 지갑을 소유하기 위해 임대료를 지불해야 함을 의미합니다(보통 매우 작음). 그러나 장기간 TON 지갑을 사용하지 않은 경우 평소보다 훨씬 더 많은 수수료를 지불해야 합니다. tonmon.xyz에 따르면 초당 평균 TON 트랜잭션 수는 현재 1.4개이지만, 이 값이 크게 증가하면 다른 블록체인과 달리 수수료는 동일하게 유지됩니다. TON 공식 웹사이트에 따르면, 블록체인은 샤딩 지원 덕분에 초당 수백만 건, 필요한 경우 수천만 건의 거래를 수행할 수 있습니다. 오늘날 모든 거래 비용은 약 0.005TON입니다. TON의 가격이 50배 상승하더라도 거래는 다른 블록체인보다 저렴하게 유지됩니다. 그리고 수수료가 비싸졌다고 판단되면 검증인이 이 값을 낮출 수 있다는 점을 잊지 마십시오. 이 질문에 대해 구독자에게 감사드립니다. 우리는 TON의 매우 포괄적인 측면을 배우는 데 매우 관심이 있었습니다. 우리는 봇을 통해 모든 구독자의 질문과 피드백을 환영합니다. 우리는 항상 TON에 관한 모든 사항을 더 자세히 알아보고 명확하게 설명할 수 있어서 기쁩니다.

수수료 계산

보관_수수료

- Storage_fees = ceil((account.bits * bit_price + account.cells * cell_price) * 기간 / 2 ^ 16)

in_fwd_fees, out_fwd_fees

- msg_fwd_fees = (lump_price + ceil((bit_price * msg.bits + cell_price * msg.cells)/2^16))
- ihr_fwd_fees = ceil((msg_fwd_fees * ihr_price_factor)/2^16)
- // 메시지 루트 셀의 비트는 msg.bits에 포함되지 않습니다(lump_price가 해당 비용을 지불함).

행동_수수료

- action_fees = 합계(out_ext_msg_fwd_fee) + 합계(int_msg_mine_fee)

구성 파일

모든 수수료는 특정 가스 금액으로 지정되며 변경될 수 있습니다. 구성 파일은 현재 수수료 비용을 나타냅니다.
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD20362

56756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05

저장_수수료 =
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam18

in_fwd_fees =
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam24 ,
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam25

계산_수수료 =
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam20 ,
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam21

action_fees =
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam24 ,
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam25

out_fwd_fees =
https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam24 ,

https://explorer.toncoin.org/config?workchain=-1&shard=8000000000000000&seqno=22185244&roothash=165D55B3CFFC4043BFC43F81C1A3F2C41B69B33D6615D46FBFD2036256756382&filehash=69C43394D872B02C334B75F59464B2848CD4E23031C03CA7F3B1F98E8A13EE05#configparam25

원래 에 등장 https://t.me/thedailyton/79

3) 저는 **TON** 블록체인에 간단한 **DApp**을 작성하고 **tonweb JavaScript** 라이브러리를 사용하여 상호 작용하고 있습니다.

먼저 트랜잭션을 보낸 다음 체인에서 확인한 후 **JavaScript**로 다른 코드를 수행해야 합니다.

예:

```
await ton.send('ton_sendTransaction', [{
        to: 'some address',
        value: '1000'
    }]
)
// wait for tx to confirm on chain
console.log('Done!')
```

거래 확인을 기다리는 방법을 모르겠습니다.

답변

- 트랜잭션 해시를 보내기 전에 저장한 다음 Toncenter API 메서드 https://toncenter.com/api/index/#/default/get_transaction_by_in_message_hash_getTransactionByInMessageHash_get 를 쿼리하여 해당 해시가 포함된 트랜잭션이 확인되었는지 확인할 수 있습니다.

예

```
// Sleep function:
const sleep = ms => new Promise(r => setTimeout(r, ms))

// `msg` is a Cell containing your external message
// Convert message Cell to BOC String
const boc = await msg.toBoc(false)

// Calculate it's hash
const hash = tonweb.utils.bytesToBase64(await msg.hash())

// Send message and run a loop until transaction with that hash confirms
await tonweb.sendBoc(boc)
var txs = []
while (txs.length == 0) {
    await sleep(1200) // some delay between API calls
    const resp = await fetch('https://toncenter.com/api/index/getTransactionByInMessageHash?&include_
    txs = await resp.json()
}

console.log('Done!')
```

4) TON 코인을 보관하기 위해 **Ledger** 또는 **Trezor**와 같은 하드웨어 지갑을 사용하고 싶습니다**.** 그러나 불행하게도 둘 다 현재 **TON** 코인을 공식적으로 지원하지 않습니다**.** 하드웨어 지갑을 사용할 수 있는 방법이 있나요**?**

답변

안타깝게도 TON 생태계는 아직 초기 단계이기 때문에 주요 하드웨어 지갑에서는 TON을 공식적으로 지원하지 않습니다(2022년 10월 현재). 하지만 곧 공식적인 지원이 이루어지길 바랍니다.

Ledger에 대한 비공식 지원은 다음 두 곳에서 제공됩니다.

- https://github.com/ton-blockchain/ledger-app-ton (핵심 팀 기준)
- https://github.com/ton-community/ledger-app-ton (TonWhales 제공)

이 접근 방식의 단점은 Ledger에서 아직 공식적인 지원을 제공하지 않으며 일부 사람들은 하드웨어 장치에 비공식 앱을 설치하는 것을 좋아하지 않는다는 것입니다.