Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

«Блокчейн та децентралізовані системи»

Лабораторна робота №2

Реалізація смарт-контракту або анонімної криптовалюти

Виконали:
студенти групи ФБ-42мп
Косигін Олександр
Андреєв Данило
Зінов’єв Андрій

Київ — 2025

Мета

Отримання навичок роботи із смарт-контрактами або анонімними криптовалютами.

Завдання

1. Розгортання та запуск обраної анонімної валюти, протоколювання майнінгу, пошук слідів деанонімізації;
2. Розгортання та запуск обраного смарт-контракту, підвищення ефективності роботи смарт-контракту з точки зору витрати гасу;
3. Розробка власного смарт-контракту.

Хід роботи

Розгортання та запуск обраної анонімної валюти, протоколювання майнінгу, пошук слідів деанонімізації

Для виконання даної частини була використана віртуальна машина, розгорнута на Google Cloud, що суттєво допомагає під час синхронізації блокчейна
Налаштовуємо брандмауер

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ sudo ufw allow 18080/tcp
Rules updated
Rules updated (v6)
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ sudo ufw allow 18089/tcp
Rules updated
Rules updated (v6)
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$
```

Тут sudo ufw allow ssh - дозволяємо ssh підключення
sudo ufw allow 18080/tcp - дозволяємо порт
sudo ufw default deny incoming - забороняємо вхідний трафік
sudo ufw default allow outgoing - дозволяємо вихідний трафік
sudo ufw enable - вмикаємо брандмауер
Монтуємо образ monero на докер

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ docker run -d --restart unless-stopped --name="monerod" -p 1
8080:18080 -p 18089:18089 -v bitmonero:/home/monero ghcr.io/sethforprivacy/simple-monerod:latest --rpc-restricte
d-bind-ip=0.0.0.0 --rpc-restricted-bind-port=18089 --no-igd --no-zmq --enable-dns-blocklist
docker run -d \
    --name watchtower --restart unless-stopped \
    -v /var/run/docker.sock:/var/run/docker.sock \
    containrrr/watchtower --cleanup \
    monerod tor
Unable to find image 'ghcr.io/sethforprivacy/simple-monerod:latest' locally
latest: Pulling from sethforprivacy/simple-monerod
f18232174bc9: Pull complete
309d7f10bc40: Pull complete
b1bcba6b7d8d: Pull complete
d0a075a546f4: Pull complete
d10ad02bffa1: Pull complete
4806606efe31: Pull complete
c2084eeed32b: Pull complete
4f4fb700ef54: Pull complete
285fc49e0d92: Pull complete
f197e7b76551: Pull complete
Digest: sha256:eb143b50b427348f74b94d9342aaa6c9a1b4e74310cf1bfb1aa8623361ec20a1
Status: Downloaded newer image for ghcr.io/sethforprivacy/simple-monerod:latest
efc3aa090614e2ba05a8a24bef1346751ea55f1e4262a04b1e10935d40358d6d
Unable to find image 'containrrr/watchtower:latest' locally
latest: Pulling from containrrr/watchtower
57241801ebfd: Pull complete
3d4f475b92a2: Pull complete
1f05004da6d7: Pull complete
Digest: sha256:6dd50763bbd632a83cb154d5451700530d1e44200b268a4e9488fefdfcf2b038
Status: Downloaded newer image for containrrr/watchtower:latest
869164fd1e0e673510736c707fe1dc2754e1ea37f9cc5043d46c6d0c150cd9c2
```

Перевіряємо логи, у логах процес побудови ноди

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ docker logs --follow monerod
2025-04-13 19:19:37.741 I Monero 'Fluorine Fermi' (v0.18.4.0-release)
2025-04-13 19:19:37.741 I Initializing cryptonote protocol...
2025-04-13 19:19:37.741 I Cryptonote protocol initialized OK
2025-04-13 19:19:37.742 I Initializing core...
2025-04-13 19:19:37.742 I Loading blockchain from folder /home/monero/.bitmonero/lmdb ...
2025-04-13 19:19:37.844 I Loading checkpoints
2025-04-13 19:19:37.844 I Core initialized OK
2025-04-13 19:19:37.844 I Initializing p2p server...
2025-04-13 19:19:37.848 I p2p server initialized OK
2025-04-13 19:19:37.848 I Initializing core RPC server...
2025-04-13 19:19:37.848 I Binding on 127.0.0.1 (IPv4):18081
2025-04-13 19:19:40.483 I core RPC server initialized OK on port: 18081
2025-04-13 19:19:40.483 I Initializing restricted RPC server...
2025-04-13 19:19:40.483 I Binding on 0.0.0.0 (IPv4):18089
2025-04-13 19:19:41.177 I restricted RPC server initialized OK on port: 18089
2025-04-13 19:19:41.177 I Starting core RPC server...
2025-04-13 19:19:41.178 I core RPC server started ok
2025-04-13 19:19:41.178 I Starting restricted RPC server...
2025-04-13 19:19:41.178 I restricted RPC server started ok
2025-04-13 19:19:41.178 I Starting p2p net loop...
2025-04-13 19:19:42.178 I
2025-04-13 19:19:42.178 I ************************************************************************
2025-04-13 19:19:42.178 I The daemon will start synchronizing with the network. This may take a long time to com
plete.
2025-04-13 19:19:42.178 I
2025-04-13 19:19:42.178 I You can set the level of process detailization through "set_log <level|categories>" co
mmand,
2025-04-13 19:19:42.178 I where <level> is between 0 (no details) and 4 (very verbose), or custom category based
```

Додатково перевіряємо статус докера

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED         STATUS
                PORTS
   NAMES
869164fd1e0e   containrrr/watchtower                          "/watchtower --clean…"   2 minutes ago   Up 2 minu
tes (healthy)   8080/tcp
   watchtower
efc3aa090614   ghcr.io/sethforprivacy/simple-monerod:latest   "/entrypoint.sh --rp…"   2 minutes ago   Up 2 minu
tes (healthy)   0.0.0.0:18080->18080/tcp, [::]:18080->18080/tcp, 0.0.0.0:18089->18089/tcp, [::]:18089->18089/tcp
   monerod
```

Отже монеро розгорнута, чекаємо поки відбудеться синхронізація вузла з основоню мережею

Для запуску та протоколювання майнінгу потрібно отримати адресу гаманця, зробимо свій.

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~/monero-x86_64-linux-gnu-v0.18.4.0$ ./monero-wallet-cli --gene
rate-new-wallet mywallet1
This is the command line monero wallet. It needs to connect to a monero
daemon to work correctly.
WARNING: Do not reuse your Monero keys on another fork, UNLESS this fork has key reuse mitigations built in. Doi
ng so will harm your privacy.

Monero 'Fluorine Fermi' (v0.18.4.0-release)
Logging to ./monero-wallet-cli.log
Enter a new password for the wallet:
```

Вводимо пароль, отримаємо адресу, ключ, слова для повернення гаманця у разі втрати паролю. Перевіряємо статус гаманця.

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~/monero-x86_64-linux-gnu-v0.18.4.0$ ./monero-wallet-cli --wall
et-file mywallet
This is the command line monero wallet. It needs to connect to a monero
daemon to work correctly.
WARNING: Do not reuse your Monero keys on another fork, UNLESS this fork has key reuse mitigations built in. Doi
ng so will harm your privacy.

Monero 'Fluorine Fermi' (v0.18.4.0-release)
Logging to ./monero-wallet-cli.log
Wallet password:
Opened wallet: 47UuMLCKFtmUSN5RHK9Ge4eFGuExGeMGDcGP5KfwtDms61JDAVqASj52dsBoiTGVU3Ceg6fwhaSJbXn5S5Vi3HCR58e5k3L
**********************************************************************
Use the "help" command to see a simplified list of available commands.
Use "help all" to see the list of all available commands.
Use "help <command>" to see a command's documentation.
**********************************************************************
Background mining not enabled. Run "set setup-background-mining 1" to change.
Error: wallet failed to connect to daemon: http://localhost:18081. Daemon either is not started or wrong port wa
s passed. Please make sure daemon is running or change the daemon address using the 'set_daemon' command.
Background refresh thread started
[wallet 47UuML (no daemon)]: balance
Currently selected account: [0] Primary account
Tag: (No tag assigned)
Balance: 0.000000000000, unlocked balance: 0.000000000000
[wallet 47UuML (no daemon)]:
```

Гаманець створений та працює. Дивимося синхронізацію ноди

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ docker exec -it monerod monerod status
2025-04-14 22:24:14.517 I Monero 'Fluorine Fermi' (v0.18.4.0-release)
Height: 3390138/3390138 (100.0%) on mainnet, mining at 260 H/s, net hash 4.31 GH/s, v16, 12(out)+72(in) connecti
ons, uptime 1d 3h 4m 39s
```

Починаємо майнінг

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ docker exec -it monerod monerod start_mining 47UuMLCKFtmUSN5
RHK9Ge4eFGuExGeMGDcGP5KfwtDms61JDAVqASj52dsBoiTGVU3Ceg6fwhaSJbXn5S5Vi3HCR58e5k3L 1
2025-04-14 22:23:14.983 I Monero 'Fluorine Fermi' (v0.18.4.0-release)
Mining started
```

Тепер при перевірці статуса відображається процес майнінгу

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~$ docker exec -it monerod monerod status
2025-04-14 22:24:14.517 I Monero 'Fluorine Fermi' (v0.18.4.0-release)
Height: 3390138/3390138 (100.0%) on mainnet, mining at 260 H/s, net hash 4.31 GH/s, v16, 12(out)+72(in) connecti
ons, uptime 1d 3h 4m 39s
```

Подивимося логи, тепер є повідомлення про початок майнінгу

```
2025-04-14 22:01:36.203 I Host 69.48.204.218 unblocked.
2025-04-14 22:23:16.152 I Miner thread was started [0]
2025-04-14 22:54:27.459 I ----- BLOCK ADDED AS ALTERNATIVE ON HEIGHT 3390149
2025-04-14 22:54:27.461 I id:   <17b662c405e6bcd029735fbb6137c2d6beb9ccf93b83f45de845fe9d009f012a>
2025-04-14 22:54:27.462 I PoW:  <42a60306fd2b351b075ffeef08a3d9ff2e29980eb185bcda8176690000000000>
2025-04-14 22:54:27.462 I difficulty:   507411337413
2025-04-14 22:54:39.912 I ###### REORGANIZE on height: 3390149 of 3390149 with cum_difficulty 444927082323848716
2025-04-14 22:54:39.912 I  alternative blockchain size: 2 with cum_difficulty 444927589702147318
2025-04-14 22:54:41.332 I ----- BLOCK ADDED AS ALTERNATIVE ON HEIGHT 3390149
2025-04-14 22:54:41.332 I id:   <3bcf472ee72736f99a4adf6cb1f2813ccd2d3f7f7b5f0854ea2e4b7a3eaad540>
2025-04-14 22:54:41.332 I PoW:  <c6ff43574a912d8366853803058509c995f80dbde505ceaf68b8110100000000>
2025-04-14 22:54:41.332 I difficulty:   507411337413
2025-04-14 22:54:44.092 I REORGANIZE SUCCESS! on height: 3390149, new blockchain size: 3390151
```

У сучасних рішеннях комплекти “з коробки” вже мають вбудовані методи анонімізації, але якщо користувач хоче підвищити цей рівень, то можна розгорнути TOR та запускати монеро через TOR.
Завантажуємо тор, вмикаємо

```
olekos_ipt24_lll_kpi_ua@instance-20250413-184727:~/monero-x86_64-linux-gnu-v0.18.4.0$ sudo systemctl status tor
● tor.service - Anonymizing overlay network for TCP (multi-instance-master)
     Loaded: loaded (/lib/systemd/system/tor.service; enabled; preset: enabled)
     Active: active (exited) since Mon 2025-04-14 23:15:24 UTC; 1min 6s ago
    Process: 274129 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 274129 (code=exited, status=0/SUCCESS)
        CPU: 2ms

Apr 14 23:15:24 instance-20250413-184727 systemd[1]: Starting tor.service - Anonymizing overlay network for TCP>
Apr 14 23:15:24 instance-20250413-184727 systemd[1]: Finished tor.service - Anonymizing overlay network for TCP>
lines 1-9/9 (END)
```

У конфіг тора додаємо папку (якщо локально) та порти, які потрібно анонимізувати

```
  GNU nano 7.2                                  /etc/tor/torrc *
##
## If certain IPs and ports are blocked externally, e.g. by your firewall,
## you should update your exit policy to reflect this -- otherwise Tor
## users will be told that those destinations are down.
##
## For security, by default Tor rejects connections to private (local)
## networks, including to your public IP address. See the man page entry
## for ExitPolicyRejectPrivate if you want to allow "exit enclaving".
##
#ExitPolicy accept *:6660-6667,reject *:* # allow irc ports but no more
#ExitPolicy accept *:119 # accept nntp as well as default exit policy
#ExitPolicy reject *:* # no exits allowed

## Bridge relays (or "bridges") are Tor relays that aren't listed in the
## main directory. Since there is no complete public list of them, even an
## ISP that filters connections to all the known Tor relays probably
## won't be able to block all the bridges. Also, websites won't treat you
## differently because they won't know you're running Tor. If you can
## be a real relay, please do; but if not, be a bridge!
#BridgeRelay 1
## By default, Tor will advertise your bridge to users through various
## mechanisms like https://bridges.torproject.org/. If you want to run
## a private bridge, for example because you'll give out your bridge
## address manually to your friends, uncomment this line:
#PublishServerDescriptor 0

HiddenServiceDir /var/lib/tor/monero-service/
HiddenServiceVersion 3
HiddenServicePort 18081 127.0.0.1:18081
```

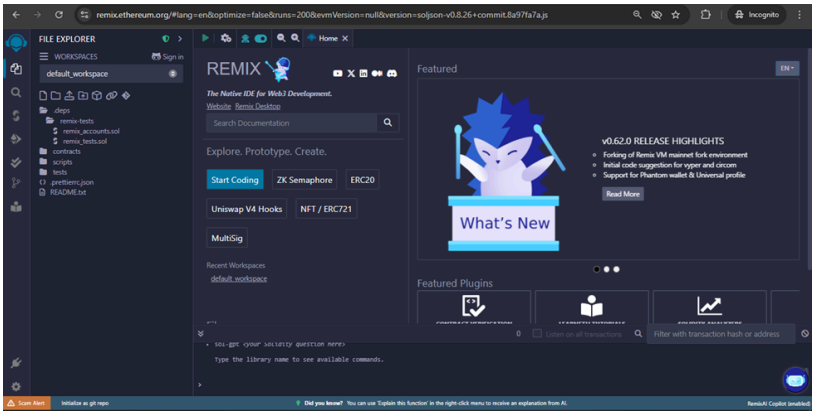
Отримуємо оніон адресу. За допомогою даної адреси можна отримати доступ до ноди не використовуючи власний айпі
За потреби анонімізації запускаємо монеро таким чином

```
monerod \

  --anonymous-inbound 127.0.0.1:18081,127.0.0.1:9050 \

  --tx-proxy tor,127.0.0.1:9050,23 \

  --hide-my-port \

  --restricted-rpc \

  --rpc-bind-ip 127.0.0.1 \

  --rpc-bind-port 18081
```

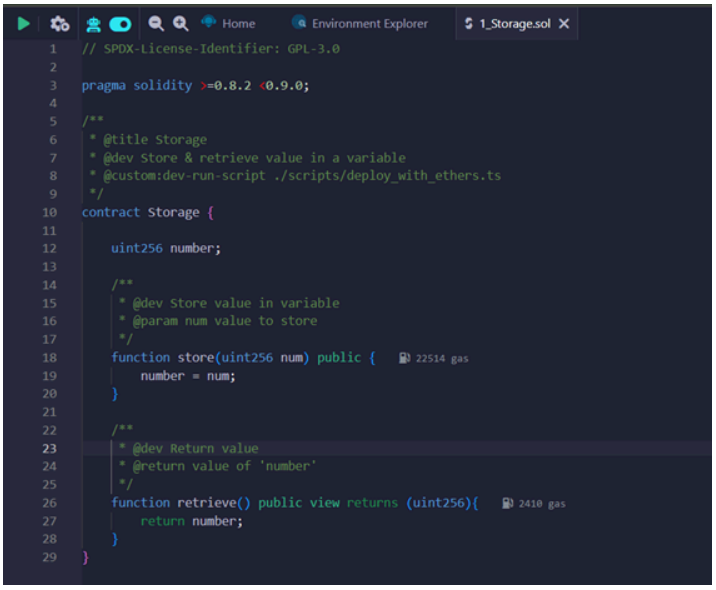
Також для пошука слідів деанонімізації можна використовувати мережеві утиліти такі як Wireshark / tcpdump, регулярно перевіряти логи (в нашому випадку докер контейнера).

Розгортання та запуск обраного смарт-контракту

Для виконання наступного завдання використовували інструмент [Remix IDE]



Створимо файл 1_Storage.sol. Скрипт виконує ініціалізацію (створення) змінної для збереження та отримання певного значення. Візьмемо його ідею за основу.

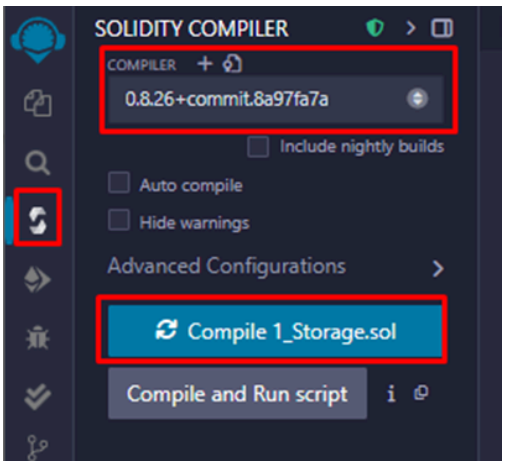


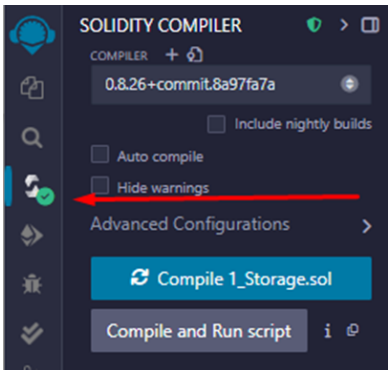
Але він виглядає дуже просто і тут немає що покращувати. Тому можемо модифікувати код, написати додатковий функціонал, щоб потім продемонструвати покращення з точки зору витрати гасу. Модифікований код продемонстрований нижче.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.2 <0.9.0;
contract Storage {
    uint256 private number;
   string private text;
   address private owner;
```

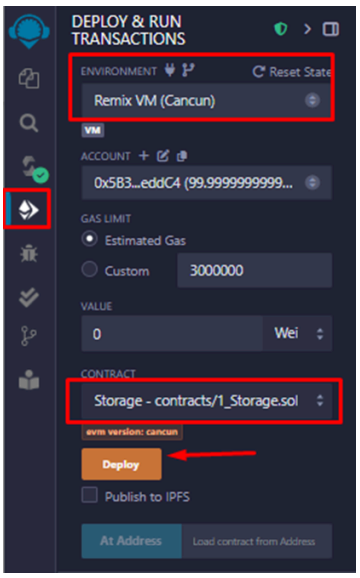
```
    event ValueUpdated(string valueType, uint256 newValue);
    event TextUpdated(string newText);
    event OwnershipTransferred(address indexed oldOwner, address indexed newOwner);

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can execute this");
        _;
    }
    constructor() {
        owner = msg.sender;
    }
    function storeNumber(uint256 num) public {
        require(num >= 0 && num <= 1000, "Value out of range");
        number = num;
        emit ValueUpdated("Number", num);
    }

    function storeText(string memory newText) public {
        text = newText;
        emit TextUpdated(newText);
    }
    function retrieveNumber() public view returns (uint256) {
        return number;
    }
    function retrieveText() public view returns (string memory) {
        return text;
    }
    function transferOwnership(address newOwner) public onlyOwner {
        require(newOwner != address(0), "Invalid address");
        address oldOwner = owner;
        owner = newOwner;
        emit OwnershipTransferred(oldOwner, newOwner);
    }
     function storePositiveNumber(uint256 num) public {
        require(num > 0, "Number must be positive");
        number = num;
        emit ValueUpdated("Positive Number", num);
    }
}
```

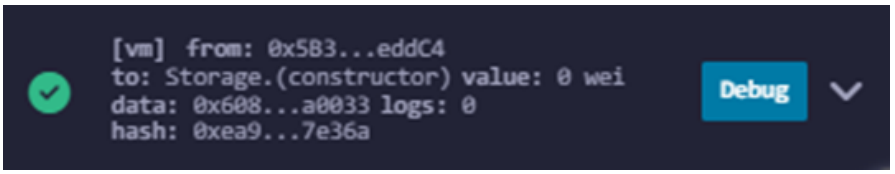
Компілювання смарт-контракту. Тут ми обираємо версію компайлера та наш скрипт. Зелена галочка при компілюванні повідомляє про успіх.



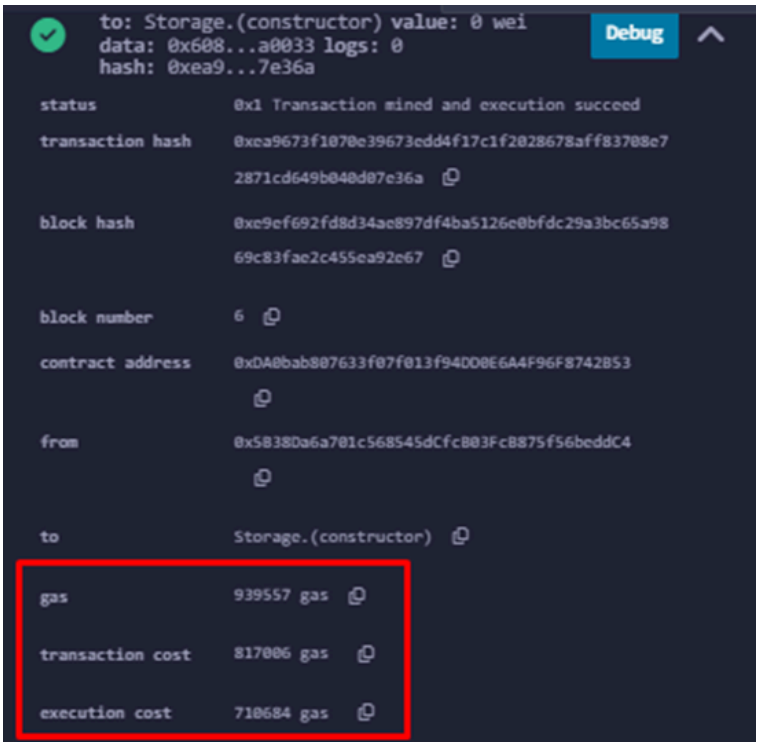


Наступним етапом є “Розгортання і запуск” смарт-контракту. За допомогою Remix IDE обираємо середовище запуску, за замовчуванням рахунок береться першим (бачимо запис 0x5B3…eddC4). Обираємо наш скрипт та робимо Deploy.

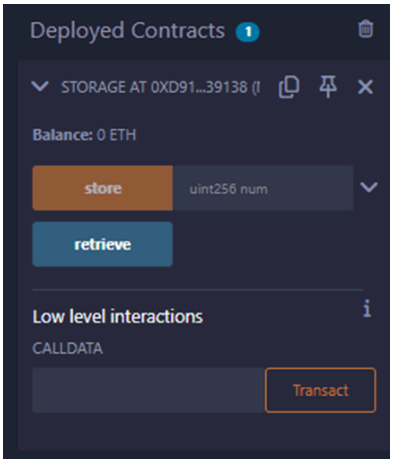




Після успішного розгортання, якщо переглянути подробиці, можна побачити інформацію про стан операції, хеші, номер блоку, адреси і найголовніше – це кількість затраченого “гасу”.

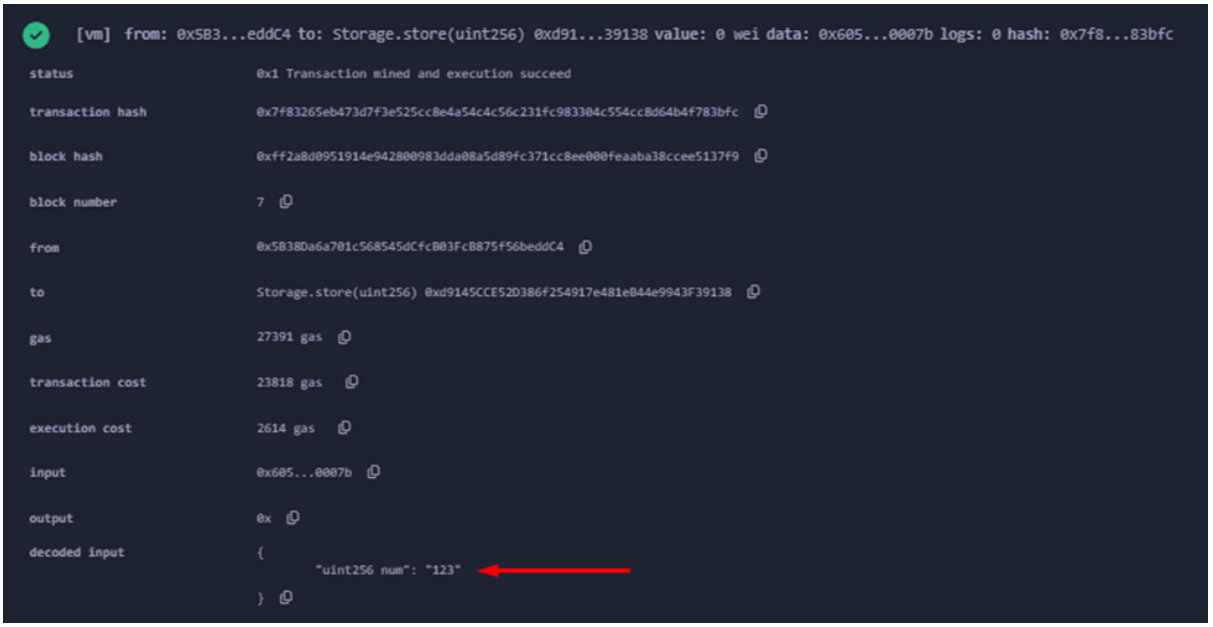


Remix IDE автоматично створив меню (інтерфейс) для нашого смарт-контракту. Маємо назву контракту (Storage) та адресу, за якою він задеплоєний. Баланс маємо 0 ETH, також згадується функція для збереження числа в контракті — store (uint256 num), де ми можемо вписати число і воно має зберегти його до блокчейну. Функція retrieve повертатиме це число.

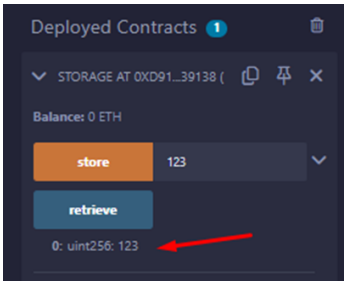


Скріншоти нижче вже демонструють принцип роботи. Спочатку запишемо число.

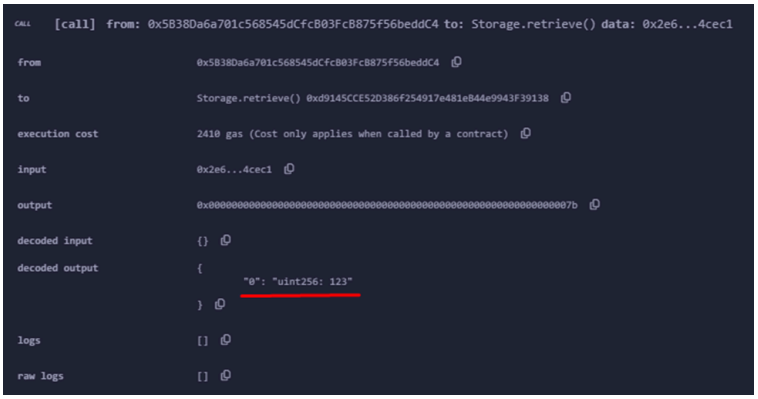
Натиснули “Store”. Отримуємо сповіщення про успішність операції, у деталях зазначається про статус операції, хеші, адреси, кількість “газу”. Бачимо в decoded input, що значення змінної uint256 змінилось на 123.



Якщо зробити retrieve, то в меню відображатиметься число, а саме: номер блоку, назва змінної, значення змінної.



Те саме і спостерігаємо у деталях про успішну операцію Retrieve.



Наступним етапом даного завдання буде оптимізація коду.
Gas – це величина яка показує скільки витратиться ресурсів на запуск скрипту, іншими словами, величина що може показати складність нашого алгоритму у смарт-контракті. І чим воно менше, тим алгоритм ощадливіше використовує ресурси системи.

Таким чином, ми для оптимізації вирішили прибрати зайву функцію, наприклад storePositiveNumber. Вона перевіряла число на те, чи є воно додатним.

```
function storePositiveNumber(uint256 num) public {
        require(num > 0, "Number must be positive");
        number = num;
        emit ValueUpdated("Positive Number", num);
    }
```

Отримали наступні значення:

	До оптимізації	Після оптимізації
gas	939557	936574
transaction cost	817006	814412
execution cost	710684	708284

Отже, з точки зору витрат “газу”, у нас алгоритм покращився, а значить система смарт-контракт обробить швидше.

Розробка власного смарт-контракту

Спочатку встановлюємо необхідні інструменти для розробки — nodejs, npm, git.

```
lingvvarm@ubuntu-lab:~$ sudo apt install nodejs npm git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.2).
The following additional packages will be installed:
  build-essential bzip2 dpkg-dev eslint fakeroot gyp handlebars javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libdpkg-perl libfakeroot libfile-fcntllock-perl libjs-async
  libjs-events libjs-inherits libjs-is-typedarray libjs-prettify libjs-regenerate libjs-source-map libjs-sprintf-js
  libjs-typedarray-to-buffer libjs-util libnode-dev libnode109 libre2-10 libssl-dev libuv1-dev lto-disabled-list
  node-abbrev node-acorn node-agent-base node-ajv node-ajv-keywords node-ampproject-remapping node-ansi-escapes
  node-ansi-regex node-ansi-styles node-anymatch node-aproba node-archy node-are-we-there-yet node-argparse
  node-arrify node-assert node-async node-async-each node-auto-bind node-babel-helper-define-polyfill-provider
  node-babel-plugin-add-module-exports node-babel-plugin-lodash node-babel-plugin-polyfill-corejs2
  node-babel-plugin-polyfill-corejs3 node-babel-plugin-polyfill-regenerator node-babel7 node-babel7-runtime
  node-balanced-match node-base node-base64-js node-binary-extensions node-brace-expansion node-braces
```

Встановлюємо пакет n — інструмент для управління версіями nodejs.

```
lingvvarm@ubuntu-lab:~$ sudo npm install -g n

added 1 package in 6s
lingvvarm@ubuntu-lab:~$
```

Встановлюємо nodejs за допомогою n.

```
lingvvarm@ubuntu-lab:~$ sudo n stable
  installing : node-v22.14.0
       mkdir : /usr/local/n/versions/node/22.14.0
       fetch : https://nodejs.org/dist/v22.14.0/node-v22.14.0-linux-x64.tar.xz
#######################################
```

Створюємо директорію проєкту та ініціалізуємо його за допомогою npm init.

```
lingvvarm@ubuntu-lab:~/Documents$ mkdir blockchain-lab2
lingvvarm@ubuntu-lab:~/Documents$ cd blockchain-lab2/
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npm init -y
Wrote to /home/lingvvarm/Documents/blockchain-lab2/package.json:

{
  "name": "blockchain-lab2",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}



lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$
```

Встановлюємо hardhat — фреймворк для створення та тестування власних смарт контрактів.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npm install --save-dev hardhat
npm warn deprecated glob@8.1.0: Glob versions prior to v9 are no longer supported
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache i
f you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerfu
l.

added 220 packages, and audited 221 packages in 57s

63 packages are looking for funding
  run `npm fund` for details

3 low severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$
```

Ініціалізуємо проєкт за допомогою hardhat. Вибираємо JavaScript project, всі інші налаштування залишаємо за замовчуванням.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npx hardhat
888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888

Welcome to Hardhat v2.23.0

✔ What do you want to do? · Create a JavaScript project
✔ Hardhat project root: · /home/lingvvarm/Documents/blockchain-lab2
✔ Do you want to add a .gitignore? (Y/n) · y
✔ Help us improve Hardhat with anonymous crash reports & basic usage data? (Y/n) · y
✔ Do you want to install this sample project's dependencies with npm (@nomicfoundation/hardhat-toolbox)? (Y/n) · y

npm install --save-dev "@nomicfoundation/hardhat-toolbox@^5.0.0"
:
```

Йде автоматична ініціалізація проєкту.

```
npm install --save-dev "@nomicfoundation/hardhat-toolbox@^5.0.0"
npm warn deprecated lodash.isequal@4.5.0: This package is deprecated. Use require('node:util').isDeepStrictEqual instead
.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@5.0.15: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.1.7: Glob versions prior to v9 are no longer supported

added 345 packages, and audited 566 packages in 55s

102 packages are looking for funding
  run `npm fund` for details

13 low severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.

Project created

See the README.md file for some example tasks you can run

Give Hardhat a star on Github if you're enjoying it!

     https://github.com/NomicFoundation/hardhat
```

Тепер створюємо наш смарт-контракт — SimpleStorage. Це простий контракт який може має 2 функції — читати та писати значення гаманця. Для написання використовуємо мову Solidity.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/contracts$ nano SimpleStorage.sol
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/contracts$ cat SimpleStorage.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleStorage {
    uint256 public storedValue;

    function set(uint256 _value) public {
        storedValue = _value;
    }

    function get() public view returns (uint256) {
        return storedValue;
    }
}
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/contracts$
```

Створюємо директорію scripts.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ mkdir scripts
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ cd scripts/
```

Створимо скрипт для деплою.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/scripts$ cat deploy.js
async function main() {
  const [deployer] = await ethers.getSigners();

  console.log("Deploying contract with the account:", deployer.address);

  const SimpleStorage = await ethers.getContractFactory("SimpleStorage");
  const storage = await SimpleStorage.deploy();
  await storage.waitForDeployment();

  console.log("Contract deployed to:", await storage.getAddress());
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/scripts$
```

Тепер піднімаємо testnet для перевірки. Було автоматично створено декілька тестових гаманців з 10000 ETH в кожному.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/scripts$ npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
========

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0x90F79bf6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4: 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65 (10000 ETH)
Private Key: 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a
```

Тепер запускаємо скрипт для деплою. Бачимо адресу, за якою було задеплоєно смарт-контракт.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npx hardhat run scripts/deploy.js --network localhost
Deploying contract with the account: 0xf39Fd6e51aad88F6F4ce6aB8827279cfffb92266
Contract deployed to: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$
```

Підключаємось до консолі для тестування смарт-контракту.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npx hardhat console --network localhost
Welcome to Node.js v22.14.0.
Type ".help" for more information.
> const SimpleStorage = await ethers.getContractFactory("SimpleStorage")
undefined
> const storage = await SimpleStorage.attach("0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512")
undefined
> await storage.set(42)
ContractTransactionResponse {
  provider: HardhatEthersProvider {
    _hardhatProvider: LazyInitializationProviderAdapter {
      _providerFactory: [AsyncFunction (anonymous)],
      _emitter: [EventEmitter],
      _initializingPromise: [Promise],
      provider: [BackwardsCompatibilityProviderAdapter]
    },
    _networkName: 'localhost',
    _blockListeners: [],
    _transactionHashListeners: Map(0) {},
    _eventListeners: []
  },
  blockNumber: 3,
  blockHash: '0x5aa93cc6e9632bef24744ea04316407c0a9c005d3ca9709b99eb4e1fc765cdb3',
  index: undefined,
  hash: '0xc8563d9aa0439a1de48cbfd5b8846a01ff70c973774162bb77c1c5020ba16e65',
  type: 2,
  to: '0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512',
  from: '0xf39Fd6e51aad88F6F4ce6aB8827279cfffb92266',
  nonce: 2,
  gasLimit: 30000000n,
```

```
  gasLimit: 30000000n,
  gasPrice: 850065176n,
  maxPriorityFeePerGas: 203632252n,
  maxFeePerGas: 850065176n,
  maxFeePerBlobGas: null,
  data: '0x60fe47b1000000000000000000000000000000000000000000000000000000000000002a',
  value: 0n,
  chainId: 31337n,
  signature: Signature { r: "0x0fd46b4d6c74dd8d6c97203f1a8ebe93f10b73c195b044c00f38b7bc4ec5ab5c", s: "0x2472ba2b023b9ca5
ed23c1add6a34bd17d05b7a66593d3b04348d4d216d94529", yParity: 0, networkV: null },
  accessList: [],
  blobVersionedHashes: null
}
> const value = await storage.get()
undefined
> console.log(value.toString())
42
undefined
>
```

Бачимо, що функції get та set працюють.

Також бачимо повідомлення про операції в логах нашої тестової мережі.

```
eth_feeHistory
eth_sendTransaction
  Contract call:       <UnrecognizedContract>
  Transaction:         0xc8563d9aa0439a1de48cbfd5b8846a01ff70c973774162bb77c1c5020ba16e65
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
  Value:               0 ETH
  Gas used:            43702 of 30000000
  Block #3:            0x5aa93cc6e9632bef24744ea04316407c0a9c005d3ca9709b99eb4e1fc765cdb3

eth_getTransactionByHash
eth_call
  Contract call:       <UnrecognizedContract>
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
```

Тепер створимо ускладнену версію смарт-контракту, додавши до неї декілька нових функцій.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/contracts$ cat SimpleStorage.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleStorage {
    uint256 public storedValue;
    uint256[] private history;

    constructor(uint256 _initialValue) {
        storedValue = _initialValue;
        history.push(_initialValue);
    }

    function set(uint256 _value) public {
        storedValue = _value;
        history.push(_value);
    }

    function get() public view returns (uint256) {
        return storedValue;
    }
```

```
    function increment() public {
        storedValue += 1;
        history.push(storedValue);
    }

    function multiply(uint256 _multiplier) public {
        storedValue *= _multiplier;
        history.push(storedValue);
    }

    function reset() public {
        storedValue = 0;
        history.push(0);
    }

    function getHistory() public view returns (uint256[] memory) {
        return history;
    }
}
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2/contracts$
```

Трохи модифікуємо функцію деплою.

```
  GNU nano 7.2                                                  deploy.js *
async function main() {
  const [deployer] = await ethers.getSigners();

  console.log("Deploying contract with the account:", deployer.address);

  const SimpleStorage = await ethers.getContractFactory("SimpleStorage");
  const storage = await SimpleStorage.deploy(100);
  await storage.waitForDeployment();

  console.log("Contract deployed to:", await storage.getAddress());
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Знову запускаємо деплой.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npx hardhat run scripts/deploy.js --network localhost
Deploying contract with the account: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Contract deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$
```

Підключаємось та перевіряємо функції смарт-контракту.

```
lingvvarm@ubuntu-lab:~/Documents/blockchain-lab2$ npx hardhat console --network localhost
Welcome to Node.js v22.14.0.
Type ".help" for more information.
> const SimpleStorage = await ethers.getContractFactory("SimpleStorage")
undefined
> const storage = await SimpleStorage.attach("0x5FbDB2315678afecb367f032d93F642f64180aa3")
undefined
> (await storage.get()).toString()
'100'
> await storage.set(42)
ContractTransactionResponse {
  provider: HardhatEthersProvider {
    _hardhatProvider: LazyInitializationProviderAdapter {
      _providerFactory: [AsyncFunction (anonymous)],
      _emitter: [EventEmitter],
      _initializingPromise: [Promise],
      provider: [BackwardsCompatibilityProviderAdapter]
    },
    _networkName: 'localhost',
    _blockListeners: [],
    _transactionHashListeners: Map(0) {},
    _eventListeners: []
  },
  blockNumber: 2,
  blockHash: '0x35cd98ffdce2b7a487312d5f8a10f104a26bd17755e55b50805937dde91319e1',
  index: undefined,
  hash: '0xa32a40b0323eb856a7062431a14324b3969115c7ceafba5577028ae73b468757',
  type: 2
```

```
      _initializingPromise: [Promise],
      provider: [BackwardsCompatibilityProviderAdapter]
    },
    _networkName: 'localhost',
    _blockListeners: [],
    _transactionHashListeners: Map(0) {},
    _eventListeners: []
  },
  blockNumber: 2,
  blockHash: '0x35cd98ffdce2b7a487312d5f8a10f104a26bd17755e55b50805937dde91319e1',
  index: undefined,
  hash: '0xa32a40b0323eb856a7062431a14324b3969115c7ceafba5577028ae73b468757',
  type: 2,
  to: '0x5FbDB2315678afecb367f032d93F642f64180aa3',
  from: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266',
  nonce: 1,
  gasLimit: 30000000n,
  gasPrice: 972574279n,
  maxPriorityFeePerGas: 232421875n,
  maxFeePerGas: 972574279n,
  maxFeePerBlobGas: null,
  data: '0x60fe47b1000000000000000000000000000000000000000000000000000000000000002a',
  value: 0n,
  chainId: 31337n,
  signature: Signature { r: "0x9b4efafdacf426089f7190f8e59084baf3746947a94ccc95621413d009ccc3ab", s: "0x2b28e87353a478da
2dea648c6f6ae54216d279c31f1c923ce74f11523c7ea692", yParity: 0, networkV: null },
  accessList: [],
  blobVersionedHashes: null
}
>
```

```
> (await storage.get()).toString()
'42'
>
```

```
> await storage.increment()
ContractTransactionResponse {
  provider: HardhatEthersProvider {
    _hardhatProvider: LazyInitializationProviderAdapter {
      _providerFactory: [AsyncFunction (anonymous)],
      _emitter: [EventEmitter],
      _initializingPromise: [Promise],
      provider: [BackwardsCompatibilityProviderAdapter]
    },
```

```
  blobVersionedHashes: null
}
> (await storage.get()).toString()
'43'
>
```

```
> await storage.multiply(2)
ContractTransactionResponse {
  provider: HardhatEthersProvider {
    _hardhatProvider: LazyInitializationProviderAdapter {
      _providerFactory: [AsyncFunction (anonymous)],
      _emitter: [EventEmitter],
      _initializingPromise: [Promise],
      provider: [BackwardsCompatibilityProviderAdapter]
    },
```

```
> (await storage.get()).toString()
'86'
>
```

```
> await storage.reset()
ContractTransactionResponse {
  provider: HardhatEthersProvider {
    _hardhatProvider: LazyInitializationProviderAdapter {
      _providerFactory: [AsyncFunction (anonymous)],
      _emitter: [EventEmitter],
      _initializingPromise: [Promise],
      provider: [BackwardsCompatibilityProviderAdapter]
    },
```

```
> (await storage.get()).toString()
'0'
>
```

```
> const history = await storage.getHistory()
undefined
> history.map(x => x.toString())
[ '100', '42', '43', '86', '0' ]
>
```

Бачимо, що всі функції працюють успішно. Знову ж таки бачимо всі операції в консолі.

```
eth_feeHistory
eth_maxPriorityFeePerGas
eth_sendTransaction
  Contract deployment: SimpleStorage
  Contract address:    0x5fbdb2315678afecb367f032d93f642f64180aa3
  Transaction:         0x0ec476597a1ef5c143451c058250df9354be0caad7672e1f10ab3578abbdfa93
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  Value:               0 ETH
  Gas used:            387943 of 30000000
  Block #1:            0x004e0dccd343ac94480aeae686278a183ce333634dfb6041a5df2d09cd23af6d

eth_getTransactionByHash
eth_getTransactionReceipt
eth_blockNumber
eth_accounts
hardhat_metadata (20)
eth_call
  Contract call:       SimpleStorage#get
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3

eth_blockNumber
eth_getBlockByNumber
eth_feeHistory
eth_sendTransaction
  Contract call:       SimpleStorage#set
  Transaction:         0xa32a40b0323eb856a7062431a14324b3969115c7ceafba5577028ae73b468757
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            53848 of 30000000
```

```
eth_getTransactionByHash
eth_call
  Contract call:       SimpleStorage#get
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3

eth_blockNumber
eth_feeHistory
eth_sendTransaction
  Contract call:       SimpleStorage#increment
  Transaction:         0x71f3e9339d2a23f72dfea06acc59955bca138de04feb1fc179d8e536eab53f0d
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            53790 of 30000000
  Block #3:            0x287e5eb196ebfb4f7b947e5efcbe5eb203d6ddd188c3db5d495a142337f8aead

eth_getTransactionByHash
eth_call
  Contract call:       SimpleStorage#get
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3

eth_blockNumber
```

```
eth_feeHistory
eth_sendTransaction
  Contract call:       SimpleStorage#multiply
  Transaction:         0x116c89e3228d1da4c1459ecca526b1f3a4be43372655d8d88719359d7f69e596
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            54234 of 30000000
  Block #4:            0xb4fe5b6096dfcc0874e0bd3eda58239c578c86321963704890d3ef6cfeb6d3df

eth_getTransactionByHash
eth_call
  Contract call:       SimpleStorage#get
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3

eth_blockNumber
eth_feeHistory
eth_sendTransaction
  Contract call:       SimpleStorage#reset
  Transaction:         0x1741fe7d672097572342492da2d61d87b606e4ccca61ef67fc0dc4718b67ddb2
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            28811 of 30000000
  Block #5:            0xe8f6e651c9fdd0cc717b8982cefa0c38a7837a16fc29d426f6b0c6cc4e535465
```

```
eth_getTransactionByHash
eth_call
  Contract call:       SimpleStorage#get
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3

eth_call
  Contract call:       SimpleStorage#getHistory
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
```

Висновки

В ході виконання лабораторної роботи було розгорнуто власну ноду блокчейну в хмарному середовищі для валюти Monero, також було досліджено способи оптимізації кількості витраченого гасу на виконання смарт контрактів. Крім цього, було створено власний смарт контракт за допомогою фреймворку Hardhat і перевірено його роботу в тестовій мережі на локальній машині.