Solidity 智能合约开发 - 玩转 Web3.py

前言

在前文《Solidity 智能合约开发 - 基础》中，我们学习了 Solidity 的基本语法，并且了解了可以通过 Brownie 与 HardHat 等框架进行调试。但在使用这些封装好的框架之前，我们可以通过 Web3.py 直接与我们本地的 Ganache 节点进行交互，以便更好了解其原理，也为我们后续更好使用框架打好基础。

本文以 Web3.py 为例，实现了基础的合约编译、部署至本地 Ganache 网络、与合约交互等功能。

可以点击这里访问本测试 Demo 代码仓库。

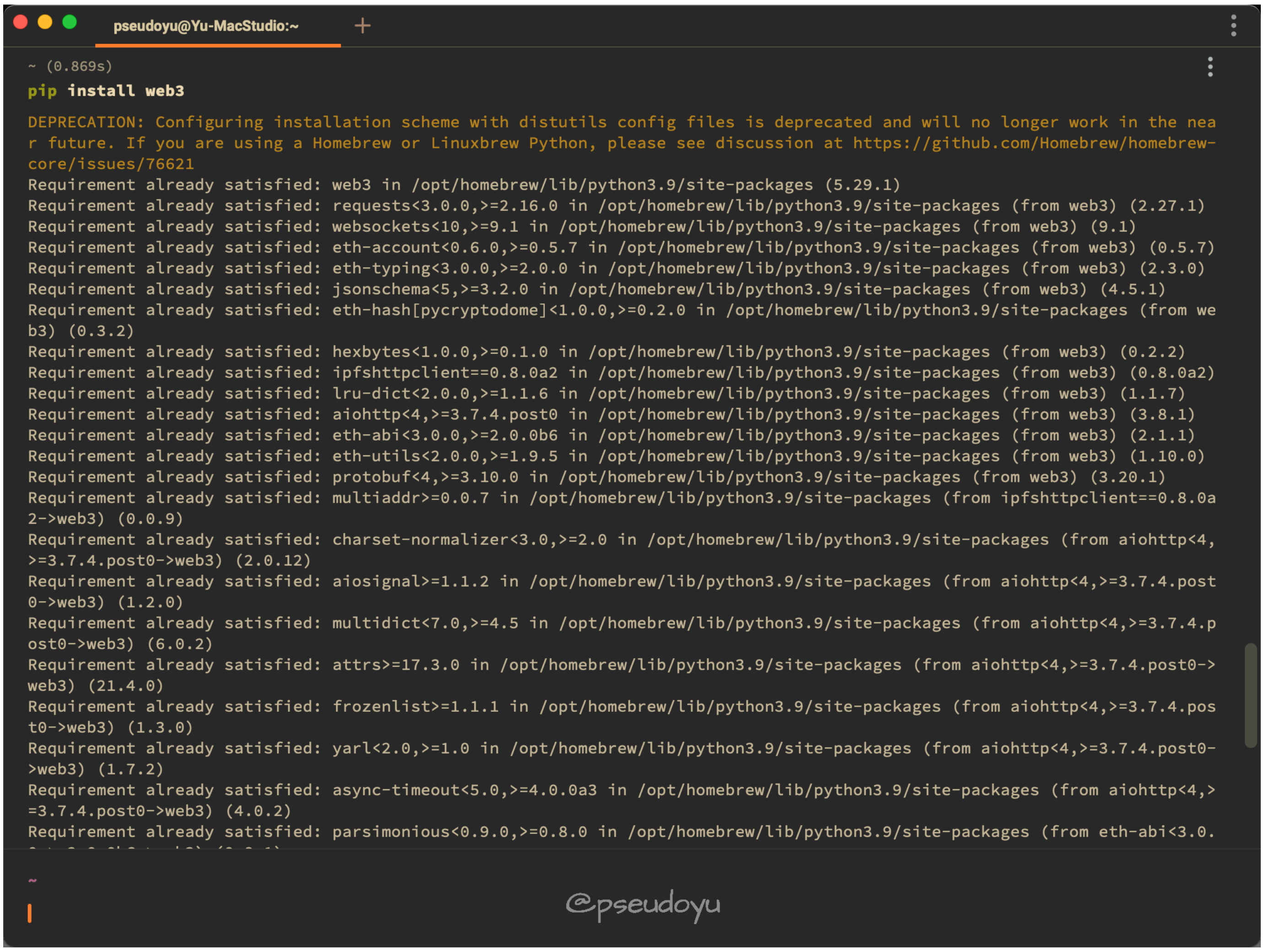
Web3.py

Web3.py 是 Python 的一个开源库，它提供了一个简单的 API，可以让我们通过 Python 程序与以太坊网络进行交互。其 GitHub 地址为 ethereum/web3.py，可以访问其官方文档进行使用。

安装

我们可以通过 Python 包管理工具 pip 安装 Web3.py，如下：

```
pip3 install web3
```



```
~ (0.869s)
pip install web3
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the nea
r future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-
core/issues/76621
Requirement already satisfied: web3 in /opt/homebrew/lib/python3.9/site-packages (5.29.1)
Requirement already satisfied: requests<3.0.0,>=2.16.0 in /opt/homebrew/lib/python3.9/site-packages (from web3) (2.27.1)
Requirement already satisfied: websockets<10,>=9.1 in /opt/homebrew/lib/python3.9/site-packages (from web3) (9.1)
Requirement already satisfied: eth-account<0.6.0,>=0.5.7 in /opt/homebrew/lib/python3.9/site-packages (from web3) (0.5.7)
Requirement already satisfied: eth-typing<3.0.0,>=2.0.0 in /opt/homebrew/lib/python3.9/site-packages (from web3) (2.3.0)
Requirement already satisfied: jsonschema<5,>=3.2.0 in /opt/homebrew/lib/python3.9/site-packages (from web3) (4.5.1)
Requirement already satisfied: eth-hash[pycryptodome]<1.0.0,>=0.2.0 in /opt/homebrew/lib/python3.9/site-packages (from we
b3) (0.3.2)
Requirement already satisfied: hexbytes<1.0.0,>=0.1.0 in /opt/homebrew/lib/python3.9/site-packages (from web3) (0.2.2)
Requirement already satisfied: ipfshttpclient==0.8.0a2 in /opt/homebrew/lib/python3.9/site-packages (from web3) (0.8.0a2)
Requirement already satisfied: lru-dict<2.0.0,>=1.1.6 in /opt/homebrew/lib/python3.9/site-packages (from web3) (1.1.7)
Requirement already satisfied: aiohttp<4,>=3.7.4.post0 in /opt/homebrew/lib/python3.9/site-packages (from web3) (3.8.1)
Requirement already satisfied: eth-abi<3.0.0,>=2.0.0b6 in /opt/homebrew/lib/python3.9/site-packages (from web3) (2.1.1)
Requirement already satisfied: eth-utils<2.0.0,>=1.9.5 in /opt/homebrew/lib/python3.9/site-packages (from web3) (1.10.0)
Requirement already satisfied: protobuf<4,>=3.10.0 in /opt/homebrew/lib/python3.9/site-packages (from web3) (3.20.1)
Requirement already satisfied: multiaddr>=0.0.7 in /opt/homebrew/lib/python3.9/site-packages (from ipfshttpclient==0.8.0a
2->web3) (0.0.9)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,
>=3.7.4.post0->web3) (2.0.12)
Requirement already satisfied: aiosignal>=1.1.2 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,>=3.7.4.post
0->web3) (1.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,>=3.7.4.p
ost0->web3) (6.0.2)
Requirement already satisfied: attrs>=17.3.0 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,>=3.7.4.post0->
web3) (21.4.0)
Requirement already satisfied: frozenlist>=1.1.1 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,>=3.7.4.pos
t0->web3) (1.3.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,>=3.7.4.post0-
>web3) (1.7.2)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /opt/homebrew/lib/python3.9/site-packages (from aiohttp<4,>
=3.7.4.post0->web3) (4.0.2)
Requirement already satisfied: parsimonious<0.9.0,>=0.8.0 in /opt/homebrew/lib/python3.9/site-packages (from eth-abi<3.0.

~
```

使用

使用 `import` 导入所需方法即可使用

```
from web3 import Web3

w3 = Web3(Web3.HTTPProvider("HTTP://127.0.0.1:7545"))
```

Solidity 合约编译

合约源码

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.6.0;

contract SimpleStorage {
    uint256 favoriteNumber;
    bool favoriteBool;

    struct People {
        uint256 favoriteNumber;
        string name;
    }

    People public person = People({favoriteNumber: 2, name: "Arthur"});

    People[] public people;

    mapping(string => uint256) public nameToFavoriteNumber;

    function store(uint256 _favoriteNumber) public returns (uint256) {
        favoriteNumber = _favoriteNumber;
        return favoriteNumber;
    }

    function retrieve() public view returns (uint256) {
        return favoriteNumber;
    }

    function addPerson(string memory _name, uint256 _favoriteNumber) public {
        people.push(People({favoriteNumber: _favoriteNumber, name: _name}));
        nameToFavoriteNumber[_name] = _favoriteNumber;
    }
}
```

这是一个简单的存储合约，通过一个 People 结构体对象来存储人名和他喜欢数字，通过一个数组来存储多个人的信息，并提供了添加、查找方法。

读取合约源文件

当我们通过 VSCode 或其他编辑器完成 Solidity 合约编写与语法检查后，需要读取合约源文件并存入变量，供后续编译使用。

```python
import os

with open("./SimpleStorage.sol", "r") as file:
    simple_storage_file = file.read()
```

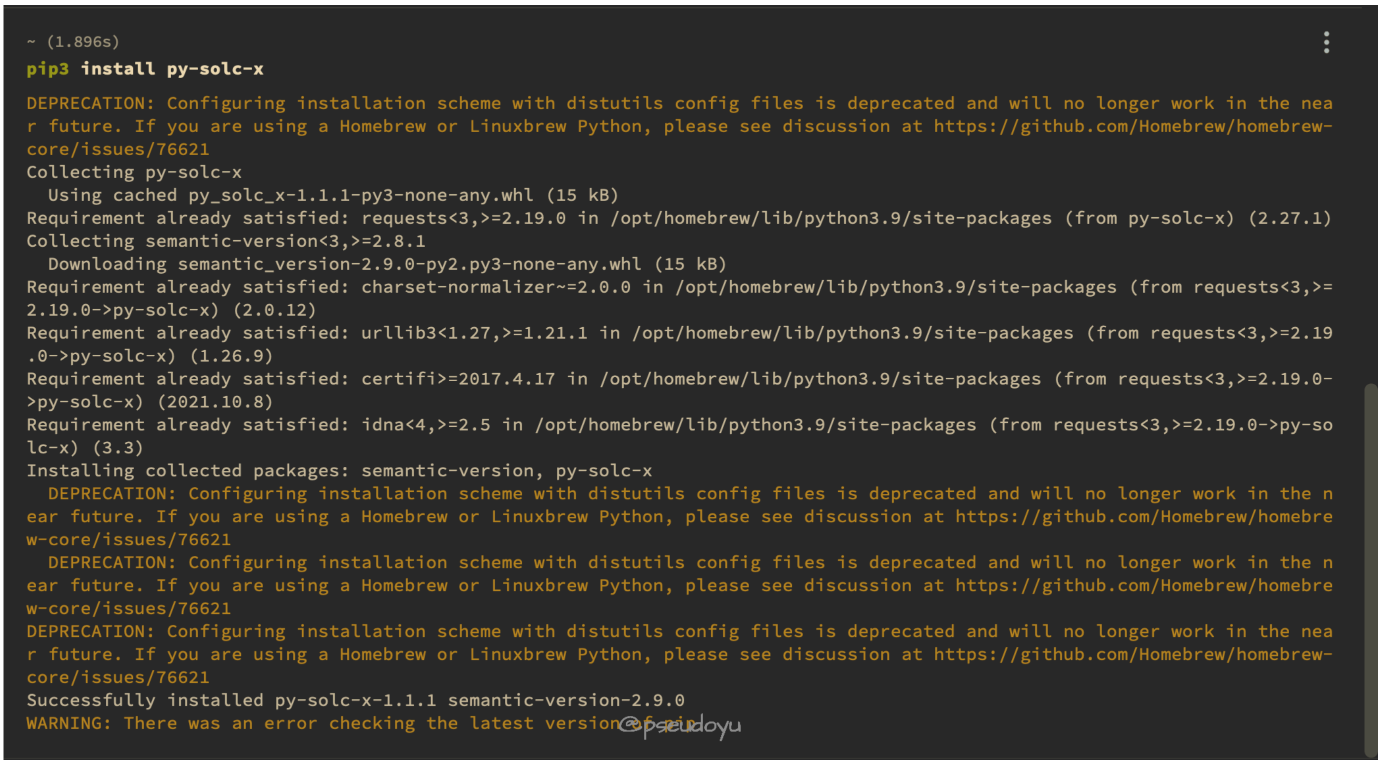
上述代码将 `SimpleStorage.sol` 文件内容读取到变量 `simple_storage_file` 中。

编译合约

安装 `solcx`

合约编译需要预先安装 `solcx` 工具。

```
pip3 install py-solc-x
```



```
~ (1.896s)
pip3 install py-solc-x
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the nea
r future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-
core/issues/76621
Collecting py-solc-x
  Using cached py_solc_x-1.1.1-py3-none-any.whl (15 kB)
Requirement already satisfied: requests<3,>=2.19.0 in /opt/homebrew/lib/python3.9/site-packages (from py-solc-x) (2.27.1)
Collecting semantic-version<3,>=2.8.1
  Downloading semantic_version-2.9.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/homebrew/lib/python3.9/site-packages (from requests<3,>=
2.19.0->py-solc-x) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/homebrew/lib/python3.9/site-packages (from requests<3,>=2.19
.0->py-solc-x) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in /opt/homebrew/lib/python3.9/site-packages (from requests<3,>=2.19.0-
>py-solc-x) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /opt/homebrew/lib/python3.9/site-packages (from requests<3,>=2.19.0->py-so
lc-x) (3.3)
Installing collected packages: semantic-version, py-solc-x
  DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the n
ear future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebre
w-core/issues/76621
  DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the n
ear future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebre
w-core/issues/76621
DEPRECATION: Configuring installation scheme with distutils config files is deprecated and will no longer work in the nea
r future. If you are using a Homebrew or Linuxbrew Python, please see discussion at https://github.com/Homebrew/homebrew-
core/issues/76621
Successfully installed py-solc-x-1.1.1 semantic-version-2.9.0
WARNING: There was an error checking the latest version of pip.
```

导入 `solcx`

使用 `import` 导入所需方法即可使用

```python
from solcx import compile_standard, install_solc
```

编译

```python
install_solc("0.6.0")
compiled_sol = compile_standard(
    {
        "language": "Solidity",
        "sources": {"SimpleStorage.sol": {"content": simple_storage_file}},
        "settings": {
            "outputSelection": {
                "*": {"*": ["abi", "metadata", "evm.bytecode", "evm.sourceMap
            }
        },
    },
    solc_version="0.6.0",
)
```

上述代码我们安装了 0.6.0 版本的 Solidity 编译程序，使用 `solcx` 库中的 `compile_standard` 方法对上文读取的合约源文件进行编译，并将编译结果存入变量 `compiled_sol` 中。

获取编译结果

编译成功后，使用以下代码将编译好的合约写入文件

```
import json

with open("compiled_code.json", "w") as file:
    json.dump(compiled_sol, file)
```

获取 bytecode 与 abi

Solidity 合约的部署与交互需要 bytecode 与 abi 两个部分，我们可以通过通过以下代码将其写入对应变量供后续操作使用。

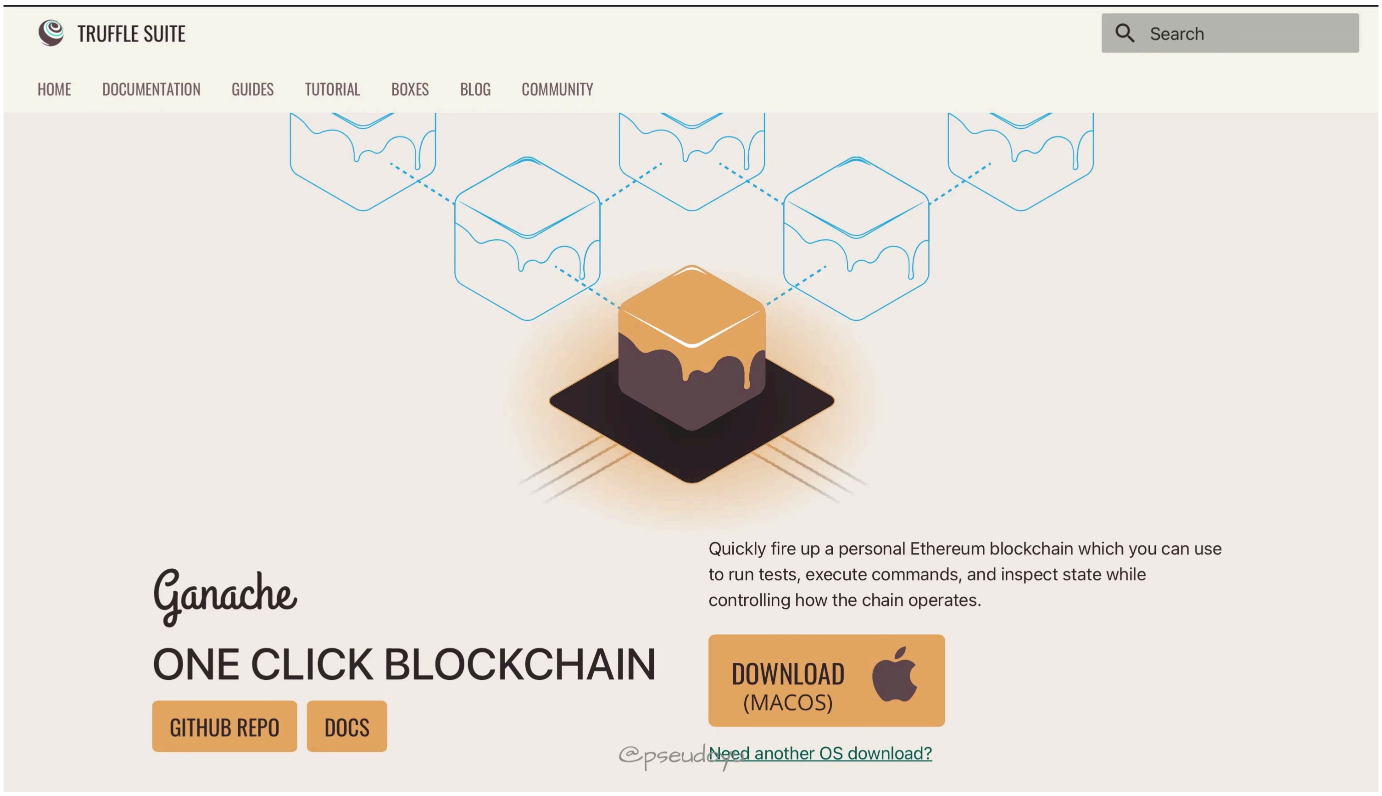
```
# get bytecode
bytecode = compiled_sol["contracts"]["SimpleStorage.sol"]["SimpleStorage"]["e
    "bytecode"
]["object"]

# get abi
abi = compiled_sol["contracts"]["SimpleStorage.sol"]["SimpleStorage"]["abi"]
```

本地 Ganache 环境

智能合约的调试需要将合约部署到实际的链上，而部署到 Ethereum 主网络或 Rinkeby/Koven 等测试网等也不方便调试，因此，我们需要一个本地的区块链环境，Ganache 就给我们提供了一个这样的本地调试环境。Ganache 主要分为 GUI 和 CLI 两种安装方式。

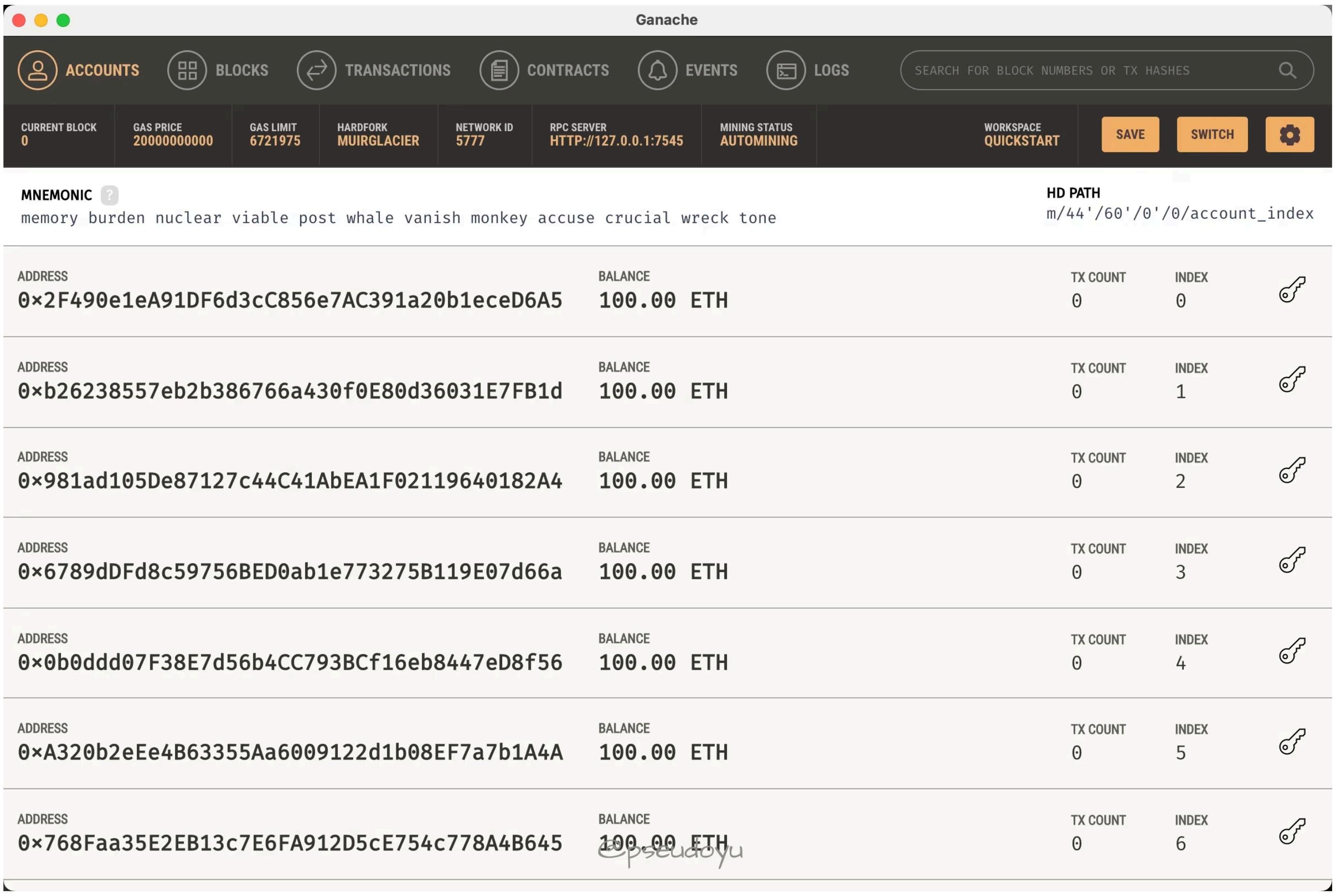
Ganache GUI

在自己的本地环境，如 Mac/Windows 等系统，我们可以选择带图形界面的 Ganache 客户端，安装与使用都十分便捷，在 Ganache 官网选择对应版本即可。



Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.

安装完成后选择 Quick Start 即可快速启动一条本地运行的区块链网络，并初始化了十个拥有 100 ETH 的账户，开发调试过程中可使用。



CURRENT BLOCK 0 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

MNEMONIC
memory burden nuclear viable post whale vanish monkey accuse crucial wreck tone

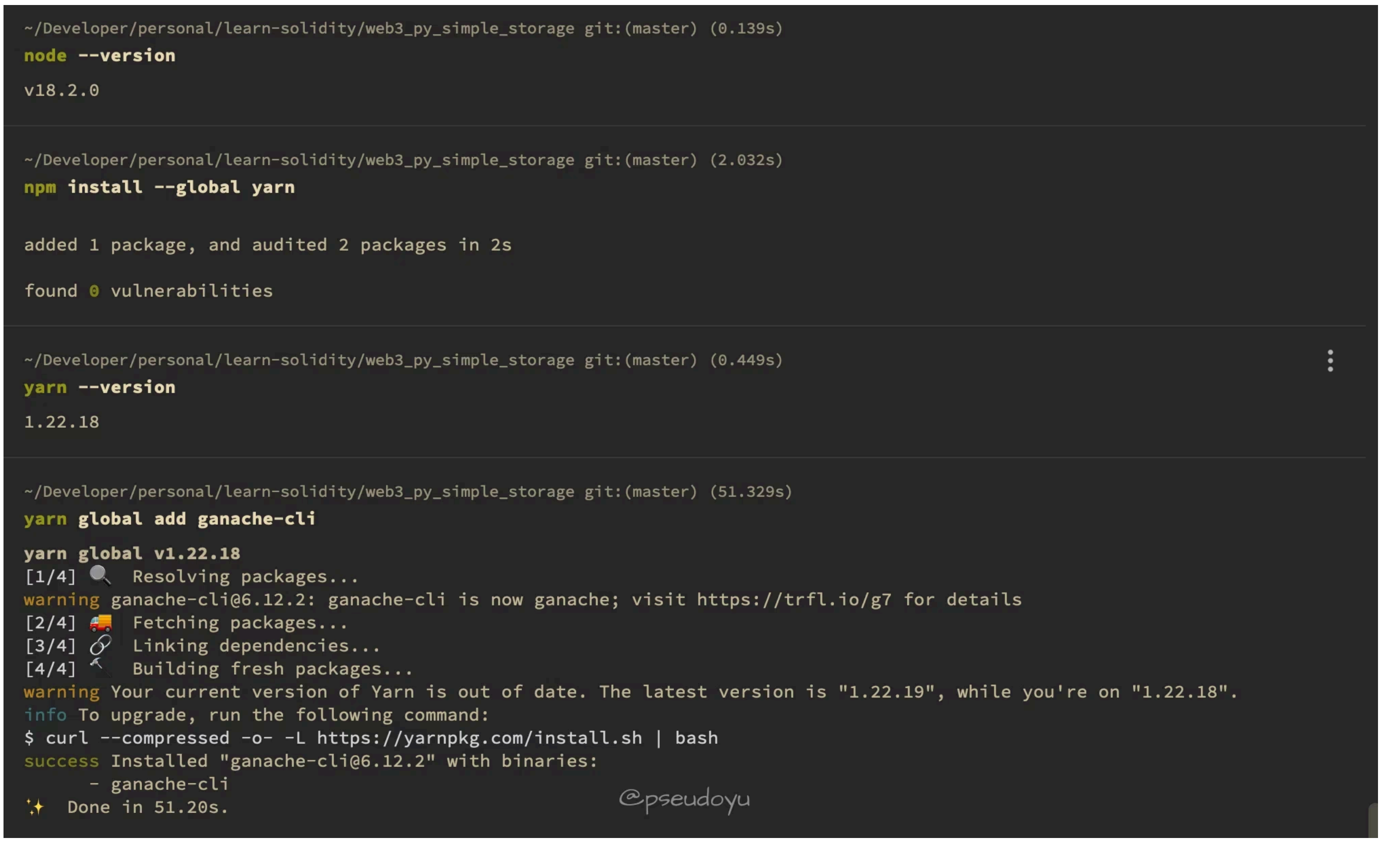
HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x2F490e1eA91DF6d3cC856e7AC391a20b1eceD6A5	100.00 ETH	0	0
0xb26238557eb2b386766a430f0E80d36031E7FB1d	100.00 ETH	0	1
0x981ad105De87127c44C41AbEA1F02119640182A4	100.00 ETH	0	2
0x6789dDFd8c59756BED0ab1e773275B119E07d66a	100.00 ETH	0	3
0x0b0ddd07F38E7d56b4CC793BCf16eb8447eD8f56	100.00 ETH	0	4
0xA320b2eEe4B63355Aa6009122d1b08EF7a7b1A4A	100.00 ETH	0	5
0x768Faa35E2EB13c7E6FA912D5cE754c778A4B645	100.00 ETH	0	6

Ganache CLI 安装

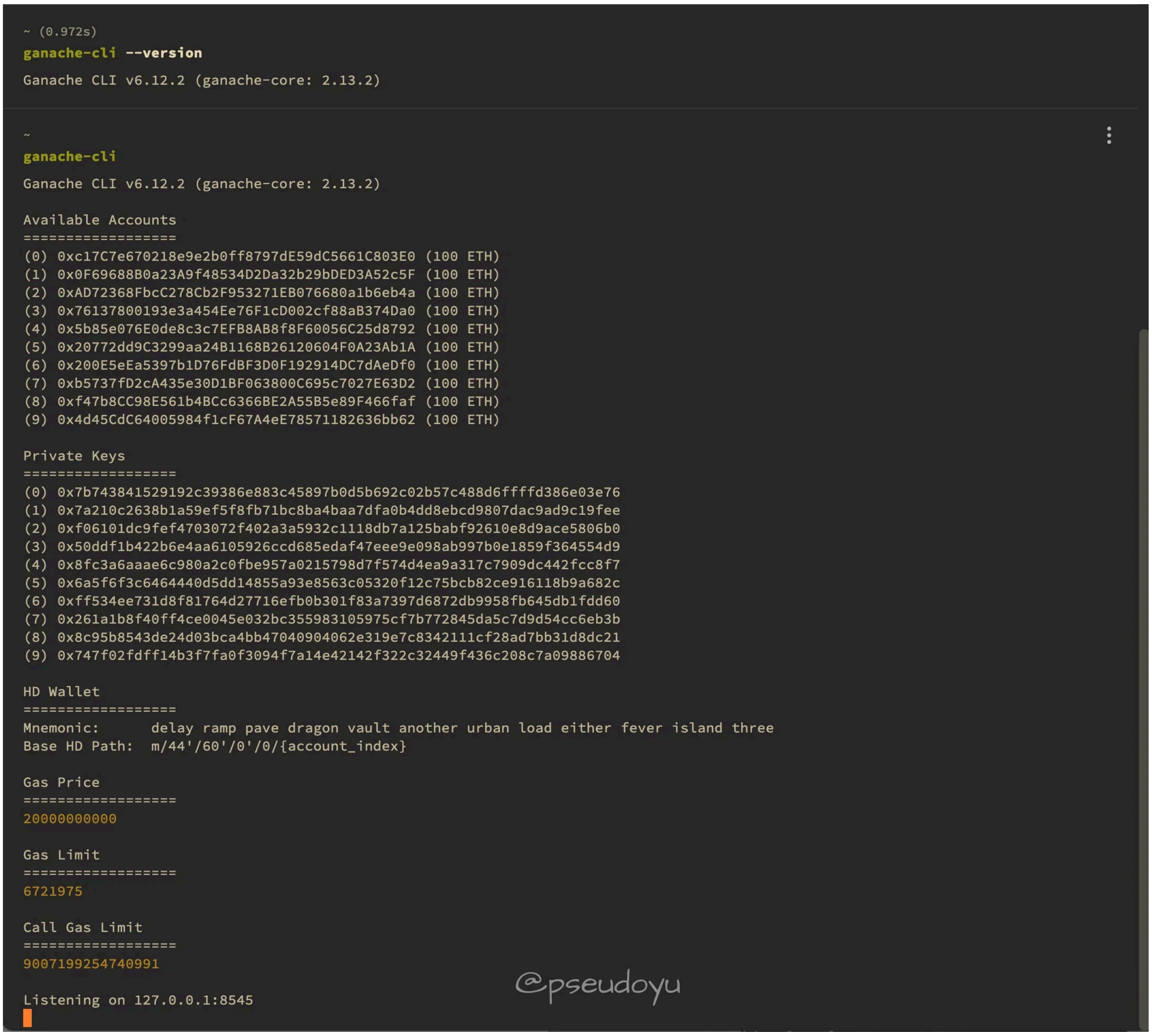
如果您的系统不支持 GUI 安装，我们可以使用 CLI 安装，安装方式如下：

```
npm install --global yarn
yarn global add ganache-cli
```



```
~/Developer/personal/learn-solidity/web3_py_simple_storage git:(master) (0.139s)
node --version
v18.2.0

~/Developer/personal/learn-solidity/web3_py_simple_storage git:(master) (2.032s)
npm install --global yarn

added 1 package, and audited 2 packages in 2s

found 0 vulnerabilities

~/Developer/personal/learn-solidity/web3_py_simple_storage git:(master) (0.449s)
yarn --version
1.22.18

~/Developer/personal/learn-solidity/web3_py_simple_storage git:(master) (51.329s)
yarn global add ganache-cli

yarn global v1.22.18
[1/4] 🔍  Resolving packages...
warning ganache-cli@6.12.2: ganache-cli is now ganache; visit https://trfl.io/g7 for details
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
[4/4] 🔨  Building fresh packages...
warning Your current version of Yarn is out of date. The latest version is "1.22.19", while you're on "1.22.18".
info To upgrade, run the following command:
$ curl --compressed -o- -L https://yarnpkg.com/install.sh | bash
success Installed "ganache-cli@6.12.2" with binaries:
      - ganache-cli
✨  Done in 51.20s.
```

等待其安装完成后即可启动本地测试网络，与 Ganache GUI 一致，也包含初始化账户与余额。



```
~ (0.972s)
ganache-cli --version
Ganache CLI v6.12.2 (ganache-core: 2.13.2)

~
ganache-cli
Ganache CLI v6.12.2 (ganache-core: 2.13.2)

Available Accounts
==================
(0) 0xc17C7e670218e9e2b0ff8797dE59dC5661C803E0 (100 ETH)
(1) 0x0F69688B0a23A9f48534D2Da32b29bDED3A52c5F (100 ETH)
(2) 0xAD72368FbcC278Cb2F953271EB076680a1b6eb4a (100 ETH)
(3) 0x76137800193e3a454Ee76F1cD002cf88aB374Da0 (100 ETH)
(4) 0x5b85e076E0de8c3c7EFB8AB8f8F60056C25d8792 (100 ETH)
(5) 0x20772dd9C3299aa24B1168B26120604F0A23Ab1A (100 ETH)
(6) 0x200E5eEa5397b1D76FdBF3D0F192914DC7dAeDf0 (100 ETH)
(7) 0xb5737fD2cA435e30D1BF063800C695c7027E63D2 (100 ETH)
(8) 0xf47b8CC98E561b4BCc6366BE2A55B5e89F466faf (100 ETH)
(9) 0x4d45CdC64005984f1cF67A4eE78571182636bb62 (100 ETH)

Private Keys
==================
(0) 0x7b743841529192c39386e883c45897b0d5b692c02b57c488d6ffffd386e03e76
(1) 0x7a210c2638b1a59ef5f8fb71bc8ba4baa7dfa0b4dd8ebcd9807dac9ad9c19fee
(2) 0xf06101dc9fef4703072f402a3a5932c1118db7a125babf92610e8d9ace5806b0
(3) 0x50ddf1b422b6e4aa6105926ccd685edaf47eee9e098ab997b0e1859f364554d9
(4) 0x8fc3a6aaae6c980a2c0fbe957a0215798d7f574d4ea9a317c7909dc442fcc8f7
(5) 0x6a5f6f3c6464440d5dd14855a93e8563c05320f12c75bcb82ce916118b9a682c
(6) 0xff534ee731d8f81764d27716efb0b301f83a7397d6872db9958fb645db1fdd60
(7) 0x261a1b8f40ff4ce0045e032bc355983105975cf7b772845da5c7d9d54cc6eb3b
(8) 0x8c95b8543de24d03bca4bb47040904062e319e7c8342111cf28ad7bb31d8dc21
(9) 0x747f02fdff14b3f7fa0f3094f7a14e42142f322c32449f436c208c7a09886704

HD Wallet
==================
Mnemonic:      delay ramp pave dragon vault another urban load either fever island three
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
```

通过 web3 连接本地 Ganache 环境

web3 提供了库可以方便地连接到本地 Ganache 环境：

```
w3 = Web3(Web3.HTTPProvider("HTTP://127.0.0.1:7545"))
chain_id = 5777
my_address = "0x2F490e1eA91DF6d3cC856e7AC391a20b1eceD6A5"
private_key = "0fa88bf96b526a955a6126ae4cca0e72c9c82144ae9af37b497eb6afbe8a97
```

Solidity 合约部署

创建合约

我们可以通过 web3 库创建合约。

```
SimpleStorage = w3.eth.contract(abi=abi, bytecode=bytecode)
```

部署合约

部署合约分为三个主要步骤：

1. 构造交易
2. 签名交易
3. 发送交易

构造交易

```
nonce = w3.eth.getTransactionCount(my_address)

transaction = SimpleStorage.constructor().buildTransaction(
    {
        "chainId": chain_id,
        "gasPrice": w3.eth.gas_price,
        "from": my_address,
        "nonce": nonce,
    }
)
```

签名交易

```
signed_txn = w3.eth.account.sign_transaction(transaction, private_key=private
```

发送交易

```
tx_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)
tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
```

与合约交互

与部署合约步骤类似，我们可以通过 web3 库与合约交互，也分为构造交易、签名交易和发送交易三个步骤。

构造交易

```
simple_storage = w3.eth.contract(address=tx_receipt.contractAddress, abi=abi)

store_transaction = simple_storage.functions.store(67).buildTransaction(
    {
        "chainId": chain_id,
        "gasPrice": w3.eth.gas_price,
        "from": my_address,
        "nonce": nonce + 1,
    }
)
```

签名交易

```
signed_store_txn = w3.eth.account.sign_transaction(
    store_transaction, private_key=private_key
)
```

发送交易

```
send_store_tx = w3.eth.send_raw_transaction(signed_store_txn.rawTransaction)
tx_receipt = w3.eth.wait_for_transaction_receipt(send_store_tx)
```

总结

以上就是我们通过 Web3.py 库与本地 Ganache 测试网络进行交互的步骤，在真正的生产项目开发中我们一般不会直接使用 Web3.py 这样的库，而是会使用 Brownie、HardHat 等进一步封装的库，但了解 Web3.py 或 Web3.js 等库的使用方法也非常重要。

参考资料

1. Solidity 智能合约开发 - 基础
2. ethereum/web3.py
3. Solidity, Blockchain, and Smart Contract - Beginner to Expert Full Course | Python Edition