Ambisafe
Software

Blockchain School

2018

Урок 3. Unit-тестирование

Запускаем локальный блокчейн, учимся использовать его для тестирования контрактов и пишем unit-тесты.

План

1. Тестирование умных контрактов с помощью Truffle
2. TestRPC (Ganache CLI)
3. Запуск тестов
4. Promise
5. Web3.js
6. Задания

1. Тестирование умных контрактов с помощью Truffle

Truffle – фреймворк на javascript для тестирования и разработки децентрализованных приложений.

На сайте Truffle вы можете посмотреть, как установить этот фреймворк, почитать документацию к нему, и ознакомиться с туториалами.

Устанавливается Truffle через npm (Node package manager). Поэтому вам нужно убедиться, что у вас стоит Node.js (желательно последней версии). Если не стоит, то установите сначала ноду, потом npm, и только потом Truffle.

Примечание: Если у вас операционная система Windows, то лучше установите виртуальную машину Ubuntu. Это поможет вам избежать головной боли. Но можете попробовать провернуть все и через винду.

2. TestRPC (Ganache CLI)

TestRPC (недавно переименованная в ganache-cli) – эфирная нода. После ее запуска у вас появится свой мини-блокчейн, в нем будет десять аккаунтов с эфиром на счету. В каждом блоке всегда одна транзакция – каждая транзакция моментально попадает в блок и майнится. Это тестовый блокчейн, который существует локально на ваших компьютерах. Когда мы его используем, мы не платим за настоящий эфир и можем использовать блокчейн для тестирования своих контрактов: проверять нормально ли передается эфир между контрактами, нет ли уязвимостей в контракте и т.д.

```
> ganache-cli --version

Ganache CLI v6.0.3 (ganache-core: 2.0.2)

Available Accounts
==================
(0) 0xd015915d49c210b5874ddb263e6d3a61aa02dfb1
(1) 0xeb7a792eacdb7b1d31bebfd6c50e1056f72e0dfb
(2) 0xef57ef160c5e85ec7afc23b08e33f6566fd0f698
(3) 0x8bf068f82b7334106a9315e61e6355b1ca184165
(4) 0xbdfe0e5bba622bfdd6cf5725ecd2791bb81a8c16
(5) 0x3e8d44ece518170e82af4a1ad0d9c04b127d5c94
(6) 0x9a8f95c99b6666b1f1cd4e46de2ec57775cc7379
(7) 0x532856a031381c0910c3363c25065e89ee996a91
(8) 0x0cc0fa688698b54ab9d147b3535649e9c3fa1a0e
(9) 0x6347fc3be1f23054cc055ac810c767198c895235

Private Keys
==================
(0) 7aa018f2323344ae1b53db9c9ea967f1b326ab76a177ea31e0c2983055646e15
(1) 57e70d9a0ba1eed838dcd7613a7dc910bde9e0da71c5d3a7e4b220858e9a5f89
(2) 4bcc516d51df8eafedeacdeaf77496b8def812182b31685878be556abb4f30ff
(3) 08d456ee8a606b7224f0993868ea027a2b746aa374e1c3ec21e77835c4ba310a
(4) d841ef1b7ad4d6b19d7d72cda48bb620a4689d23a0d5d33a85c10d1b68ce5e99
(5) c405cf5d67729d1e6c583411cfc55dc89fadc317f2a748468732b502c1637b35
(6) 547b53c5e6d26b617235a778e60759b04d839d5be5310e6225c4437d12b99f95
(7) 635ce8af2685106c040bcaac6ab026b5460e02ad449cfbcbb206d4f940330365
(8) 576b1e6b4da324dc3f3126f2bc2d679b5ebedd7f3d03c954db18db9da06fe56f
(9) 2e9f88a692a0f1d4be53ba815adad363d38a38b7464b93e728752de5fc7e5cc4

HD Wallet
==================
Mnemonic:      puppy discover august atom shy there sunset unveil release wrap soup wheel
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

Нода **TestRPC** сетапится на localhost 8545. После установки ноды все 10 аккаунтов можно использовать для тестирования контрактов, которые мы написали. При помощи тестов мы сможем убедиться, что контракты делают именно то, что по нашему плану они и должны делать.

Подробнее про фреймворк **TestRPC** можно почитать тут:

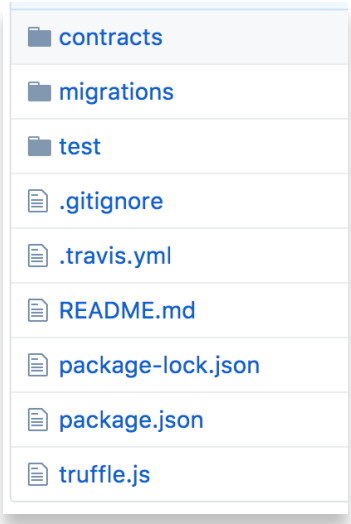
https://github.com/trufflesuite/ganache-cli

3. Запуск тестов

Для начала вам нужно сделать форк репозитория по этой ссылке и склонировать его:

https://github.com/lastperson/ethse

В репозитории вы найдете:



.gitignore – ээто то, что гит не будет подтягивать в репозиторий: временные файлы, файлы не имеющие прямого отношения к репозиторию. Чтобы не показывать кому-то файлы, над которыми работаем, их достаточно не коммитить

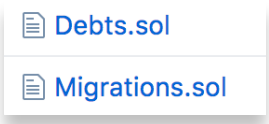
.travis.yml – конфигурационный файл для приложения TravisCI, которое будет проверять ваши тесты прямо на гитхабе

README.md – основная информация о проекте

package.json – команды, по которым можно запускать testRPC, truffle, компилить код, ссылка на гитхаб, версии фреймворков. *Тут много всего, поэтому всматривайтесь.*

truffle.js – адрес, по которому нужно искать testRPC и газ лимит каждой транзакции, а также другие настройки

Также, в репозитории есть три папки.



В папке **contracts** пока что находятся два контракта:

- **Мой контракт, описывающий логику займа/возврата денег;**
- **Контракт, описывающий логику миграции – он есть в каждом проекте truffle.**

В эту папку вам нужно будет добавить свой контракт.

В папке **migrations** есть файл **2_deploy_contracts.js.** В этот файл нужно будет добавить название своего контракта, чтобы он тоже деплоился при запуске тестов.

В папке **test** находятся тесты и хэлперы. В **debts.js** вам нужно написать тесты к моему контракту и потом создать еще один js-файл в этой папке для тестирования своего контракта.

Если вы написали тест и он подходит, то нужно проверить его еще раз. Нужно намеренно допустить ошибку и убедиться, что в этом случае тест не подходит.

4. Promise

Promise (обещание) – обертка для значения, неизвестного на момент создания обещания. Он позволяет обрабатывать результаты асинхронных операций так, как если бы они были синхронными: вместо конечного результата асинхронного метода возвращается обещание получить результат в некоторый момент в будущем. Например, когда отправленная транзакция уже смайнилась.

С помощью **.then** мы создаем целые цепочки ожиданий, тестируя таким образом наш код. Сначала мы вызываем функцию **borrow** и дожидаемся пока транзакция смайнится, потом вызываем **repay** и дожидаемся пока эта транзакция смайнится, и т.д.

Результат выполнения любого promise – это тоже promise.

Вот тут можно почитать подробнее:

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Promise

https://learn.javascript.ru/promise

Можно попробовать более читабельный вариант async/await:

https://javascript.info/async-await

https://github.com/yortus/asyncawait

5. Web3

Web3.js – это JS-библиотека, позволяющая использовать API Ethereum с помощью обычного JS. По сути, с ее помощью вы просто подключаетесь к ноде.

Вот пример ее использования (предварительно нужно запустить testRPC):

```
$ node
> var Web3 = require('web3');
> var web3 = new Web3(new Web3.providers.HttpProvider("http://local-
host:8545"));
> web3.eth.accounts
[ '0x5f7aaf2199f95e1b991cb7961c49be5df1050d86',
  '0x1c0131b72fa0f67ac9c46c5f4bd8fa483d7553c3',
  '0x10de59faaea051b7ea889011a2d8a560a75805a7',
  '0x56e71613ff0fb6a9486555325dc6bec8e6a88c78',
  '0x40155a39d232a0bdb98ee9f721340197af3170c5',
  '0x4b9f184b2527a3605ec8d62dca22edb4b240bbda',
  '0x117a6be09f6e5fbbd373f7f460c8a74a0800c92c',
  '0x111f9a2920cbf81e4236225fcbe17c8b329bacd7',
  '0x01b4bfbca90cbfad6d6d2a80ee9540645c7bd55a',
  '0x71be5d7d2a53597ef73d90fd558df23c37f3aac1' ]
>
```

Подробнее тут:

https://github.com/ethereum/web3.js/

6. Задания

Задание 1

На этой неделе работаем с репозиторием по этой ссылке:

https://github.com/lastperson/ethse

Форкайте его, клонируйте, запускайте, чтобы работало. Дописывайте тесты для контракта и делайте пулл-реквесты для ревью.

В тестах, которые я скинул, каждый тест начинается с идентичного состояния системы (в отличие от того, как обычно работают тесты в Truffle).

Задание 2

Теперь добавляйте свою версию контракта и тесты на нее.

Для этого нужно добавить деплой в файл миграции (в тот же, где деплоится Debts) и отдельный файл для тестов.

Задавайте вопросы в нашем Телеграм-чате!

Также, можете заливать готовый код в github и кидать ссылку в телеграм, чтобы все желающие могли проверить, все ли правильно, и указать на возможные проблемы.

Полезные ссылки:

Mocha JavaScript test framework

Chai Assertion Library