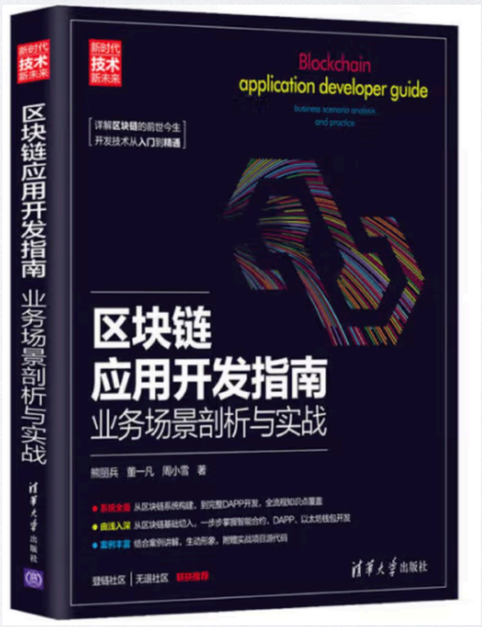
区块链集训营
二期

登链社区 - Tiny熊

ABOUT ME

- Tiny 熊
- 创新工场（点心）、猎豹移动、合伙创业
- 登链社区发起人（from 2017）
- 出版过几本区块链书籍：



课程介绍

- 视频直播、实时讨论
- 代码开源：https://github.com/xilibi2003/training_camp_2
- 图文教程：https://decert.me/tutorial/solidity/intro/
- 课后录播巩固、习题练习
- 微信群答疑
- 优秀学员推荐就业

合约开发工具

- Remix: 适用于小合约开发
- VSCode + solidity 扩展
- Truffle + Ganache
- Hardhat
- Foundry

如何开发智能合约

分别用三个方式演示 Counter 合约编译、部署、测试

- Remix
- Truffle
- Hardhat
- Foundry

Remix

- remixd : 在remixd中访问本地文件
- 安装remixd： npm install -g @remix-project/remixd
- 共享文件： remixd -s <path-to-the-shared-folder> -u <remix-ide-instance-URL>

VSCode + 插件


Solidity
v0.6.11
Nomic Foundation
94,398
★★★★★(10)
Solidity and Hardhat support by the Hardhat team
Disable
Uninstall
This extension is enabled globally.

Truffle

- Truffle：编译、部署、测试合约的一整套开发工具
  - 文档：https://trufflesuite.com/docs/truffle/
  - 中文文档：https://learnblockchain.cn/docs/truffle/
- Ganache：开发区块链，提供本地模拟的链上环境

Truffle – 创建工程

- Truffle 安装：npm install -g truffle
- 创建工程：truffle init、 truffle unbox metacoin

```
➜  home > truffle init

✔ Preparing to download
✔ Downloading
✔ Cleaning up temporary files
✔ Setting up box


Unbox successful. Sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
```

Truffle – 创建工程

- truffle工程包含：
  - contracts：智能合约目录
  - Migrations：迁移文件、用来指示如何部署智能合约
  - test：智能合约测试用例文件夹。
  - truffle-config.js：配置文件，配置truffle连接的网络及编译选项。

Truffle – 编译

truffle compile： 合约编译

truffle-config.js:

```
module.exports = {
    compilers: {
    solc: {
      version: "0.8.9"
    }
  }
}
```

```
→  > truffle compile

Compiling your contracts...
===========================
> Compiling ./contracts/Counter.sol
> Artifacts written to …/build/contracts
> Compiled successfully using:
   - solc: 0.8.9+commit.e5eed63a.Emscripten.clang
```

Truffle – 部署 – 配置网络

合约部署: 1: 配置网络，2: 编写部署脚本，3：启动网络（本地） 4. 执行部署

1. 配置网络

truffle-config.js 本地网络配置

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      gas: 5500000            //  gas limit
      gasPrice: 10000000000,  // 10 Gwei
    }
  }
};
```

Truffle – 部署 – 配置网络

真实网络

```
goerli: {
  provider: () => new HDWalletProvider(MNEMONIC, `https://goerli.infura.io/v3/${PROJECT_ID}`),
  network_id: 5,       // Goerli's id
  confirmations: 2,    // # of confirmations to wait between deployments. (default: 0)
  timeoutBlocks: 200,  // # of blocks before a deployment times out  (minimum/default: 50)
  skipDryRun: true     // Skip dry run before migrations? (default: false for public nets )
},
```

需提供助记词与 RPC URL

Truffle – 部署脚本

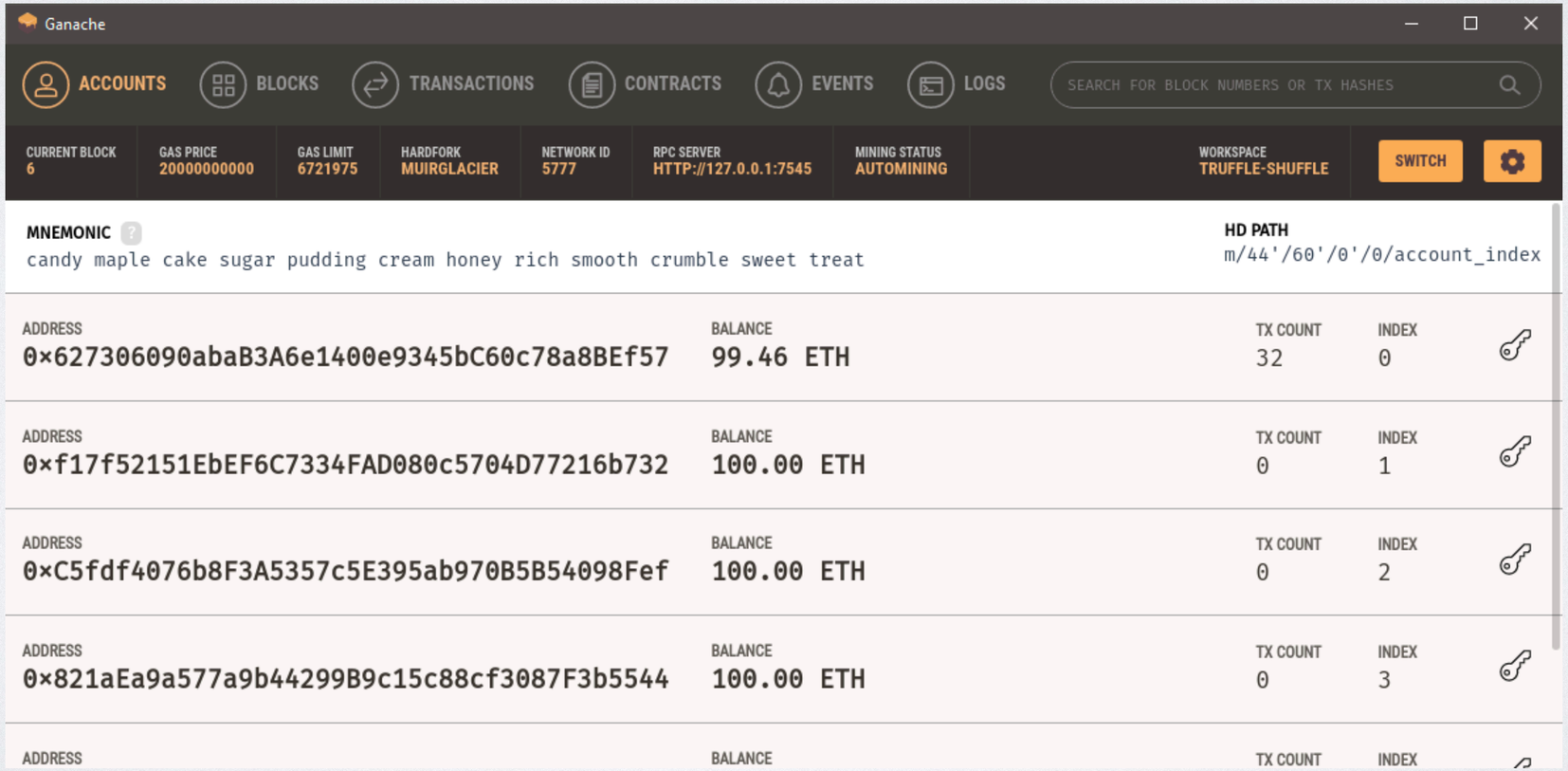
2. 编写部署脚本（migrations/1_counter.js）

Truffle 按序号（从小到大）依次执行部署脚本

```
const Counter = artifacts.require("Counter");

module.exports = function (deployer) {

  deployer.deploy(Counter);

};
```

启动网络 – Ganache

3. 启动本地网络ganache



Truffle – 执行部署

4. 执行部署：`truffle migrate [ -f 序号 –network 网络]`

```
1_counter.js
============

   Replacing 'Counter'
   -------------------
   > transaction hash:    0x188ebce772b716ed1e51a57cb8d4086466b7f57d7febd4af9fa3262d0b6eaf4f
   > Blocks: 0            Seconds: 0
   > contract address:    0x5715221024DbB1c12eA6553f409128593F8efdD8
   > block number:        2
   > block timestamp:     1643970337
   > account:             0x0CB8c19a5f73D9Bd221cB8b7CB17Ae9d191Ae71E
   > balance:             99.99468028
   > gas used:            132993 (0x20781)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00265986 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00265986 ETH
```

Truffle – 查看部署状态

完成部署后，在 Ganache 中，可以查看到相关的交易。

部署信息如合约地址也会写入之前编译生成的构建文件build/contracts/Counter.json中

Truffle – 测试

支持使用 Solidity 和 JavaScript/TypeScript 编写测试，使用:

```
$ truffle test

$ truffle test ./path/to/test/file.js
```

Truffle – 测试

JS test.js

```
var Counter = artifacts.require("Counter");

contract("Counter", function(accounts) {
  var counterInstance;
  it("Counter", function() {  // it定义一个测试用例
    return Counter.deployed()
      .then(function(instance) {
        counterInstance = instance;
        return counterInstance.count();
      }).then(function() {
        return counterInstance.counter();
      }).then(function(count) {
        assert.equal(count, 1);  // 满足断言则测试用例通过
    });
  });
});
```

Truffle – 执行脚本

$ truffle exec /path/to/script.js

JS test.js

```
var Counter = artifacts.require("Counter");

module.exports = async function(callback) {
  var counter = await Counter.deployed()

  await counter.count();
  let value = await counter.counter();

  console.log("current conter value:" + value);
}
```

补充

- truffle console 使用
  - 直接在控制台调用合约
- Truffle-min.sh
  - 压缩 artifacts 文件

Q & A

Hardhat

- Hardhat：编译、部署、测试和调试以太坊应用的开发环境，围绕 task(任务)和plugins(插件)概念设计。
- 在命令行运行Hardhat时，都是在运行任务，例如： npx hardhat compile正在运行compile任务。
- Hardhat node：开发区块链，提供本地模拟的链上环境
- 文档
  - 官方文档：https://hardhat.org/getting-started/
  - 中文文档：https://learnblockchain.cn/docs/hardhat/getting-started/

Hardhat – 创建工程

- 安装：npm install —save hardhat
- 在目录下执行 npx hardhat 创建项目
- 查看所有任务：npx hardhat

Hardhat – 工程

- Hardhat 工程包含：
  - contracts：智能合约目录
  - Scripts ： 部署脚本文件
  - test：智能合约测试用例文件夹。
  - hardhat.config.js：配置文件，配置hardhat连接的网络及编译选项。

Hardhat

合约编译: npx hardhat compile

hardhat.config.js:

```
module.exports = {
   Solidity: “0.8.18"
}
```

```
→  > npx hardhat compile
Compiling 3 files with 0.8.18
Solidity compilation finished successfully
```

Hardhat – 部署 – 配置网络

合约部署: 1: 配置网络，2: 编写部署脚本，3：启动网络（本地） 4. 执行部署

1. 配置网络

hardhat.config.js 本地网络配置

```
module.exports = {
  networks: {
    development: {
      url: “http://127.0.0.1:8545”,
      chainId: 31337
    }
  }
};
```

Harhdat – 部署脚本

2. 编写部署脚本（script/xxx_deploy.js）

```
async function main() {
  // await hre.run('compile');
  const Counter = await ethers.getContractFactory("Counter");
  const counter = await Counter.deploy();

  await counter.deployed();
  console.log("Counter deployed to:", counter.address);

}

main()
```

启动 Hardhat Node 网络

3. 启动本地网络hardhat node

```
➜  w1_code git:(main) ✗ 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
```

Hardhat – 执行部署

4. 执行部署： npx hardhat run script/xxx_deploy.js [–network 网络]

Hardhat – 测试

编写测试用例

```
describe("Counter", function () {
  async function init() {
    const [owner, otherAccount] = await ethers.getSigners();
    const Counter = await ethers.getContractFactory("Counter");
    counter = await Counter.deploy();
    await counter.deployed();
    console.log("counter:" + counter.address);
  }

  before(async function () {
    await init();
  });

  //
  it("init equal 0", async function () {
    expect(await counter.counter()).to.equal(0);
  });
```

Hardhat – 测试

```
> npx hardhat test


Greeter
Deploying a Greeter with greeting: Hello, world!
Changing greeting from 'Hello, world!' to 'Hola,
mundo!'
✓ Should return the new greeting once it's changed
(694ms)


1 passing (697ms)
```

Hardhat – 技巧

- 调试利器： console.log
- 灵活参数部署，利用 Hardhat 可以在代码中引用
  - 设置 HARDHAT_NETWORK 环境变量
  - 使用 node 部署
- 代码扁平： npx hardhat flatten xxx.sol > xxx.sol

Hardhat – 实战

- 代码验证：
  - npx hardhat verify address --network xxx
    - https://mumbai.polygonscan.com/address/0xEFD8007710Ab294B9B5C2c844890c321Aee3955E
- ABI 导出：
  - npm install --save-dev hardhat-abi-exporter
  - require('hardhat-abi-exporter') & add config
  - npx hardhat export-abi
- 自定义任务（Task）
- Harhat 项目管理（使用 network.name 动态加载配置）

Hardhat – 小结

- 合约编译: npx hardhat compile
- 启动节点： npx hardhat node/console.log
- 部署： npx hardhat run scripts/deploy.js —network xxx
- 测试： nix hardhat test

Q & A

练习题

- 修改 Counter 合约，仅有部署者 可以调用 count();
- 使用 Hardhat 部署修改后的 Counter
- 使用 Hardhat 测试 Counter:
  - Case1: 部署者成功调用 count()
  - Case 2: 其他地址调用 count() 失败
- 代码开源到区块浏览器（npx hardhat verify …） / 写上合约地址

Foundry

- Foundry：Rust 构建编译、部署、测试和调试合约开发环境。
- 特色：
  - 使用 Solidity 编写测试（而不是JavaScript）
  - 作弊代码（操控链状态）
- 文档：
  - 官方文档：https://book.getfoundry.sh/
  - 中文文档：https://learnblockchain.cn/docs/foundry/i18n/zh/index.html

Foundry

- 安装：

```
> curl -L https://foundry.paradigm.xyz | bash
> foundryup
```

- 3 个命令工具
  - Forge: 用来测试、构建和部署智能合约
  - cast：执行以太坊 RPC 调用的命令行工具, 进行智能合约调用、发送交易或检索任何类型的链数据
  - anvil： 创建一个本地测试网节点, 也可以用来分叉其他与 EVM 兼容的网络。

Foundry – 创建项目

- forge init:

  > forge init project_name

- 默认项目结构：
  - foundry.toml 项目配置文件
  - src: 合约目录
  - Test：测试目录
  - script： 部署脚本目录
  - Lib：依赖库，默认安装Forge 标准库（forge-std）

w1_foundry > foundry.toml

```toml
[profile.default]
src = 'src'
out = 'out'
libs = ['lib']

```

```
→  w1_foundry git:(main) ✗ tree . -d -L 1
.
├── lib
├── script
├── src
└── test

4 directories
```

Foundry – 编译、测试、部署

- 编译: `forge build`
- 测试：`forge test`
- 部署：`forge create/forge script script/xxx.sol --rpc-url $PRC_URL --broadcast`

Demo It

Foundry 标准库

- 提供了开始编写测试所需的所有基本功能
  - Vm.sol：作弊码接口（使用一个账号发起交易、设置账号余额、回滚区块…）
  - console.sol 和 console2.sol：Hardhat 风格的日志打印功能
  - Script.sol：声明式部署合约的实用方法
  - Test.sol：DSTest 的超集，包含标准库、作弊码实例 (vm) 和 Hardhat 控制台

Foundry – 实战

- 环境变量设置
- BaseScript.s.sol:
  - 加载助记词
  - 序列化，保存合约地址