4.2智能合约的定义、编译与测试

01 智能合约的定义

智能合约：指的是由计算机程序定义、可自动执行的承诺和协议。

以太坊智能合约：代码（即合约功能）和数据（即合约状态）的集合，存在于以太坊区块链的特定地址。 合约账户能够在彼此之间传递信息，进行图灵完备的运算。**合约依靠以太坊虚拟机(EVM) 以字节代码的二进制格式在区块链运行。、**

以太坊智能合约可以分为5种：**数据库合约、管理员合约、合约管理合约（CMC）、应用程序逻辑契约（ALC）和公用合约。**

02 智能合约的定义

数据库合约：仅用作数据存储。允许其他合约写入、更新和获取数据，以及检查调用者权限。

管理员合约：在数据库合约上运行。执行批量读/写操作。

合约管理合约（CMC）：目的是管理其他合约。主要任务是跟踪系统的所有合约/组件，处理这些组件之间的通信，并简化模块化设计。保持此功能与正常业务逻辑分离。

应用逻辑合约（ALC）：包含应用程序特定的代码。一般来说，如果合约使用控制器和其他合约来执行特定的任务，则它是ALC。

公用合约：执行特定的任务，并且可以**被其他合约无限制地调用**。它可能使用某种算法散列字符串，提供随机数字或其他东西。通常不需要太多存储空间，而且只有很少或没有依赖关系。

03 智能合约的编译

1.智能合约源码

在部署智能合约之前，需要两件事情：编译代码和应用程序的二进制接口

```
contract HelloWorld
{
    address creator;
    string greeting;

    function HelloWorld(string _greeting) public
    {
        creator = msg.sender;
        greeting = _greeting;
    }

    function greet() constant returns (string)
    {
        return greeting;
    }

    function setGreeting(string _newgreeting)
    {
        greeting = _newgreeting;
    }

    function kill()
    {
        // kills this contract and sends remaining funds back to creator
        if (msg.sender == creator)
            suicide(creator);
    }
}
```

04 智能合约的编译

2.智能合约编译工具的安装

安装智能合约编译工具solc，solc 是一个Solidiy命令行编译器，是Solidity编译方法之一。

在 Ubuntu 系统

在终端中执行以下命令：

```
sudo add-apt-repository ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install solc
which solc
```

在 Mac OSX 系统中

在终端中执行以下命令：

```
brew tap ethereum/ethereum
brew install solidity
which solc
```

在 Windosw 系统中

需要安装 chocolatey（https://chocolatey.org/），该工具可以在 windows 上管理软件。

在终端中执行以下命令：

```
cinst -pre solC-stable
```

05 智能合约的编译

2.智能合约编译工具的安装

在终端中执行以下命令：

```
git clone https://github.com/ethereum/cpp-ethereum.git
mkdir cpp-ethereum/build
cd cpp-ethereum/build
cmake -DJSONRPC=OFF -DMINER=OFF -DETHKEY=OFF -DSERPENT=OFF -DGUI=OFF
-DTESTS=OFF -DJSCONSOLE=OFF ..
make -j4
make install
which solc
```

3.智能合约的编译

在命令行输入下面的bin命令进行智能合约代码solc编译：

solc --bin HelloWorld.sol

编译以后，得到以下字节码信息：

```
60606040523415610000f57600080fd5b6040516104d0380380610
4d083398101604052808051820191905050336000806101000a8
1548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffff
ffffffffffffff160217905550806001908051906020019061008192919
0610088565b505061012d565b828054600181600116156101000
203166002900490600052602060002090601f016020900481019
```

部分字节码信息

3.智能合约的编译

除了字节码数据，还需要一个abi接口，通过执行abi命令:

solc --abi HelloWorld.sol

得到abi信息，在之后的测试部署中可能会用到：

[{"constant":false,"inputs":[],"name":"kill","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":false,"inputs":[{"name":"_newgreeting","type":"string"}],"name":"setGreeting","outputs":[],"payable":false,"stateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"greet","outputs":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":[{"name":"_greeting","type":"string"}],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]

08 智能合约的测试

三种智能合约测试的方式，分别是Remix+MetaMask、wallet +MetaMask 和truffle

```
pragma solidity ^0.4.20;
contract SimpleStorage {
        uint storedData;
        function set(uint x) {
                storedData = x;
        }
        function get() constant returns (uint) {
          return storedData;
        }
}
```

09 智能合约测试工具安装及使用

1.Remix+MetaMask

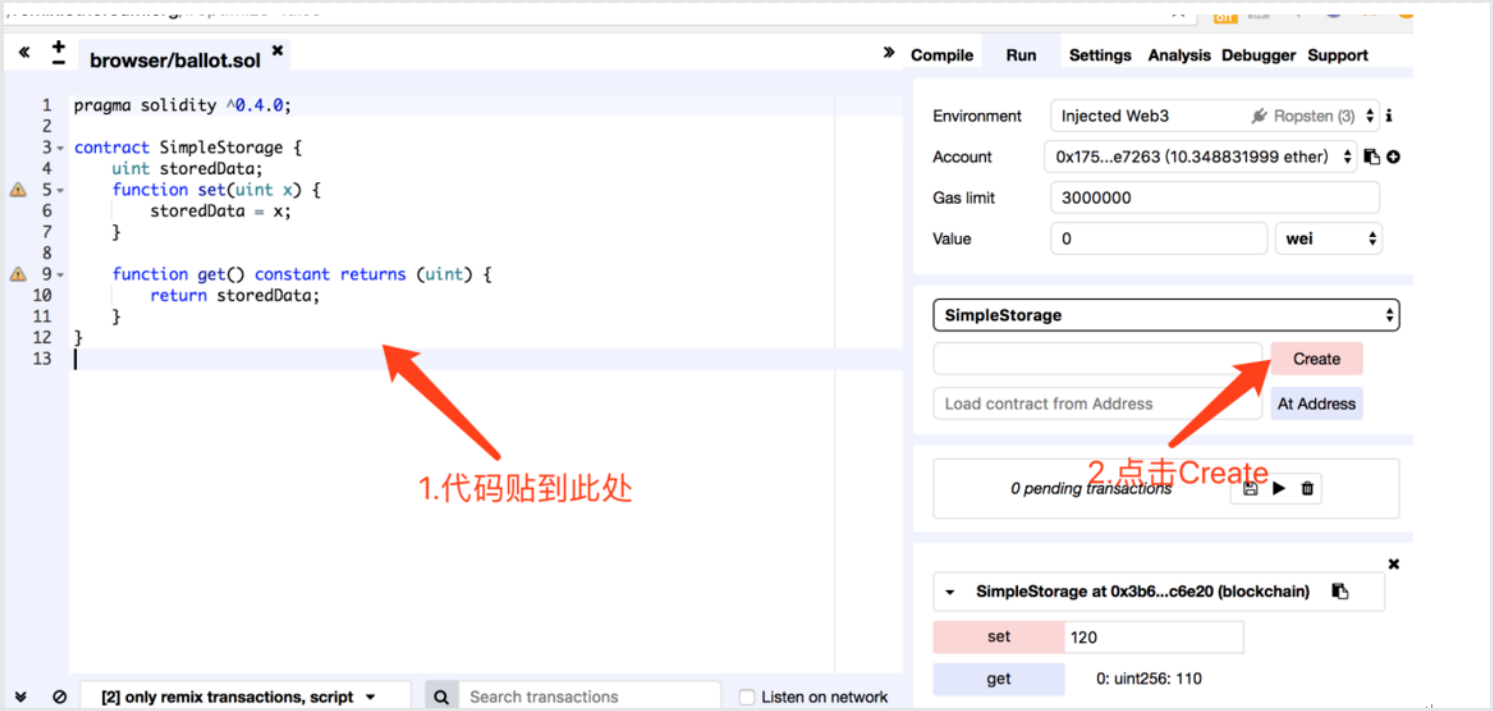
- remix 是一个浏览器版的**solidity开发 IDE（集成开发环境），**可以使用在线版的https://remix.ethereum.org/下载下来，也可以通过https://github.com/ethereum/remix-ide 安装到本地。

- MetaMask是一个浏览器插件，作用相当于**一个轻型的以太坊钱包**，谷歌或者火狐浏览器都可以使用。安装方法是：直接在浏览器应用商店中搜索MetaMask，或者在MetaMask官网https://metamask.io/根据需要直接点击下载安装。

10 智能合约测试工具安装及使用

1.Remix+MetaMask

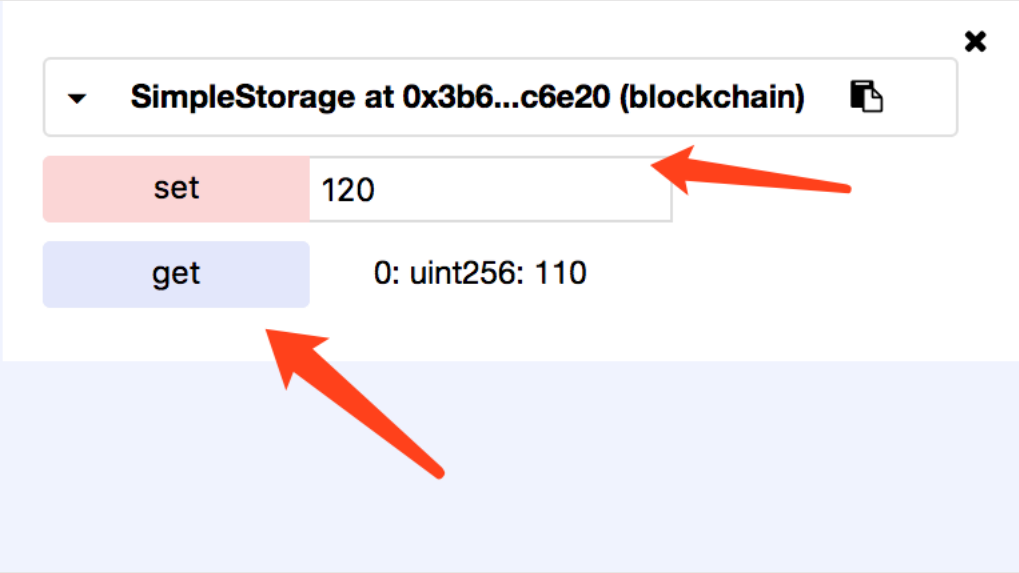
- 以太坊钱包安装完成之后，首先使用Remix发布智能合约，发布成功之后，右边就会显示已经发布成功的智能合约以及合约相关的信息，包含合约地址以及暴露到外部的接口。



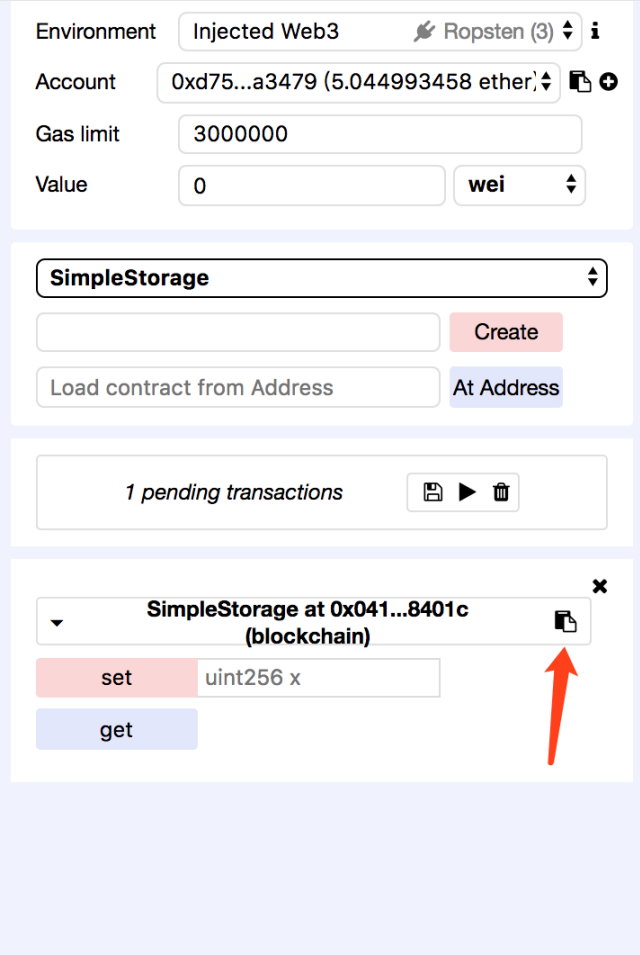
11 智能合约测试工具安装及使用

1.Remix+MetaMask

- 点击接口就能测试我们刚发布的智能合约了。



测试智能合约

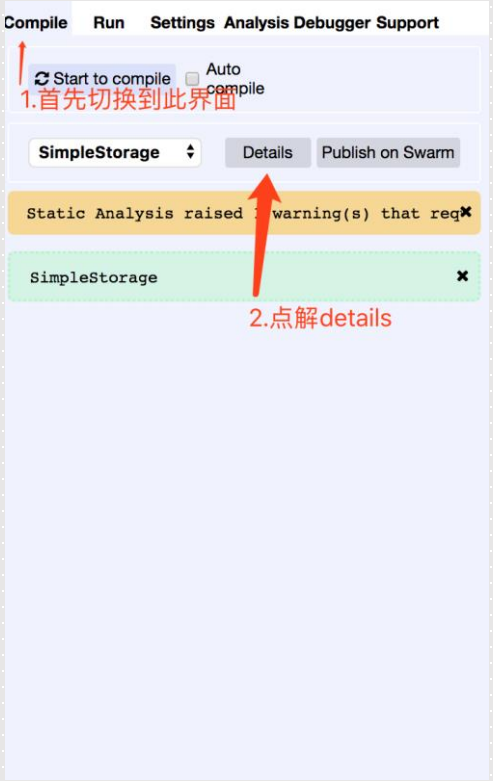


复制智能合约

12 智能合约测试工具安装及使用

1.Remix+MetaMask

- 如果想要获取abi（Application binary interface，即合约对外暴露的接口json）按照以下步骤获取：
- 首先，切换到complie界面
- 然后，点击details，显示详情，便可复制abi



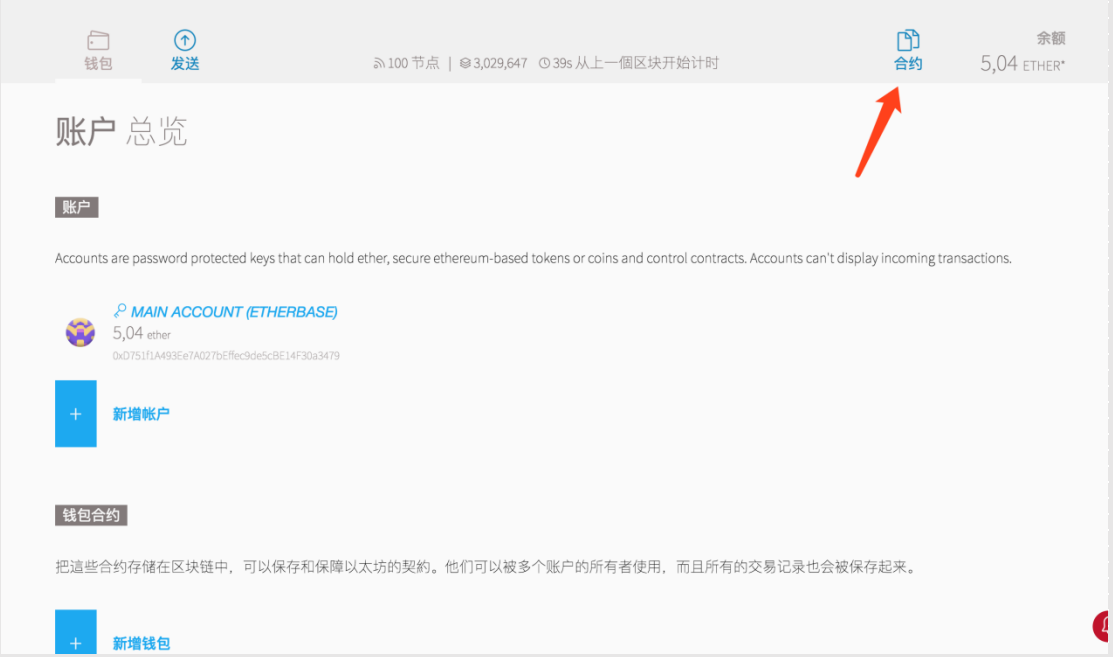
第一步

第二步

13 智能合约测试工具安装及使用

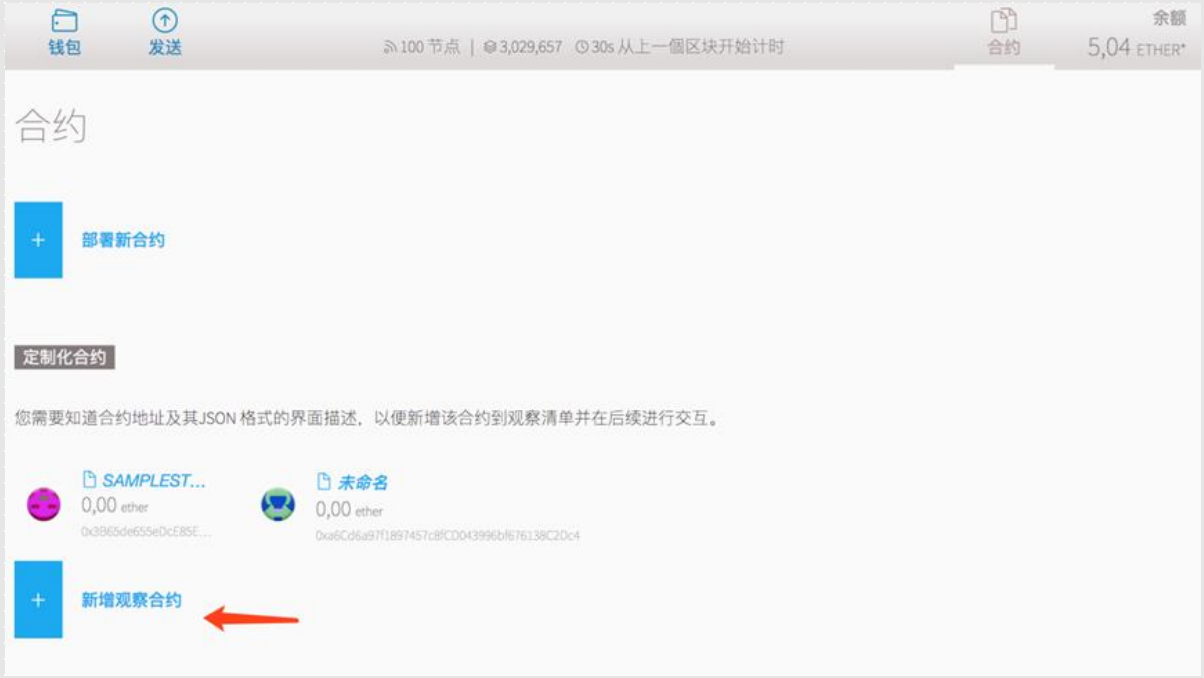
2. wallet +MetaMask

- 如果已经安装好了MetaMask，那么直接打开以太坊钱包的官网https://wallet.ethereum.org/：



14 智能合约测试工具安装及使用

2. wallet +MetaMask



点击“新增观察合约”

新增观察合约页面

15 智能合约测试工具安装及使用

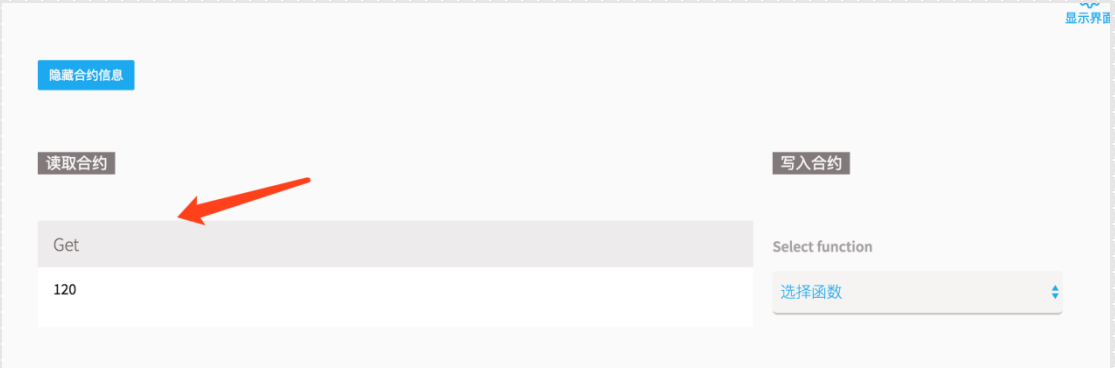
2. wallet +MetaMask

- 信息填写完成之后会跳转到如下页面：

16 智能合约测试工具安装及使用

2. wallet +MetaMask

- 点击我们刚添加的合约会跳转到如下界面。



- 然后点击右方的“选择函数”就能看到另一个接口了。

17 智能合约测试工具安装及使用

3. truffle（首先需要自行安装node环境以及cnpm）

（1）安装 truffle：

cnpm i -g truffle

（2）创建项目工程：

mkdir truffle && cd truffle

truffle init

（3）项目结构

contracts目录下存的是solidity合约代码， migrations中存的是js脚本，test中存的是测试用例。

18 智能合约测试工具安装及使用

3. truffle（首先需要自行安装node环境以及cnpm）

（4）编写代码、部署、调试

a. contracts 中新建 Hello.sol 文件，代码如右图。

注意：

类名 Hello 需要跟文件名 Hello.sol 保存一致

Migrations.sol 文件不能删除

```
pragma solidity ^0.4.20;

contract Hello {

    string weight = "18cm";
    string height = "180cm";

    function getAge() public pure returns (uint){
        return 30;
    }

    function getWeight() public view returns (string) {
        return  weight;
    }

    function getHeight() public constant returns (string) {
        return height;
    }

    function test() public returns (uint) {
        return 250;
    }
}
```

19 智能合约测试工具安装及使用

3. truffle（首先需要自行安装node环境以及cnpm）

（4）编写代码、部署、调试

b. 在 migrations 目录下添加对应的js脚本 2_depoly_hello.js

```
var myHello = artifacts.require("./Hello.sol");

module.exports = function(deployer) {
   deployer.deploy(myHello);
};
```

20 智能合约测试工具安装及使用

3. truffle（首先需要自行安装node环境以及cnpm）

代码添加完后，打开终端，切换到项目所在路径，执行代码。

```
//启动测试网络
truffle develop

//编译
compile

// 将合约部署到本地测试网络，成功后会返回合约地址
(如:0x75c35c980c0d37ef46df04d31a140b65503c0eed)
migrate

//通过合约地址，得到合约对象，赋值给变量 c
var c;
Hello.at('0x75c35c980c0d37ef46df04d31a140b65503c0eed').then((obj) => {
    c = obj;
})

//调用合约暴露的方法
c.getAge()

//如果合约中暴露出的有返回值的函数，没有用 pure/view/constant 声明，则需要调用
底层的 call 方法才能调用方法
//如上面代码中的 test 方法
c.test.call()

//修改完代码，编译后，重新部署时需重置之前的合约
migrate --reset
```