MetaMask 钱包与 Remix IDE 安装使用

以太坊Metamask安装以及remixIDE下Solidity简单智能合约编写部署

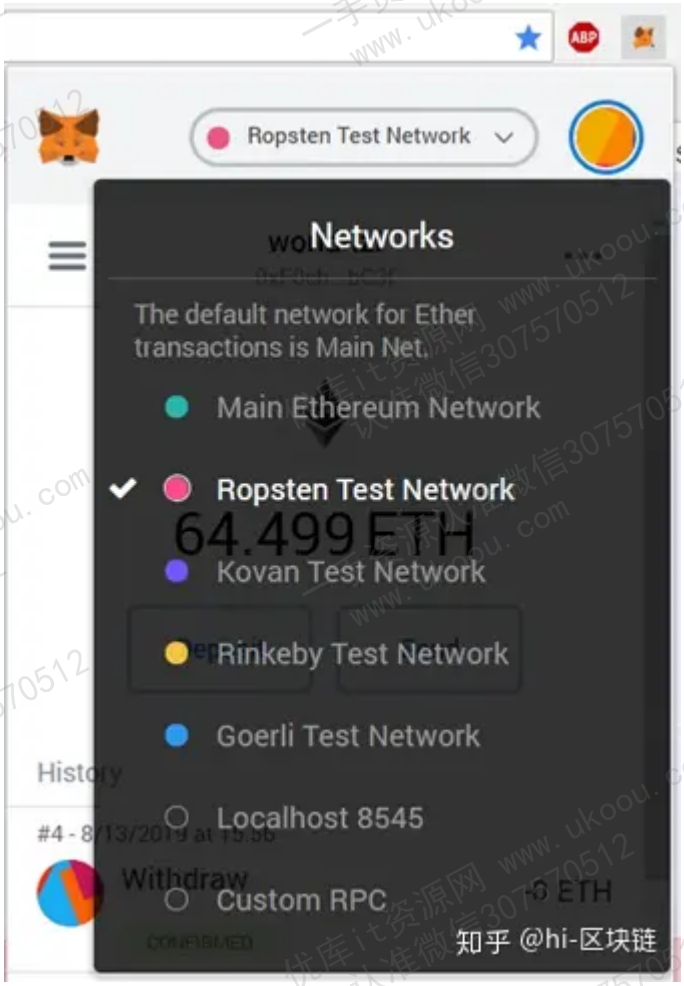
本文使用MetaMask创建钱包，并且使用Ropsten测试网络上的一个faucet为它充值。利用Solidity编写简单的faucet合约并使用Remix IDE将合约编译为EVM字节码，进行简单测试学习。

1． 谷歌浏览器下安装MetaMask插件

https://chrome.google.com/webstore/category/extensions

安装输入密码注册账号，并保存相应助记词

切换至ropsten测试网络



注：

Main Ethereum Network

以太坊区块链主网。真正的ETH，真正的价值。

Ropsten Test Network

以太坊公开测试区块链和网络，使用工作证明共识（挖矿）。

Kovan Test Network

以太坊公开测试区块链和网络，使用 Aura 共识协议（Proof-of-Authority）该测试网络仅由“Parity”支持。Other Ethereum clients use the Clique consensus protocol, which was proposed later, for proof of authority–based verification.

Rinkeby Test Network

以太坊公开测试区块链和网络，using the Clique consensus protocol with proof of authority (federated signing)

Localhost 8545

连接到与浏览器在同一台计算机上运行的节点。The node can be part of any public blockchain (main or testnet), or a private testnet.

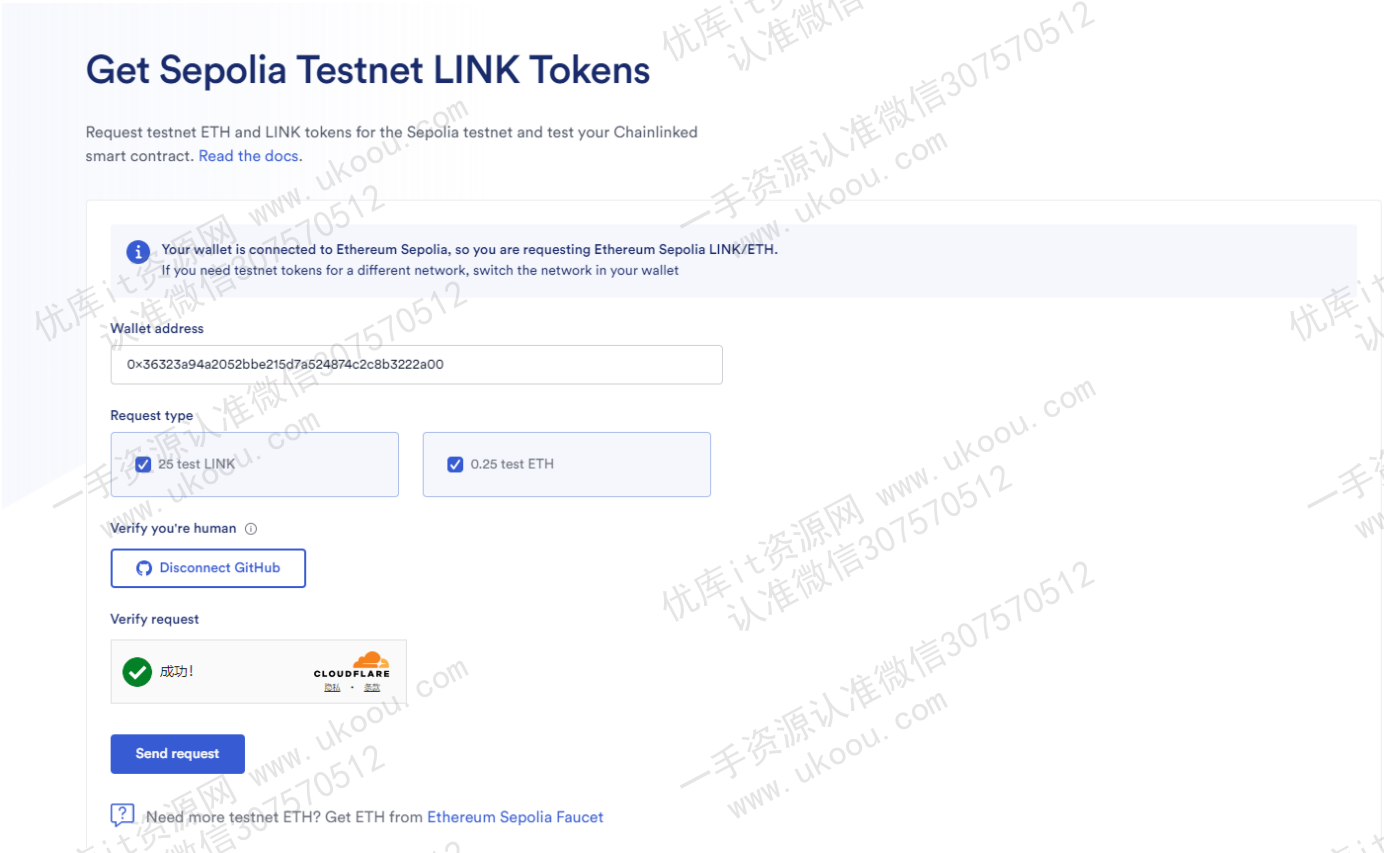
Custom RPC

允许你将MetaMask连接到任何具有geth兼容的远程过程调用（RPC）接口的节点。The node can be part of any public or private blockchain.

2.获得测试网上的以太币

点击绿色request即可

https://faucets.chain.link/



3.在线智能合约IDE

https://remix.ethereum.org

打开并选择老版本界面-Remix IDE选择使用0.4.19版本的Solidity编译器

勾选自动编译

点击左上角创建Faucet.sol智能合约并键入以下的代码

4.编写智能合约

Faucet类似刚才我们获得免费的以太币网站功能一样：它给任何地址发放ether，可以定期补充。

```
// Our first contract is a faucet!
contract Faucet {
// Give out ether to anyone who asks
function withdraw(uint withdraw_amount) public {
// Limit withdrawal amount
require(withdraw_amount <= 100000000000000000);
// Send the amount to the address that requested it
msg.sender.transfer(withdraw_amount);
}
// Accept any incoming amount
function () public payable {}
}
```

```
contract Faucet {}
```

该行声明了一个合约对象，类似于JavaScript，Java或C++等其他面向对象语言中的 class 声明。合约的定义包含了大括号中的所有行 {}，它定义了范围 ，就像在其他许多编程语言中使用花括号一样。

接下来，我们声明faucet合约的第一个函数：

```
function withdraw(uint withdraw_amount) public {}
```

函数名为 withdraw，它接收一个无符号整数（uint）名为 withdraw_amount 的参数。它被声明为 public 函数，意味着它可以被其他合约调用。函数定义在花括号之间：

```
require(withdraw_amount <= 100000000000000000);
```

withdraw方法的第一部分设置了取款限制。它使用内置的Solidity函数 require 来测试前提条件，即 withdraw_amount 小于或等于100000000000000000 wei，它是ether的基本单位，等于0.1 ether。

如果使用 withdraw_amount 大于该数量调用 withdraw 函数，则此处的 require 函数将导致合约执行停止并失败，并显示异常。

合约的这部分是我们faucet的主要逻辑。它通过设定提款限额来控制合约的资金流出。

```
1 msg.sender.transfer(withdraw_amount);
```

msg 对象是所有合约可以访问的输入之一。它代表触发执行此合约的交易。属性 sender 是交易的发件人地址。函数 transfer 是一个内置函数，它将ether从合约传递到调用它的地址。

从后往前读，表示 transfer 到触发此合约执行的 msg 的 sender。transfer 函数将一个金额作为唯一的参数。我们传递之前声明为 withdraw 方法的参数的 withdraw_amount 值。

紧接着的一行是结束大括号，表示 withdraw 函数定义的结束。

下面我们又声明了一个函数：

```
1 function () public payable {}
```

此函数是所谓的_“fallback”_或_default_函数，如果合约的交易没有命名合约中任何已声明的功能或任何功能，或者不包含数据，则触发此函数。合约可以有一个这样的默认功能（没有名字），它通常作为接收ether使用。它被定义为 public 和 payable 函数意味着它可以接受合约中的ether。除了大括号中的空白定义 {} 所指示的以外，它不会执行任何操作。

在我们的默认函数下面是最后一个关闭花括号，它关闭了合约 faucet 的定义。

5.在以太坊区块链创建合约

在把之前的合约进行编译后，它已编译成字节码。此时在区块链上注册合约涉及创建一个特殊交易，其目标是地址0x0000000000000000000000000000000000000000，称为zero address。零地址是一个特殊的地址，告诉以太坊区块链将进行合约注册。

首先，切换到“Run”选项卡，并在“Environment”下拉列表框中选择“Injected Web3”。这将Remix IDE连接到MetaMask钱包，并通过MetaMask连接到Ropsten测试网络。

此时可以在Evironment下看到“Ropsten”。在Account选择框中，显示钱包地址。在刚刚确认的“Run”设置下方，是Faucet合约，点击deploy进行创建。

从MetaMask中可以看到，合约创建交易没有ether，但它有258个字节（编译的合约），并且会消耗相应的Gwei。点击“confirm”来批准：

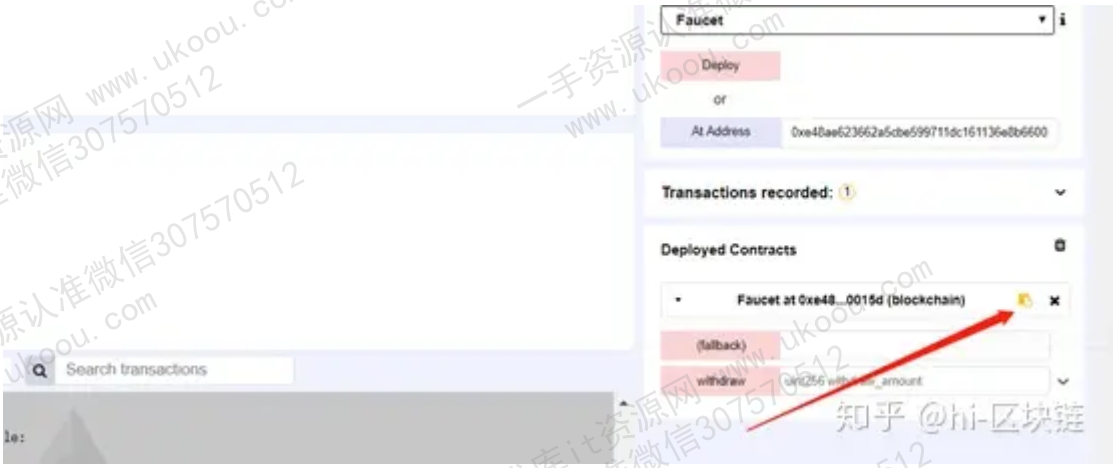
或者输入对应创建好的合约地址并点击at address

等待合约创建需要大约15到30秒的时间，合约创建后，它会显示在“运行”选项卡的底部：

6.与合约交互

此时区块链合约中有一个以太坊地址，通过点击名称旁边的剪贴板图标来复制合约的地址。

0xe48ae623662a5cbe599711dc161136e8b660015d



合约创建交易

https://ropsten.etherscan.io/tx/0x1a30ecd07d8c11854400e9e4185d236088c70d71e07e5c7d20b52dfc9479a76d

合约地址

https://ropsten.etherscan.io/address/0xe48ae623662a5cbe599711dc161136e8b660015d

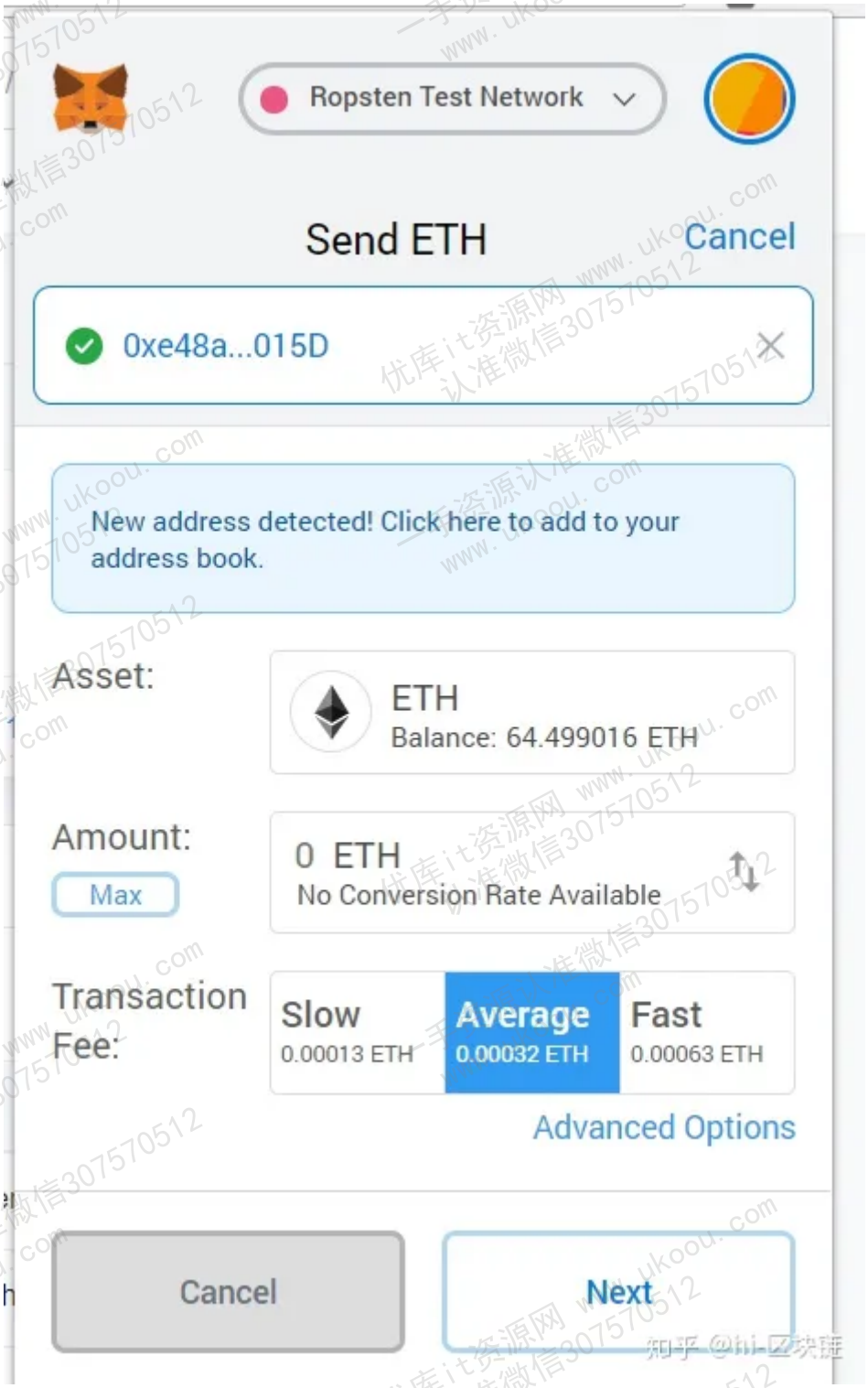
最下面这条是合约创建时候的交易

此时value只有0 ether我们需要进行如下步骤进行合约资金的增加

7.给合约增加资金余额

点击send发送

输入合约地址并发送



Transaction Fee选择交易手续费

选择太低则可能发生如下图所示的错误

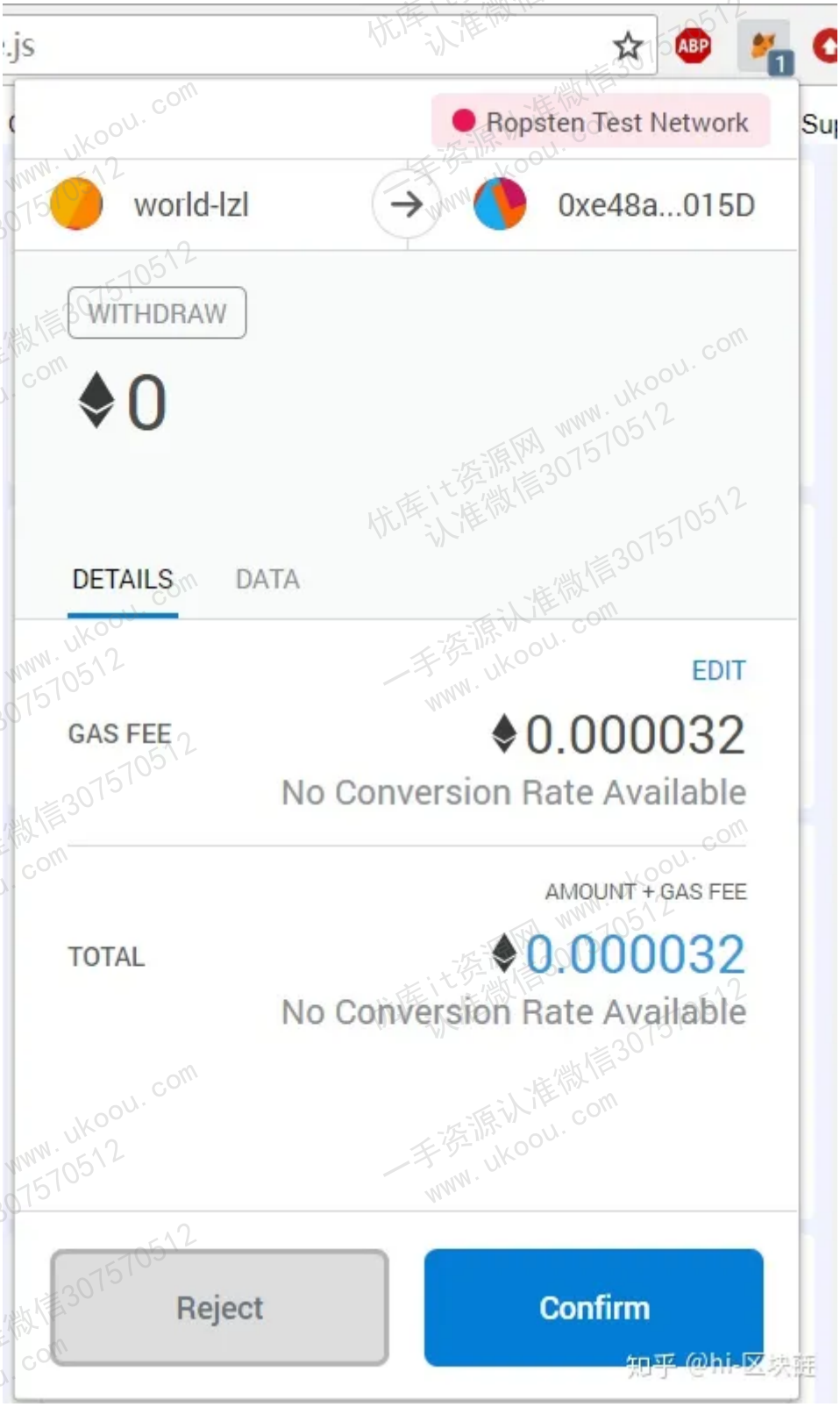
当你将交易发送到合约地址时，没有指定要调用哪个函数的数据，它将调用默认函数function () public payable {}。接受1 ether并存入合约账户余额中。交易导致合约在EVM中运行，更新其余额。

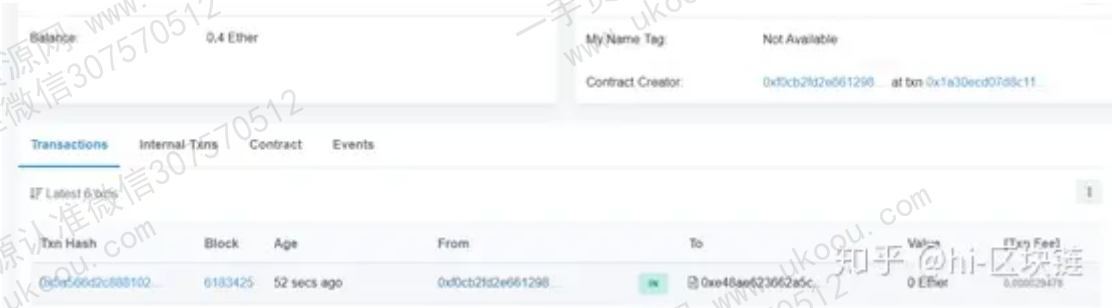
8.从合约中取出余额

返回到Remix选项卡并在“Run”选项卡下查看合约。你应该看到一个标记为 withdraw 的红色框，其中带有一个标记为 uint256 withdraw_amount：

以太坊中的所有货币值都以 wei 计价， withdraw 函数预期 withdraw_amount 也以 wei 计价。假设需要的数量是0.1 ether，这是 100000000000000000 wei（1后面跟着17个零）。输入，然后单击 withdraw 按钮：

查看右上角小狐狸图标，并点击Confirm确认

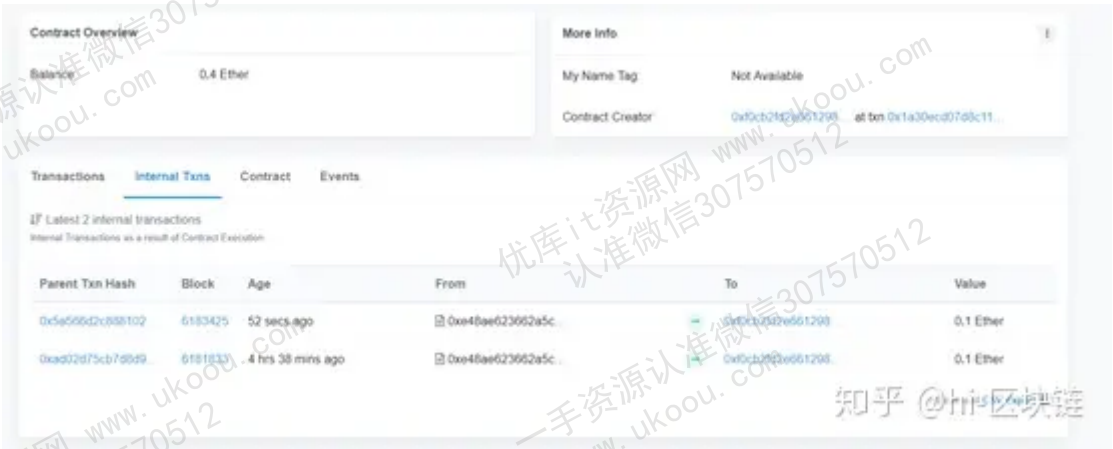




提币到自己的地址成功

我们现在看到一个新的交易，其中合约地址是目标地址，0 ether。合约余额已经改变，现在是0.9 ether，因为它按要求给了我们0.1 ether。但是我们在合约地址历史记录中看不到“OUT”交易。

提款的交易在哪里？合约的地址历史记录页面中出现了一个名为“内部交易”的新选项卡。由于0.1 ether传输源于合约代码，因此它是一个内部交易（也称为_message_）。点击“内部交易”标签查看：



这个“内部交易”是由合约在这行代码中发送的（Faucet.sol的 withdraw 方法）

```
1  msg.sender.transfer(withdraw_amount);
```

总结一下：我们从MetaMask钱包发送了一个包含数据指令的交易，以 0.1 ether 的withdraw_amount参数调用 withdraw 函数。该交易导致合约在EVM内部运行。当EVM运行faucet合约的 withdraw 功能时，首先它调用require函数并验证我们的金额小于或等于最大允许提款0.1 ether。然后它调用 transfer 函数向我们发送ether。运行 transfer 函数生成一个内部交易，从合约的余额中将0.1以太币存入我们的钱包地址。这就是 etherscan 中“内部交易”标签中显示的内容。