**实验7 以太坊宠物商店**

**【实验介绍】**

区块链技术的发展和普及，带动了一系列新的基于区块链的产品的诞生，其中一种名为去中心化应用(Decentralization Application)的产品较为火热。以太坊网络曾经遭遇过一次大规模的网络堵塞，而堵塞的原因是因为一款名为CryptoKitties的虚拟养猫DApp，这款游戏中的每只小猫都是独一无二的，用户可以通过买卖或者繁殖的手段来获取不同品种的虚拟小猫，曾有数据表明在当时最新的1000多个区块中，该去中心化应用占据了总交易数量六分之一的交易比例。仅仅一周不到的时间，该应用涉及了将近等价于1000万人民币的以太币交易，这体现了DApp的巨大潜力，同时也给当时的以太坊网络敲响了加速交易处理和性能优化的警钟。

本实验通过模仿开发一个简易的以太坊宠物商店心化应用（DAPP），让大家领养宠物，了解其运行模式。使用truffle box已提供pet-shop网站应用的框架代码，我们仅需编写智能合约及其交互部分。



官网地址：<https://www.trufflesuite.com/tutorial>

(1) 分布式应用DApp

Dapp，其英文全称为Decentralization Application，译为去中心化应用，光从字面意义可能难以对其下定义，网上最多的定义是将其分解成几个已有的概念，前端+智能合约+token(类似股票证券，表明所有者权益)。前端即继承了原有APP的概念，就是与用户交互的应用界面；智能合约即以太坊网络中去中心化的核心，其运行于区块链网络中，不依赖某个中心服务器，通过区块链本身作为中介，参与者必须遵守合约的规定，同时也是DAPP的业务逻辑支持；而token就相当于证明，持有token的人就是Dapp参与者，其规模可以远远大于那种依靠单个中心服务器的APP，因为支撑DAPP的是整个区块链网络。

不过我们这个实验目的并不是想让大家投入DAPP的潮流当中，我们仅限于对其技术层面的讨论，正如前几个智能合约实验一样。通过对DAPP的说明来展示它是如何将APP和区块链做融合和改变的。如下图7-1，我们给出了一个对于数据处理和存储层面的APP与DAPP的简单的区分。在APP中，对于数据的操作都是通过一些数据库语言实现的，例如图中给出的SQL，而存储自然就是其相应的数据库来存储信息。而对于DAPP而言，数据的操作就是通过solidity和智能合约来实现的，数据也是分布式地存储在以太坊的EVM中的，这也是为什么有人称区块链是分布式的数据库管理系统的原因。这两者的区别其实我们在实验6学生管理系统中也应该已经体会到了。

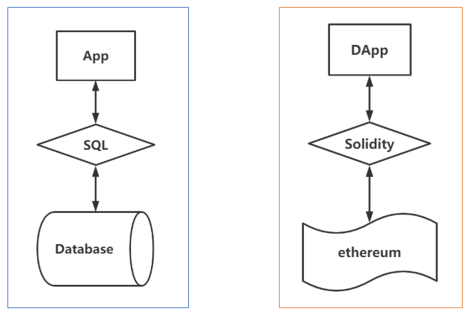


图7-1 DAPP和APP的区别

(2) Truffle框架：

Truffle是一种以太坊solidity编程语言开发框架，通过封装简化了以太坊上的开发，可以直接运行脚本文件，通过调用接口来方便地完成项目初始化以及智能合约的编译，部署与测试等功能。同时有较好的兼容性，可以支持执行外部的脚本，如web3.js等。本实验中我们会使用的宠物商店Dapp就是在truffle框架下的实现的，我们仅需修改其中的内容即可。

(3) MetaMask插件：

MetaMask是一个浏览器扩展插件，相当于一个以太坊钱包，可以通过该插件与以太仿网络进行交互，本实验通过MetaMask连接到本地测试的区块链网络，并通过私钥登入网络中的一个账户，使用太坊宠物商店Dapp。

**【实验要求】**

（1）创建Truffle项目

（2）编写智能合约

（3）编译和部署智能合约到区块链

（4）学习Ganache和MetaMask的使用

**【实验准备】**

**环境配置**

（1）创建项目目录并进入

|  |
| --- |
| **mkdir** pet-shop  **cd** pet-shop |

（2）初始化项目

使用truffle unbox拉取github仓库的box框架代码，等待下载构建完成。truffle unbox pet-shop 若网络不好无法下载项目，建议可选一个国内网络环境的项目下载地址，如果安装不下来，可以用已经提供的资源包。1. truffle unbox pet-shop 若网络不好无法下载项目，建议可选一个国内网络环境的项目下载地址）

|  |
| --- |
| truffle unbox pet-shop |

下载成功的话会有下图7-2的提示:

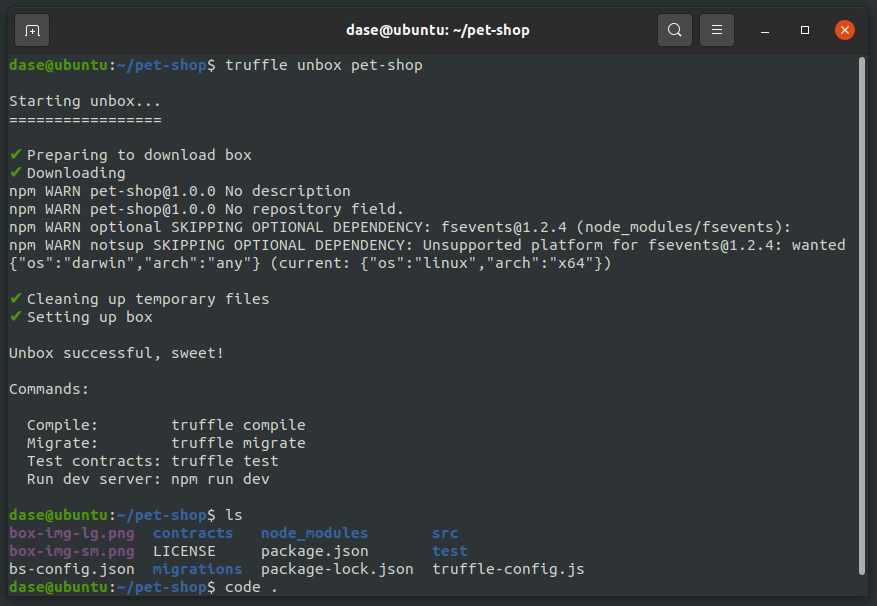


图7-2 pet-shop代码拉取成功

contracts/智能合约的文件夹，所有的智能合约文件都放置在这里，里面包含一个重要的合约Migrations.sol。migrations/用来处理部署（迁移）智能合约，迁移是一个额外特别的合约用来保存合约的变化。test/智能合约测试用例文件夹。truffle.js/配置文件。

**【实验过程】**

**1. 编写智能合约**

在contracts/目录下添加智能合约文件Adoption.sol，代码如图7-3所示：

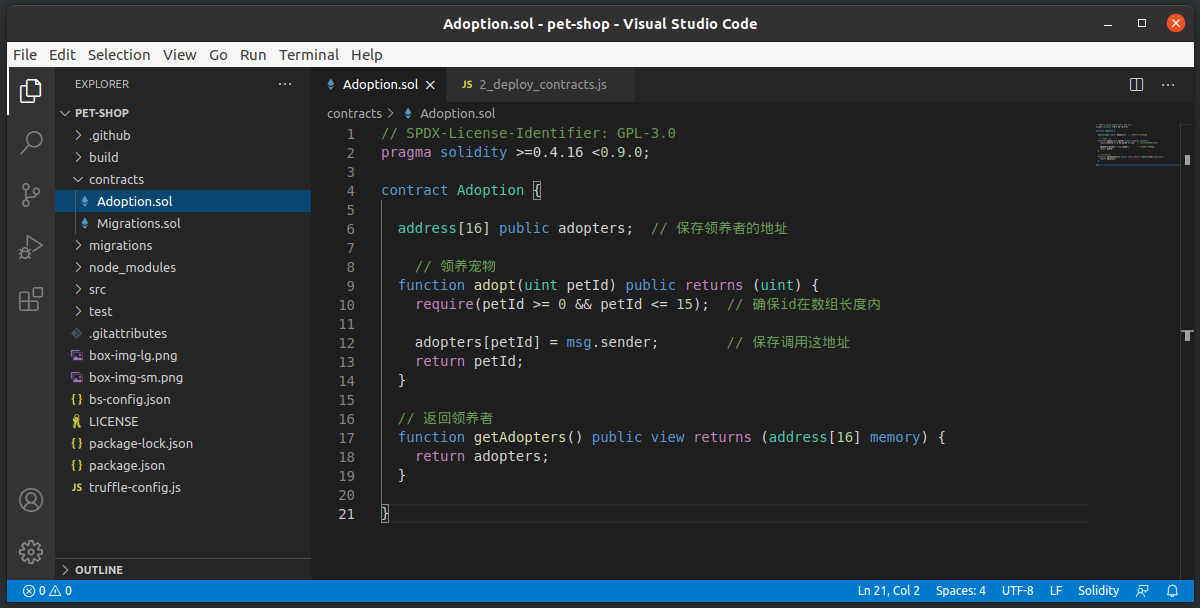


图7-3 智能合约-Adoption

**2. 编译部署智能合约**

Truffle集成了一个开发者控制台，可用来生成一个开发链用来测试和部署智能合约。

(1) 编译

在项目目录下输入truffle compile命令进行编译，显示下图7-4结果表明编译成功：

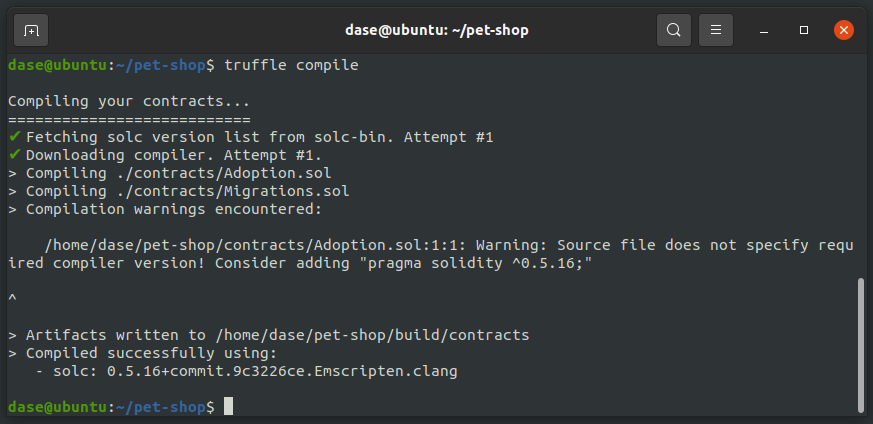


图7-4 智能合约编译成功

(2) 编写部署脚本

编译后就可以将其部署到区块链上。在migrations/文件夹下已经有一个1\_initial\_migration.js部署脚本，用来部署Migrations.sol合约。Migrations.sol用来确保不会部署相同的合约。

创建一个自己的部署脚本2\_deploy\_contracts.js，代码如图7-5所示：

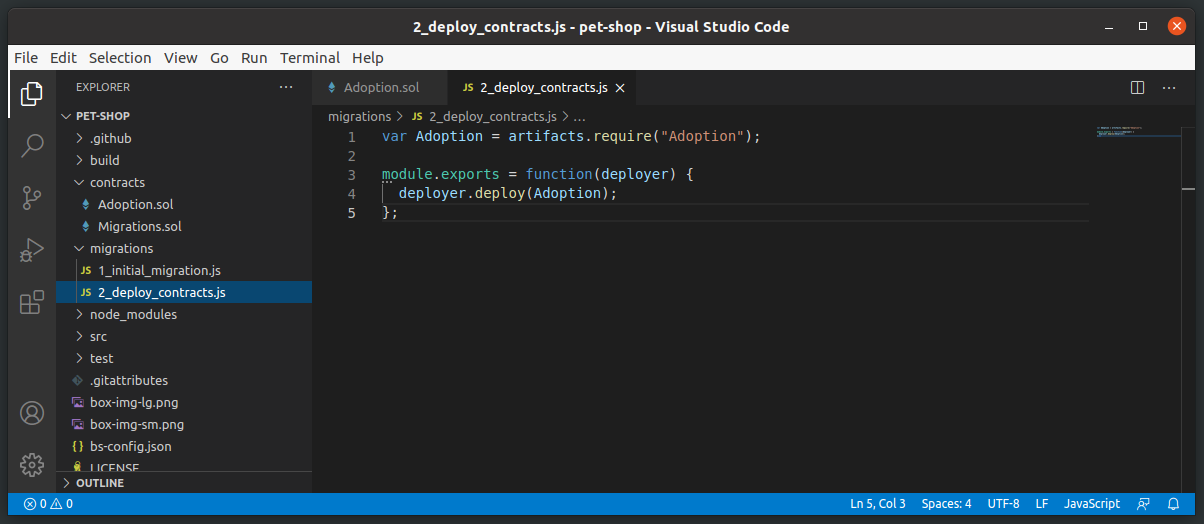


图7-5 Adoption合约部署脚本

(3) 开启区块链

部署需要区块链，使用[Ganache](http://truffleframework.com/ganache/)可快速开启一个私链来进行开发测试，默认会在7545端口上运行一个开发链。

新开一个终端，在dase/目录下输入./ganache-2.5.4-linux-x86\_64.AppImage运行Ganache，点击QUICKSTART快速启动一个以太坊区块链，如图7-6所示。

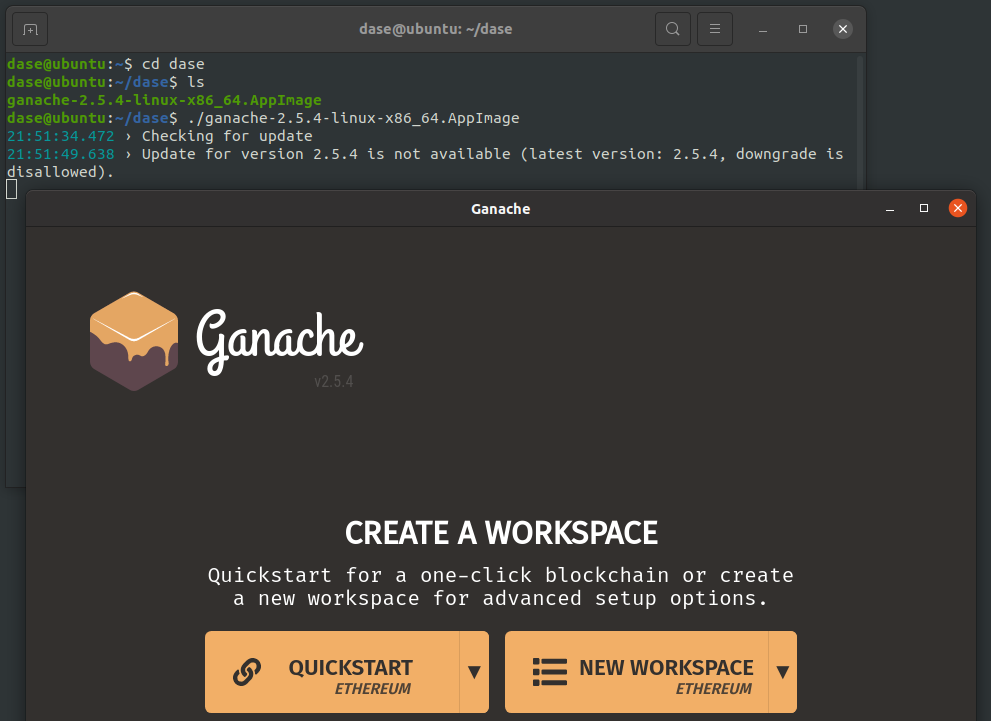


图7-6 启动一个测试链

(4) 部署

执行部署命令truffle migrate，编译情况如下图7-7，可以看到部署合约消耗的以太币。

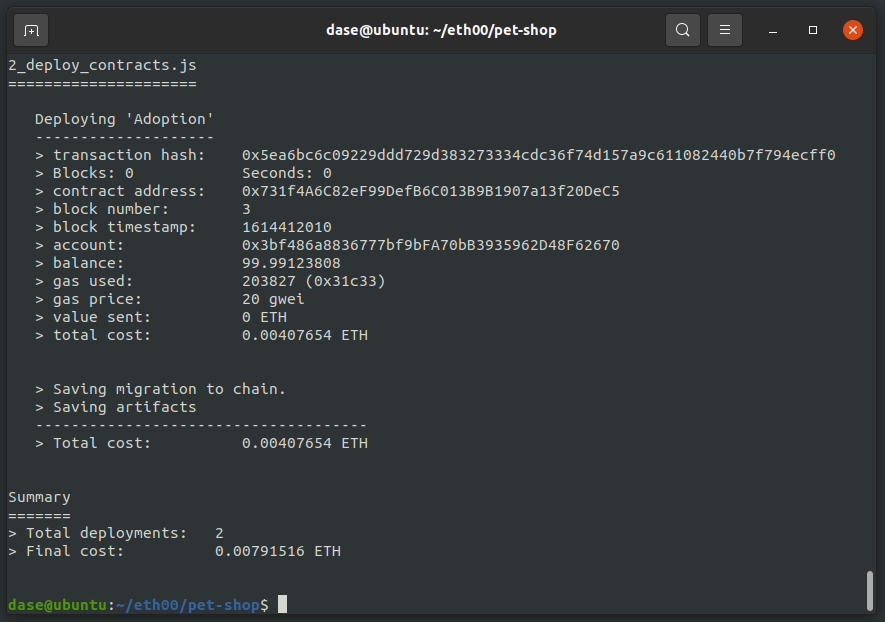


图7-7 部署合约

Ganache账户页面可见消耗的以太币，如图7-8所示，区块页面可见产生了四个区块，此时智能合约已部署完毕。

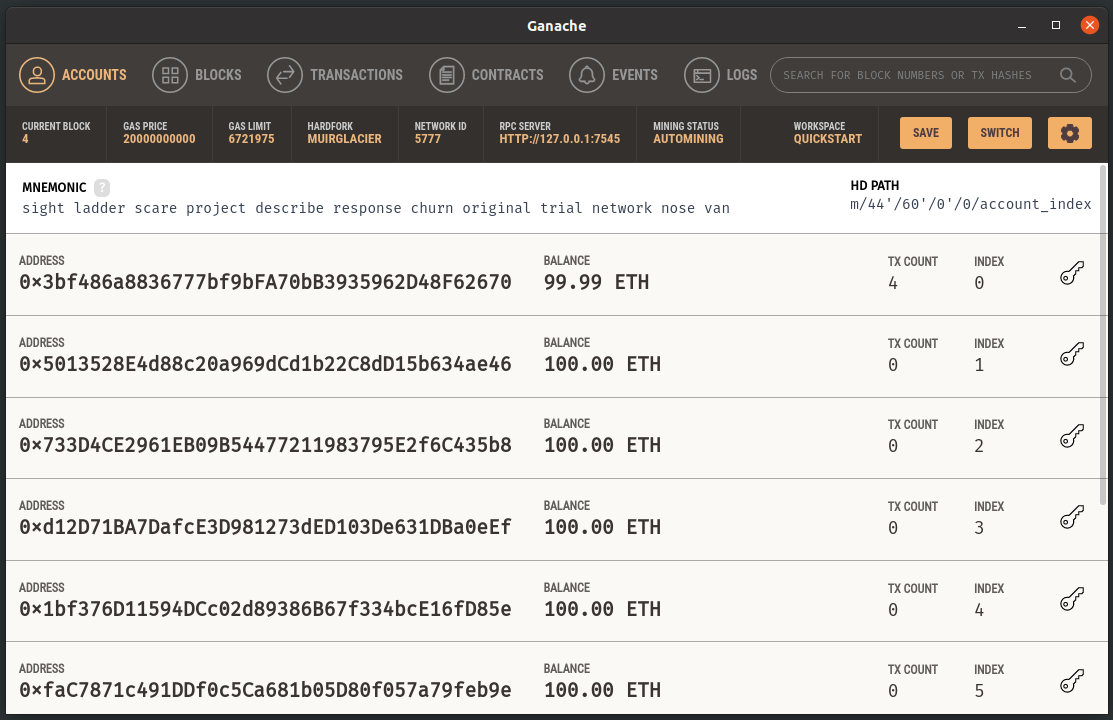


图7-8 合约部署后产生的区块和消耗的以太币

**3. 测试**

(1) 在test/目录下新建一个TestAdoption.sol，编写测试合约，其中Assert.sol及DeployedAddresses.sol由truffle框架提供，代码如下图7-9所示：

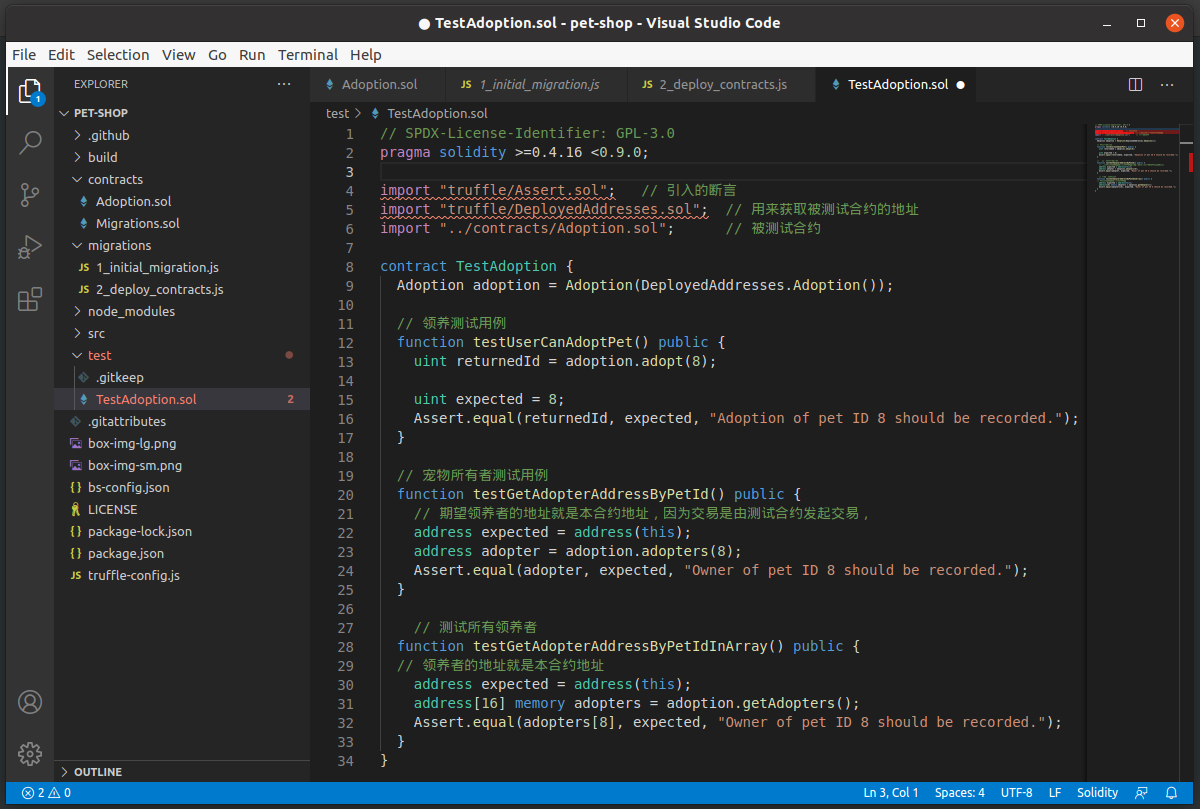


图7-9 智能合约-TestAdoption

在终端执行truffle test进行测试：

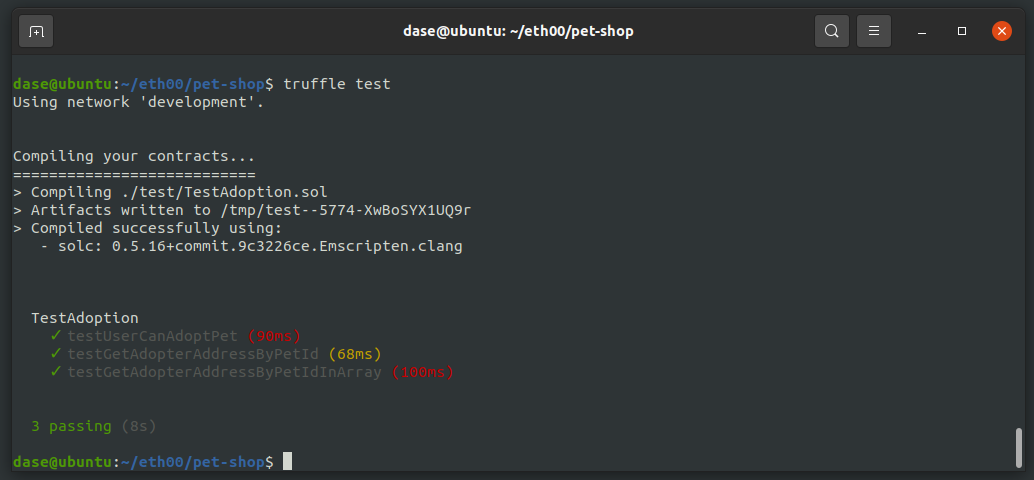


图7-10 合约TestAdoption测试结果

上图7-10表示测试通过，可通过Gnanache观察区块链的变化。

**4. 创建用户接口和智能合约交互**

Truffle Box pet-shop里，已经包含了应用的前端代码，代码在src/文件夹下。

(1) 初始化web3

修改app.js中的initWeb3: function()，添加下图7-11蓝框内容：

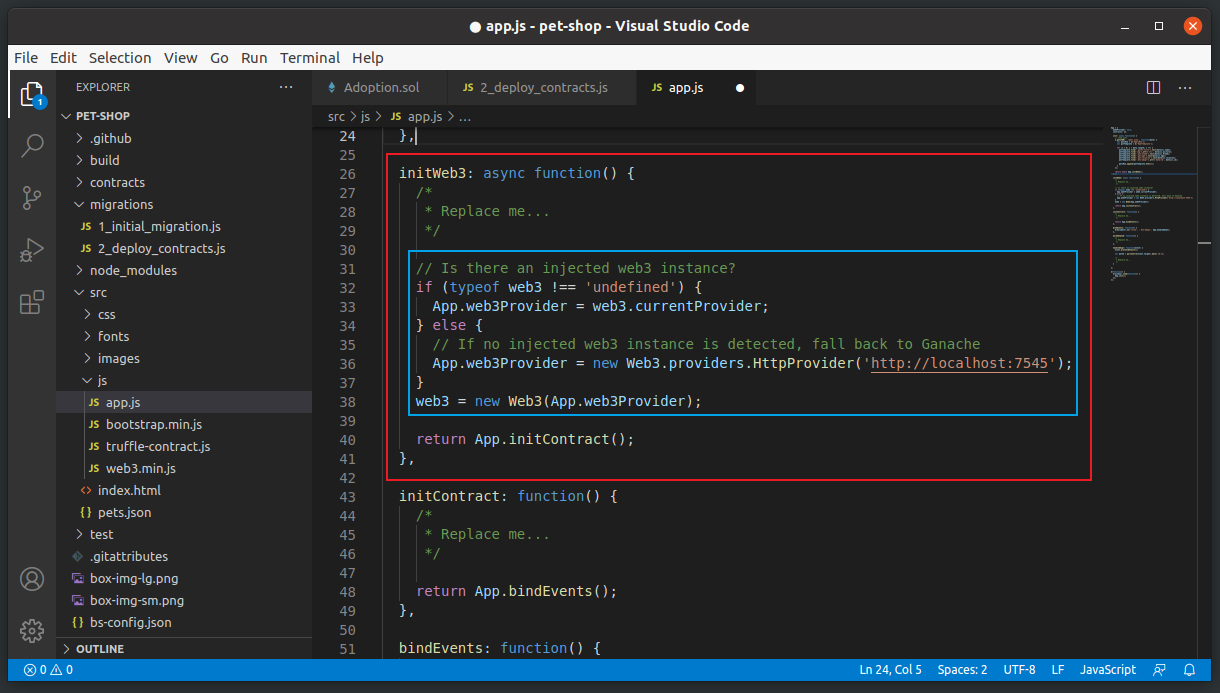


图7-11 通过web3与测试链建立连接

(2) 实例化合约

使用truffle-contract保存合约部署的信息，自动修改合约地址，添加代码如下图7-12所示：

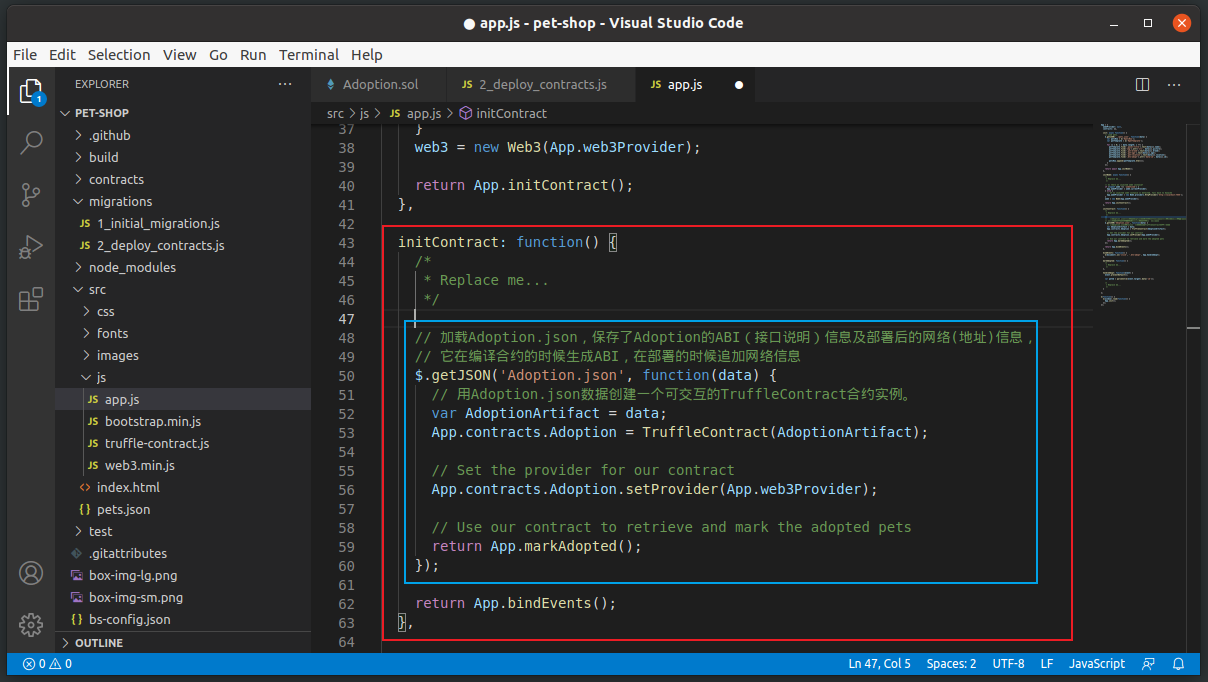


图7-12 实例化合约

(3) 处理领养

修改markAdopted: function(adopters, account)如下图7-13所示：

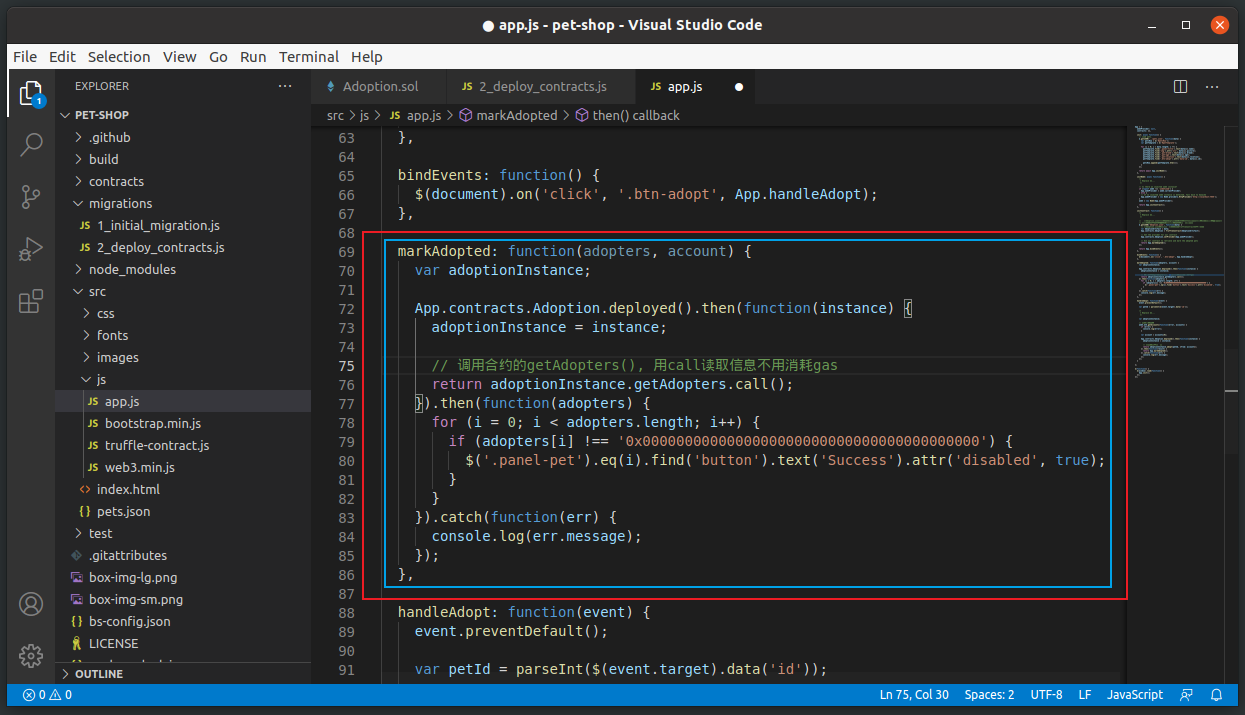


图7-13 获取领养者和宠物的信息

修改handleAdopt: function(event)如下图7-14：

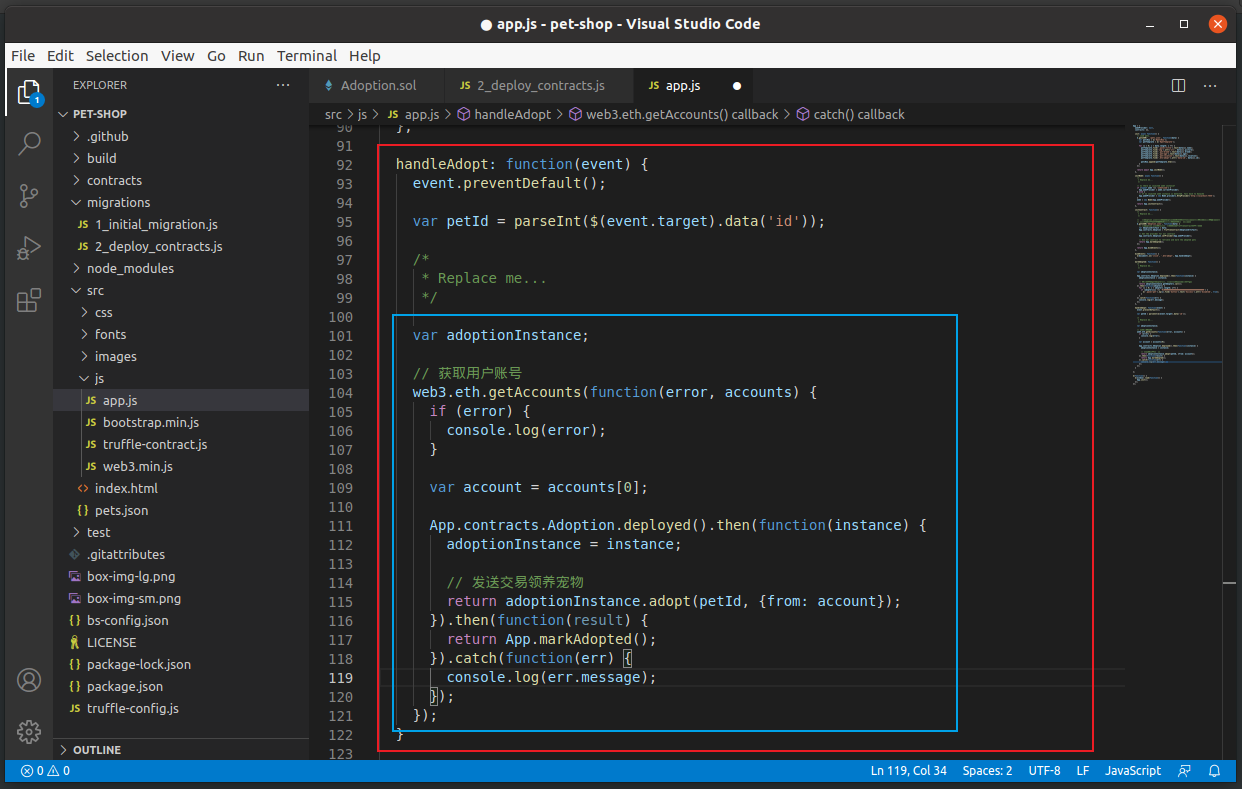


图7-14 领养宠物并更新宠物信息

**5. 在浏览器中运行**

打开chrome浏览器，点击右上方MetaMask插件，进入MetaMask，会看到下图7-15：

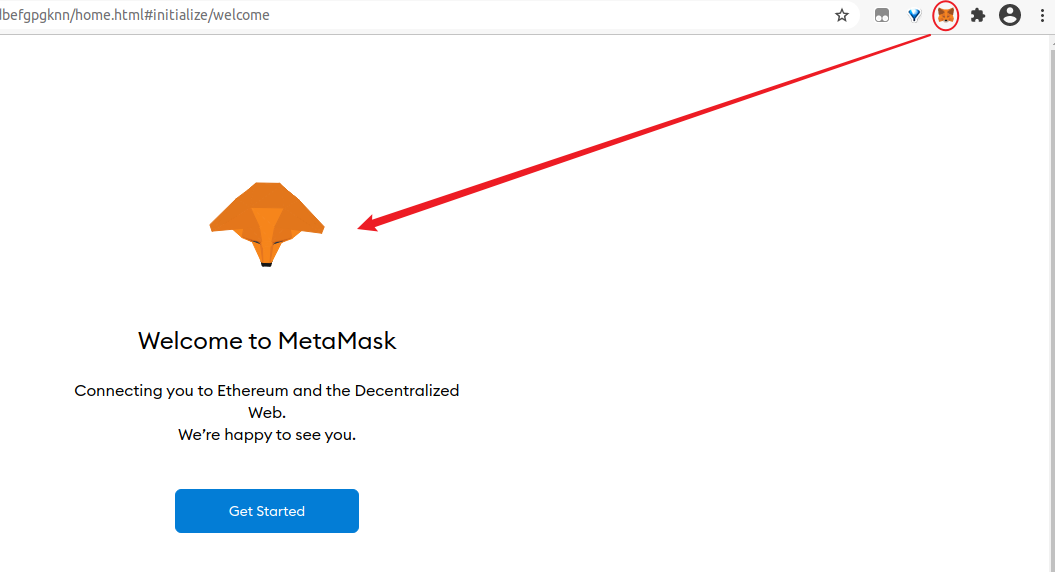


图7-15 MetaMask主界面

点击Get Start，选择左边的Import Wallet，进入输入界面，Seed phrase为Ganache显示的助记符（每次随机生成），密码为dasedase(也可自行设定)，如图7-16。

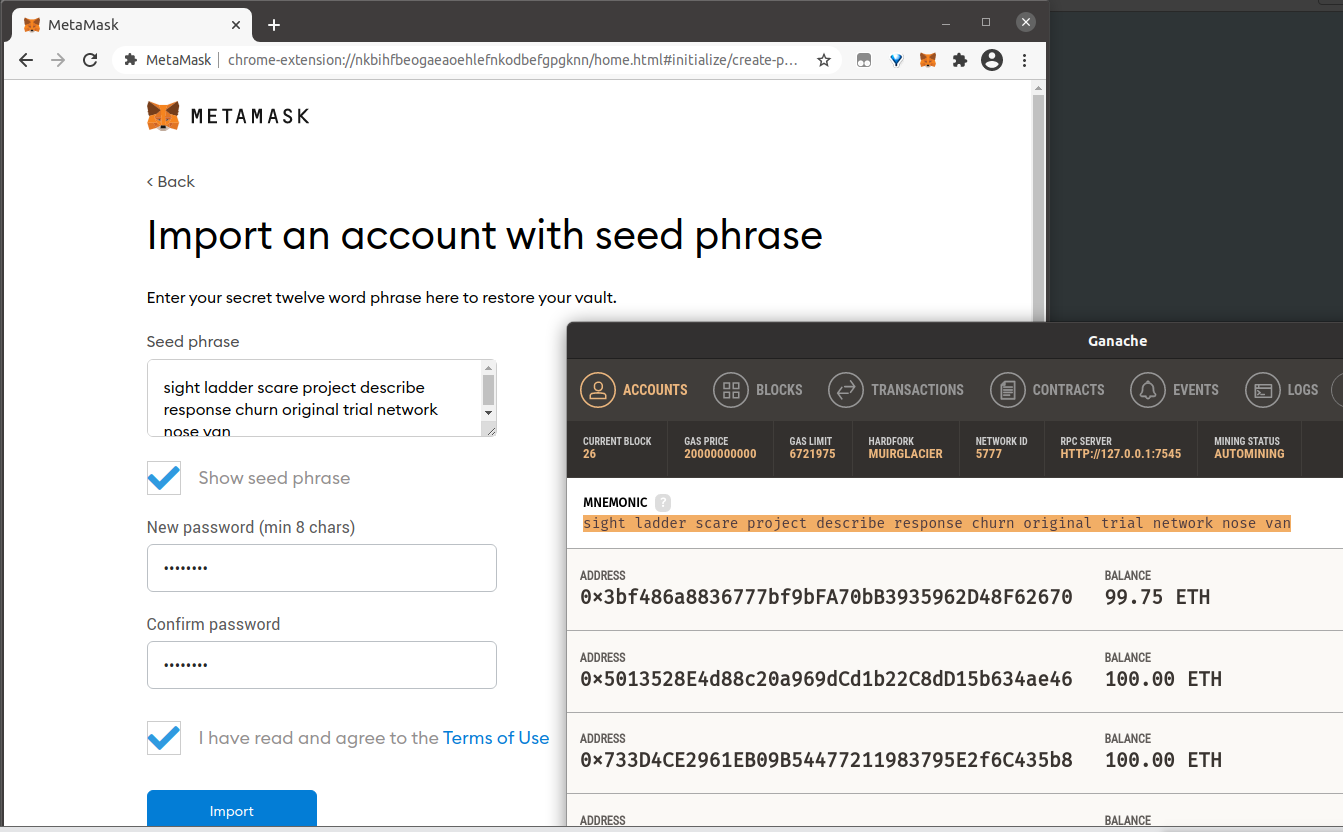


图7-16 创建账户

通过下图7-17，7-18步骤连上本次实验的测试开发链：

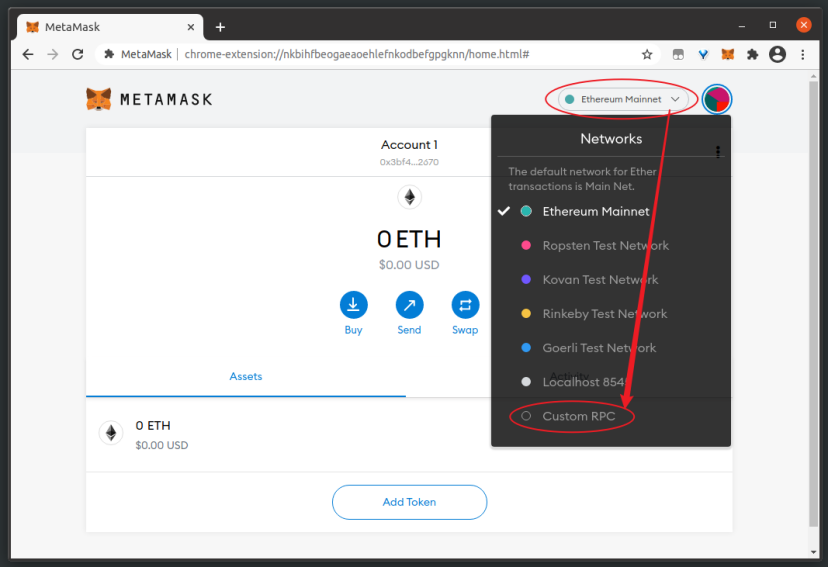


图7-17 连接测试链(上)

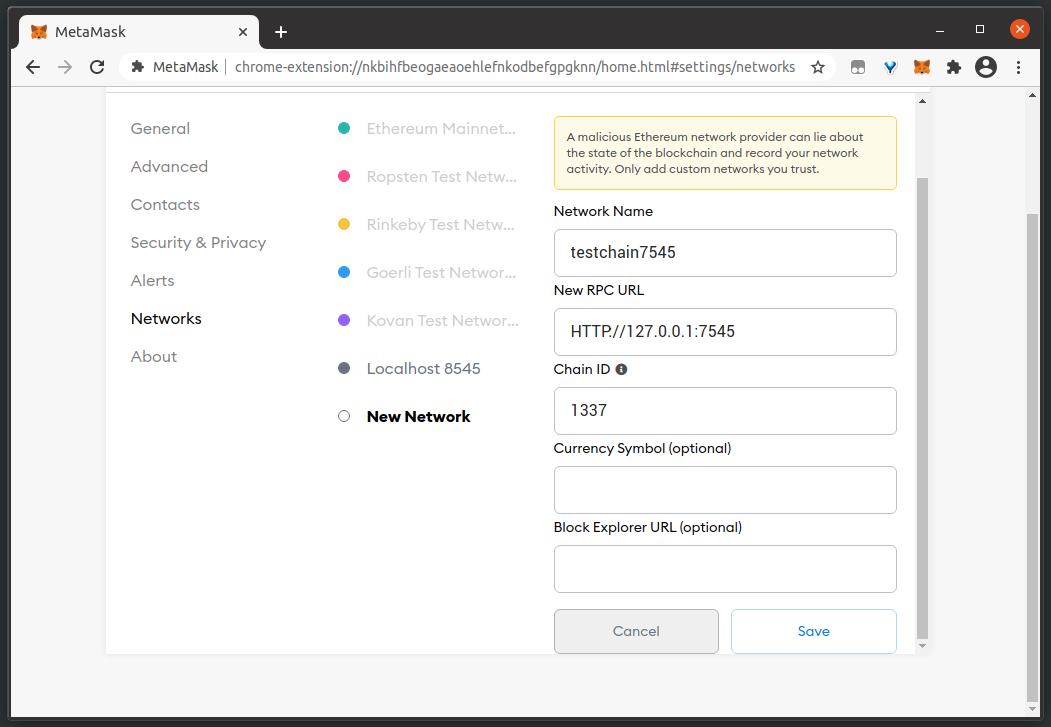


图7-18 连接测试链(下)

导入私钥的过程，如图7-19：

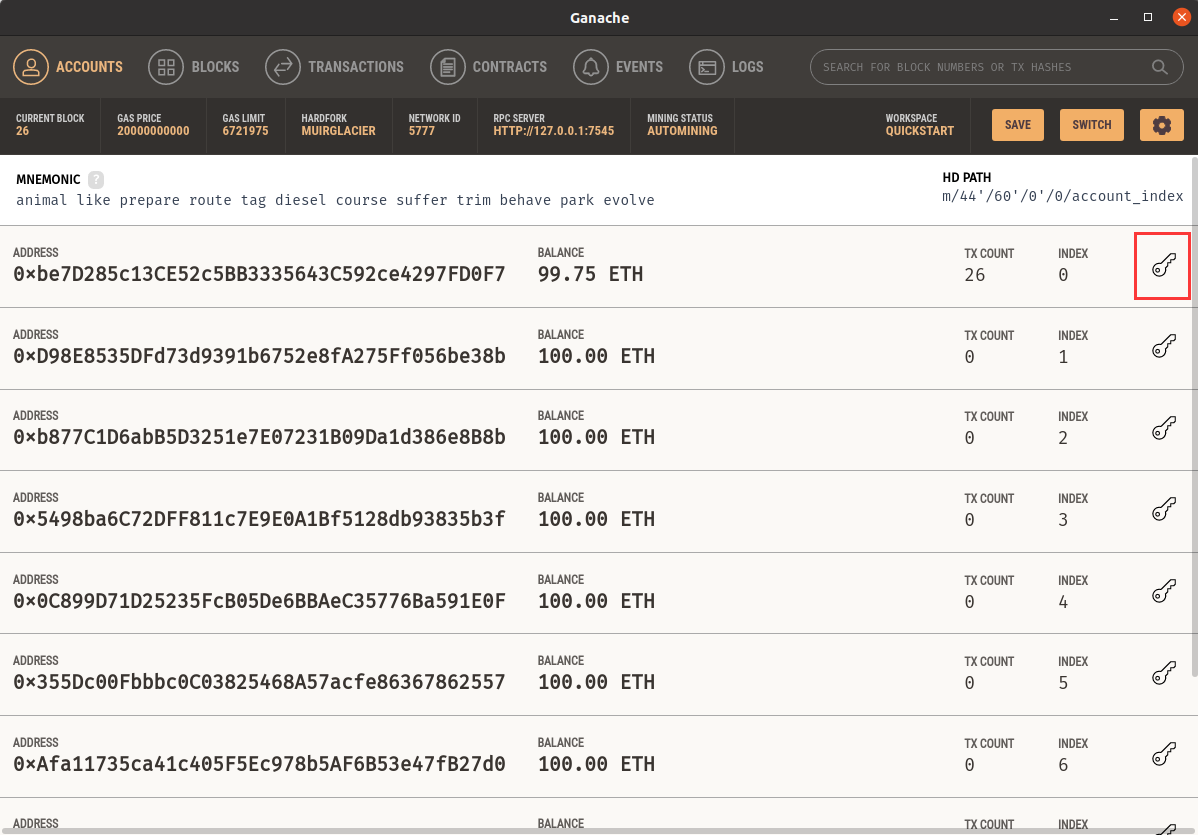


图7-19 获取私钥(上)

复制privateKey，如图7-20：

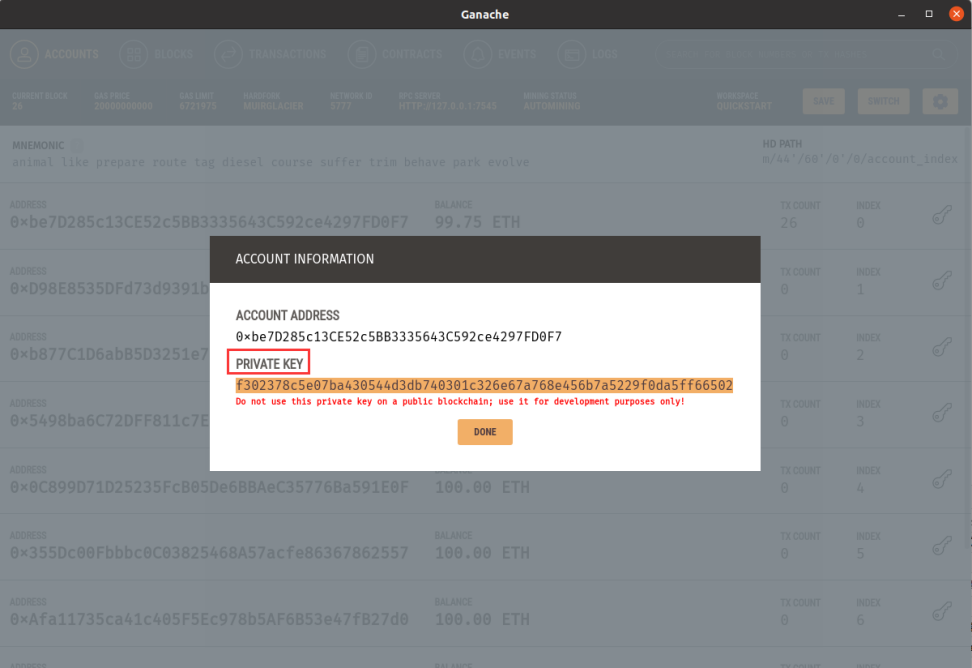


图7-20 获取私钥(下)

在MetaMask中选择导入账户，粘贴复制的私钥即可，如下图7-21,7-22所示

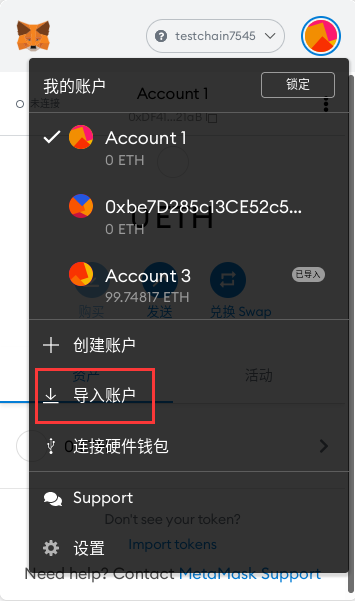


图7-21



图7-22

连上开发链后，账户1的比特币余额和Ganache中一致。

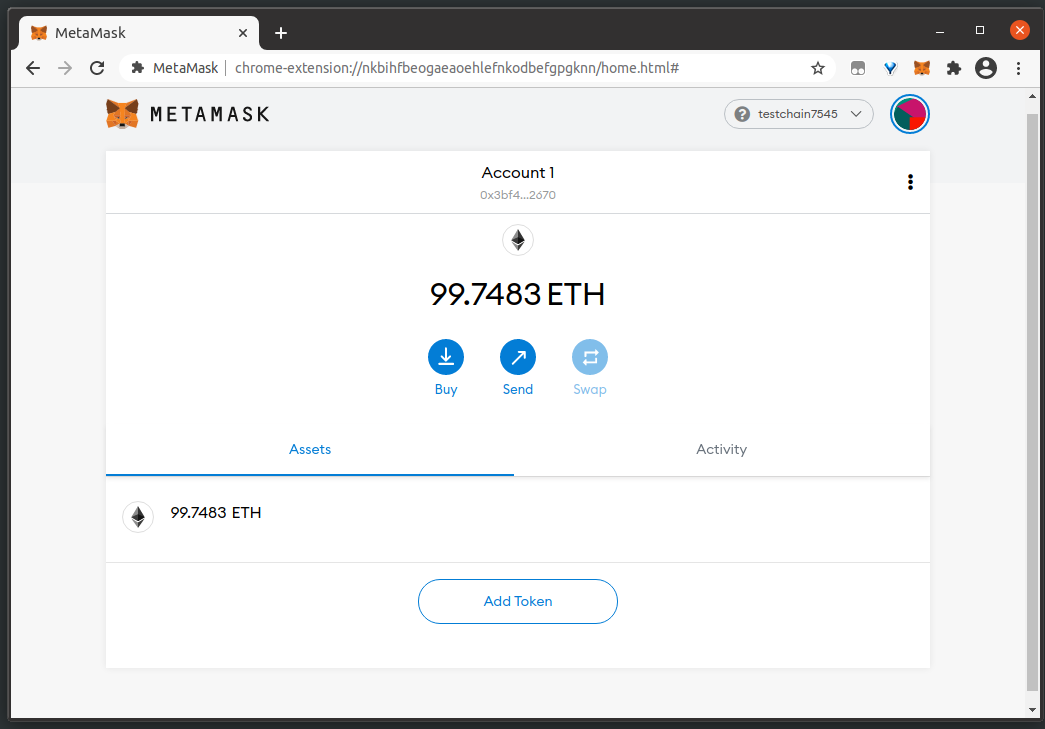


图7-23 账户中的以太币余额

**6. 启动服务**

修改index.html中的代码，将jquery.min.js的cdn源改为可访问的，如下图7-24所示。

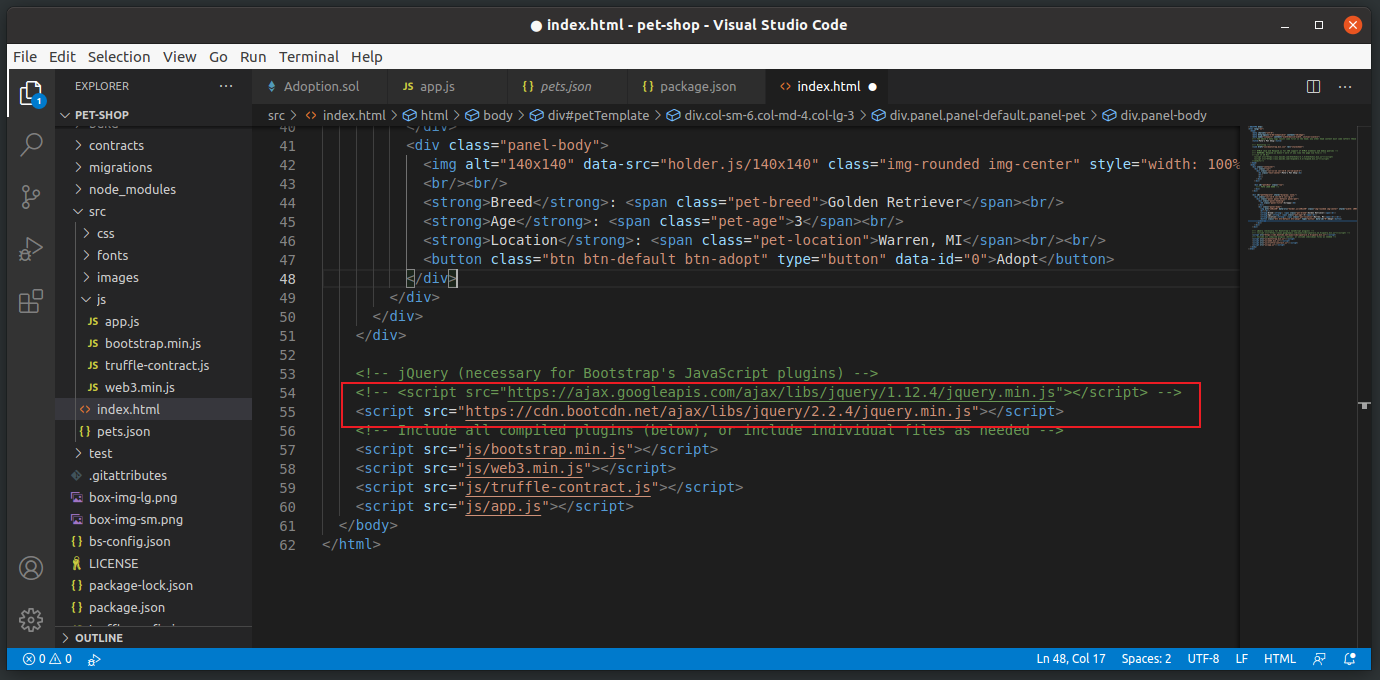


图7-24 修改cdn源

先清楚npm缓存，分别运行，得到如图7-25的结果：

|  |
| --- |
| npm cache clean –force  **rm** **-**rf node\_modules  **rm** **-**rf package-lock.json  npm install |

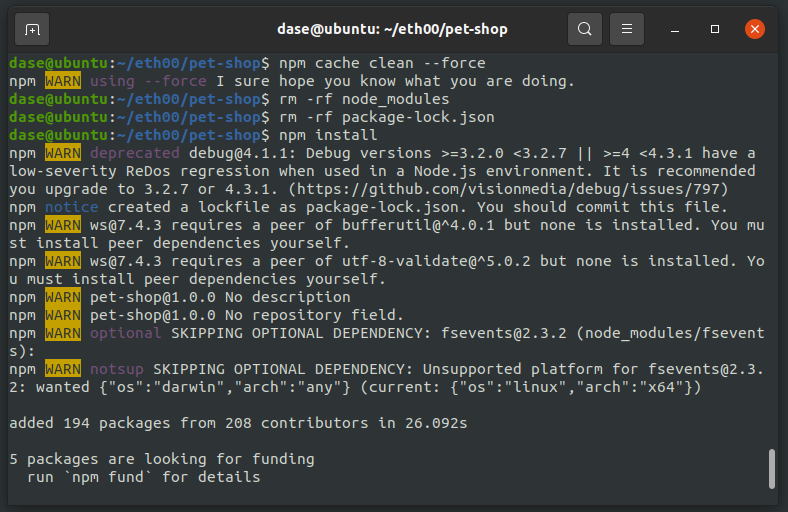


图7-25

最后运行npm run dev启动应用，弹出如下图7-26的网页：

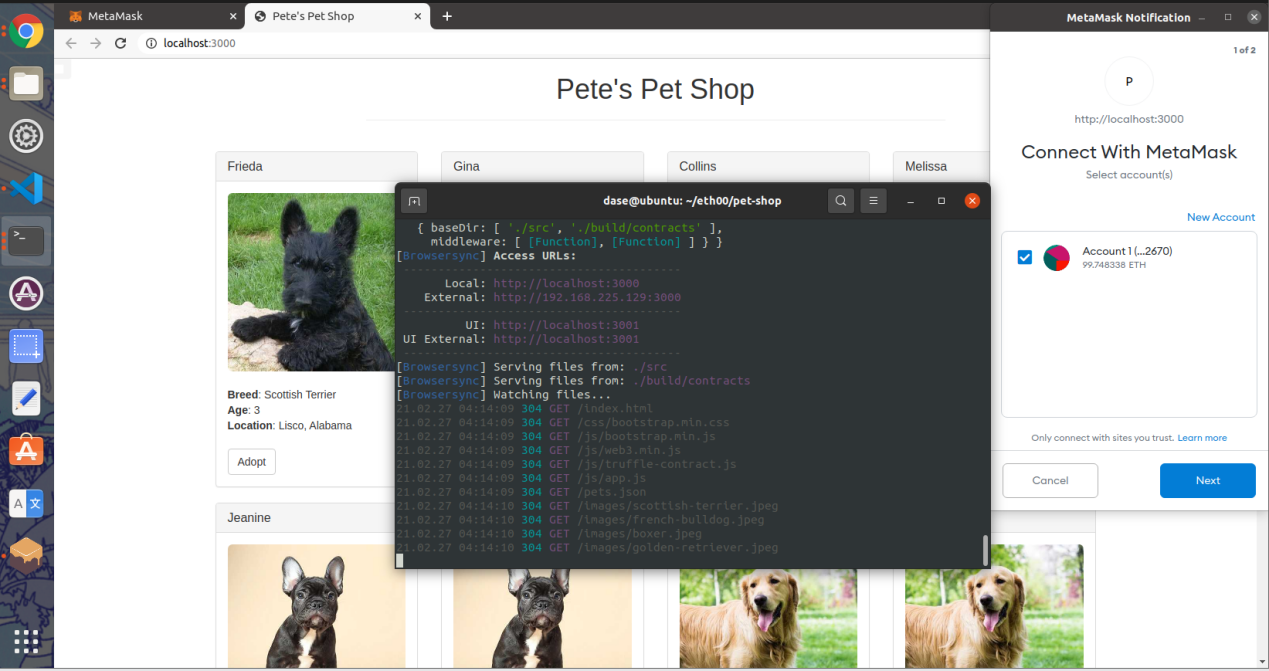


图7-26 pet-shop的Web

点击上图右方的Next，将MetaMask钱包连接到本应用中，然后后面点击任一Adopt按钮，MetaMask会弹出相应的交易确认界面，如下图7-27所示：

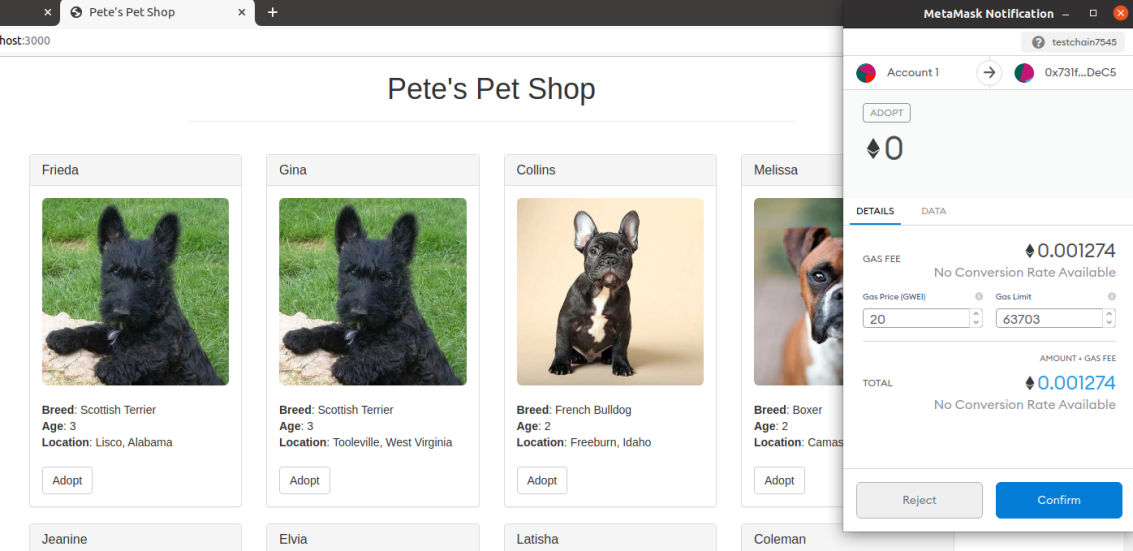


图7-27 确认交易

确认后将收养成功（这里我又收养了两只），交易记录被保存在链上，从Ganache中可查询到相关区块和交易记录。

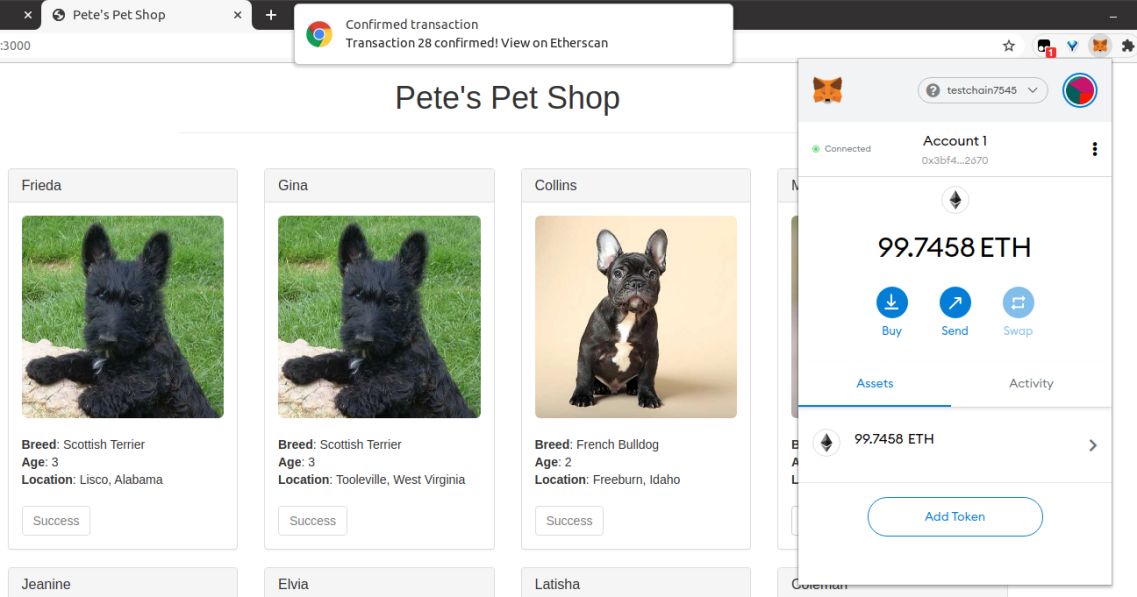


图7-28

**【实验小结】**

本实验介绍并实现一个简单的DAPP去中心化应用，通过区块链技术对于APP这一领域的改变展示了区块链技术带来的创新性。其主要的不同之处在于将原有的一些中心化的技术和存储扩展到了去中心化的整个区块链网络中，通过智能合约来实现其业务逻辑，并由整个区块链网络对数据进行维护和存储，这对于APP的扩展有着很大的帮助。本实验就是通过实现pet-shop这个DAPP和观察修改其代码逻辑，让大家对于DAPP是如何实现以及如何运作有一个初步的了解。

**【习题】**

1. 实验中，有一处是通过智能合约调用智能合约的，请简要说明一下其流程。

2. 实验介绍提到了DAPP(Cryptokitties)使得区块链网络堵塞，尝试说明一下导致这一结果的原因(从区块链生成块的过程入手)

3. 实验过程中，哪些过程会产生花费？

**4. 大家会发现在我们设置的领养合约中，仅仅是简单的将宠物分配给调用领养合约的账户，而实际上应该要给宠物规定一个价格，且最好是有一个宠物店的店主来获得这些收入，有兴趣的同学可以尝试通过修改其中脚本文件和合约的代码来实现这一功能(选做)。**

**【参考文献】**

1. 杨保华. 区块链原理, 设计与应用[M]. 机械工业出版社, 2017.
2. Andreas M. Antonopoulos, Dr Gavin Wood. Master Ethereum: Building Smart Contracts and DApps(精通以太坊: 开发智能合约和去中心化应用) 机械工业出版社
3. 贺海武, 延安, 陈泽华. 基于区块链的智能合约技术与应用综述[J]. 计算机研究与发展, 2018, 55(11): 2452.
4. Wang S, Ouyang L, Yuan Y, et al. Blockchain-enabled smart contracts: architecture, applications, and future trends[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(11): 2266-2277.