电子科技大学计算机科学与工程学

院（网络空间安全学院）

实 验 **报 告**

（实验） 课程名称： 区块链原理与开发（新生项

目课程）

**小组成员：**

**电子科技大学教务处制表**

目录

（实验）课程名称： 区块链原理与开发（新生项目课程） [1](#bookmark2)

一、 实验目的 [1](#bookmark3)

二、 实验内容 [1](#bookmark4)

三、 实验环境 [1](#bookmark5)

四、 预备知识 [1](#bookmark6)

五、 实验步骤 [2](#bookmark1)

5.1 GANACHE 和 METAMASK 的配置 [2](#bookmark1)

*5.1.1 配置Ganache*   *[2](#bookmark1)*

*5.1.2 配置METAMASK*  *[4](#bookmark7)*

5.2 智能合约的编译与部署 [8](#bookmark8)

*5.2.1 编译智能合约* *[8](#bookmark9)*

*5.2.2 部署智能合约* *[9](#bookmark10)*

5.3 前端页面设计与区块链交互 [11](#bookmark11)

5.4 结果展示 [12](#bookmark12)

**六、 实验总结………………………………………………………………………………………………………………14**

附录 16

附录 1: HTML 代码 17

附录 2: JS 代码 18

**一、 实验目的**

通过区块链技术，实现一个博彩系统。

该博彩系统可以实现的功能是：多个玩家通过投入一定的成本加入博彩奖池，在时间截止之后，由主办方进行赢家的挑选，并将奖池中的全部资金都分发给玩家。

**二、 实验内容**

本实验通过 Ganache 搭建私链， 使用 METAMASK 进行交互， 随后使用 Remix 编写 智能合约，最后将合约部署到前端，实现了一个简单的博彩系统。

**三、 实验环境**

具体开发环境有两种：一是 Geth +js, 二是 Ganache +js 。虽然我们可以用 Geth 搭一 条私链进行开发，但以太坊比特币需要通过每个区块挖矿产生， 块与块之间有一定的时间间 隔，而且用 PC 挖矿时间长，产热高，使我们的开发效率比较低。所以本实验选择通过 Ganache 在内存中模拟区块链来开发。

其他工具**:**

1. METAMASK：以太坊钱包。能帮助用户方便地管理自己的以太坊数字资产。

2. Remix：开源的 Solidity 智能合约开发环境， 提供基本的编译、部署至本地或测试网络、 执行合约等功能。

**四、 预备知识**

1. 区块链基础知识；

2. Solidity 语言；

3. 前端开发基础知识： HTML/CSS/JavaScript

**五、 实验步骤**

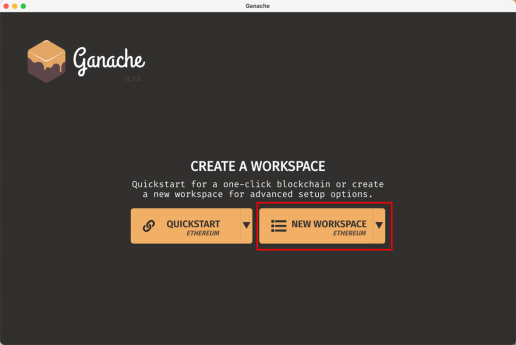
**5.1 Ganache** 和 **METAMASK** 的配置

5.1.1 配置 Ganache

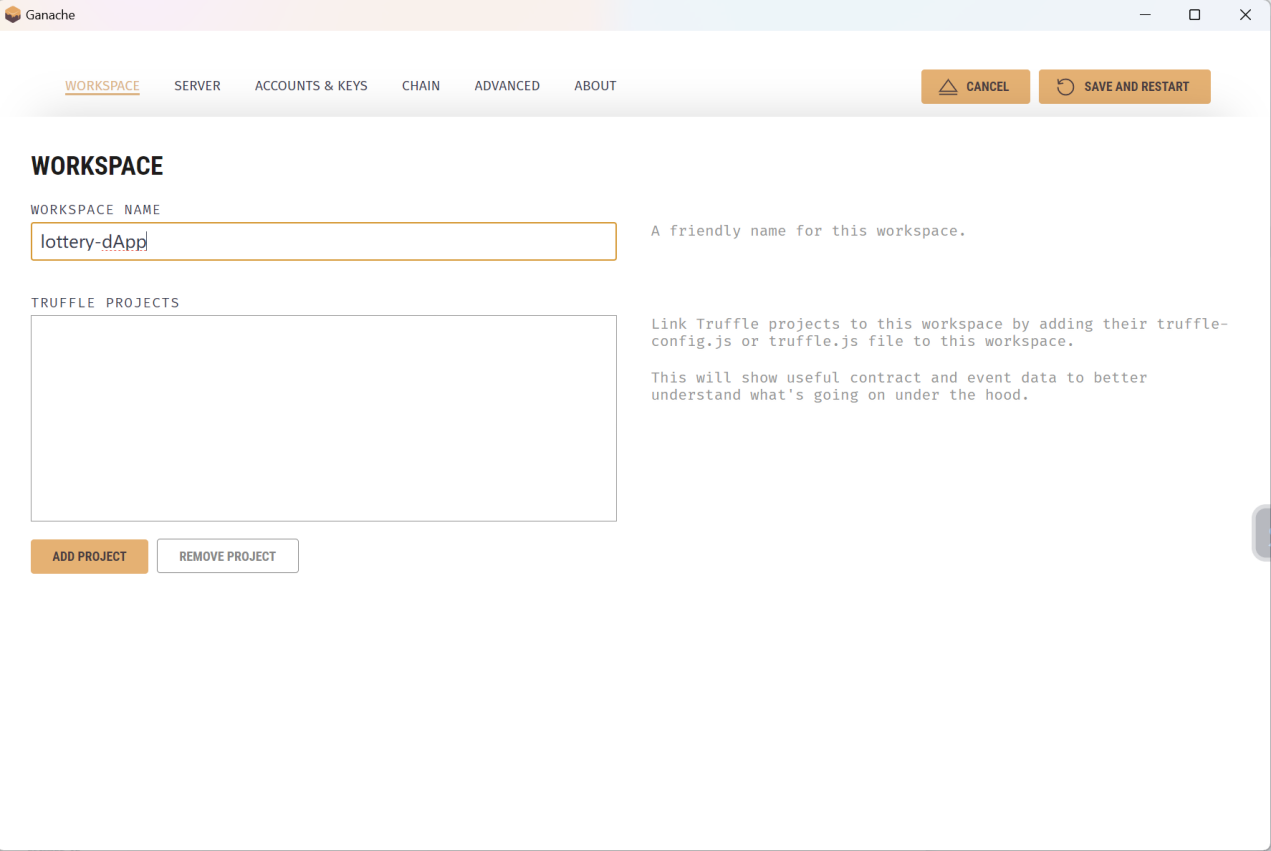
什么是 Ganache？Ganache 是一个用于本地开发的区块链， 用于在以太坊区块链上开发去中 心化的应用程序。Ganache 模拟了以太坊网络，你可以在发布到生产环境之前看到你的 DApp 将如何执行。

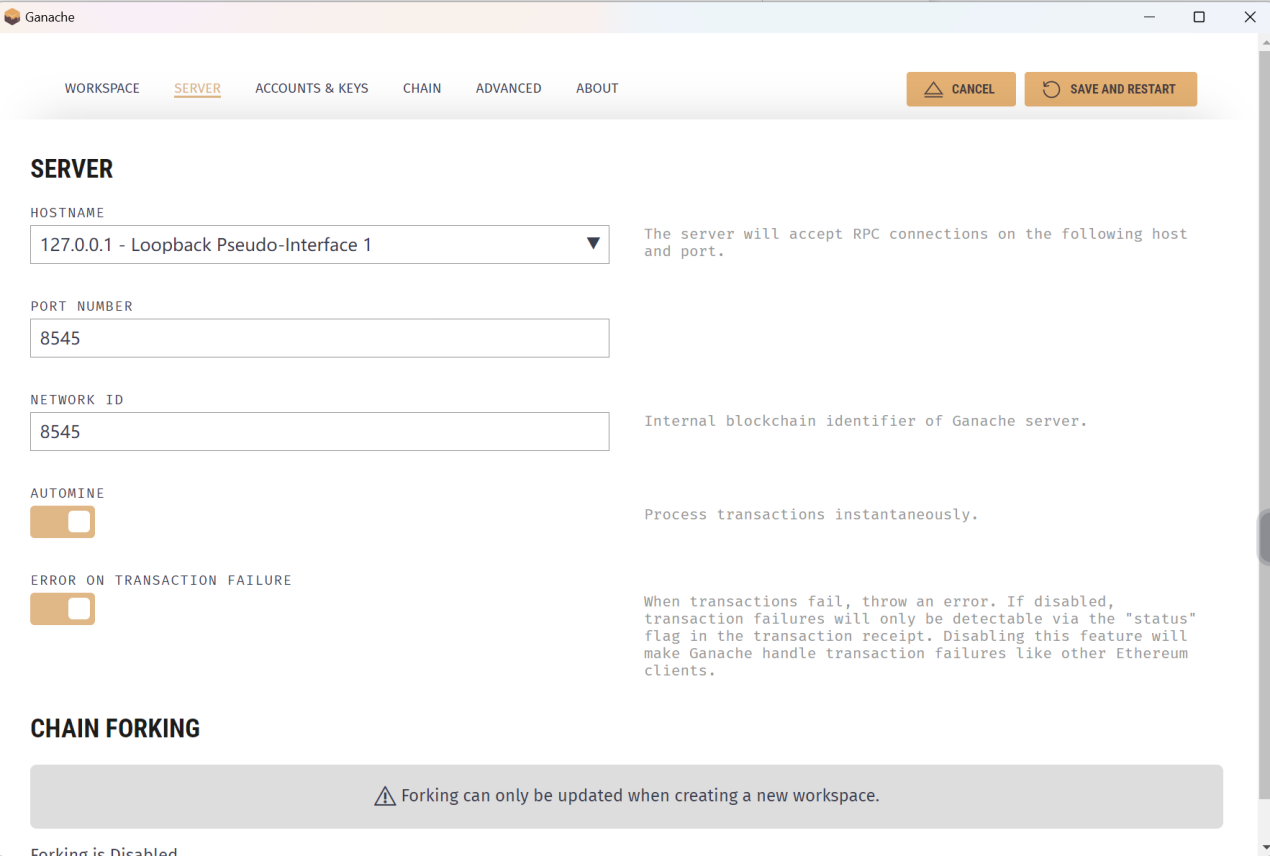
1. 下载 Ganache：<https://trufflesuite.com/ganache/>

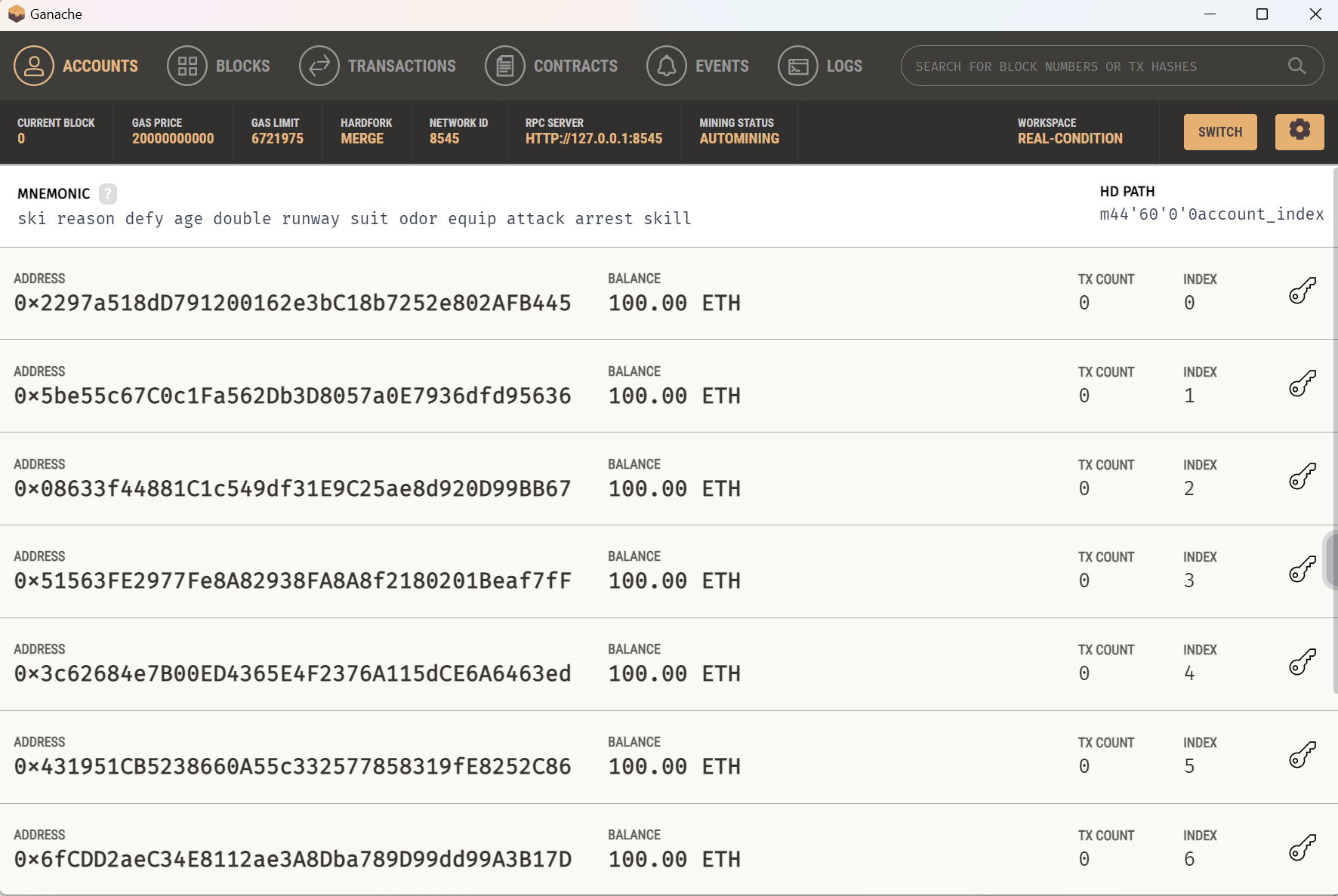
2. 下载完成后， 打开 Ganache，左边的按钮是快速启动，它的数据不会保存，每次启动后 都是全新的开发环境。右边的按钮是保存当前的数据到相应的工作空间中， 可以有多个 工作空间。我们选择右边的 new workspace：



3. 为 workspace 取一个有意义的名字，这里我们设置为 lottery-dApp：

4. 点击导航栏的 server，可以看到 RPC 服务相关设置。将 PORT NUMBER 修改为 8545 ， 修改端口号的原因是为了后续能使私链与 METAMASK 正常连接，因为 METAMASK 中默认的 ID 为 8545：

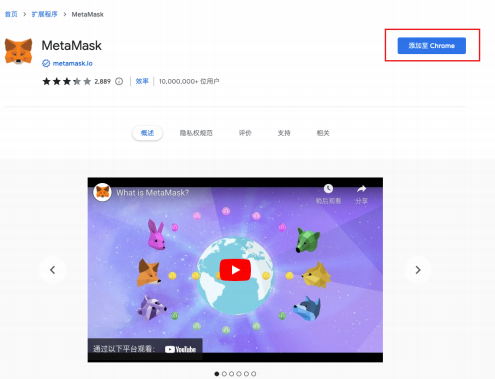
5. 点击导航栏的 Accounts & Keys，可以设置初始账号数量和账号的初始金额，我们保持默认。最后点击右上角的 start ，我们的本地以太坊就可用了。搭建完成后，可以看到私链中自动生成了存有以太币的十个账户：



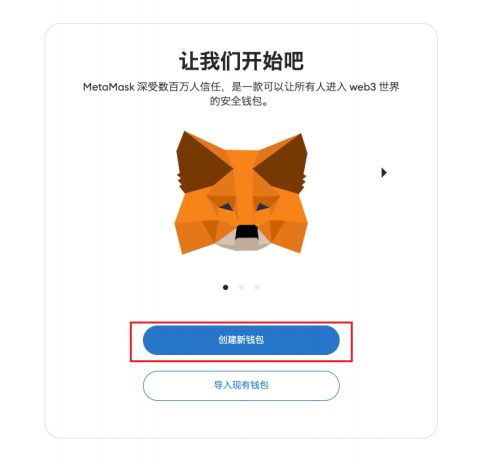
5.1.2 配置 METAMASK

什么是 METAMASK？METAMASK 是一个轻量化的以太坊钱包， 支持私人网络及自定义 RPC。在部署和调用合约时，需要消耗 gas，我们使用 Ganache 提供的本地私链和虚拟账户 通过 METAMASK 来模拟这一步骤。

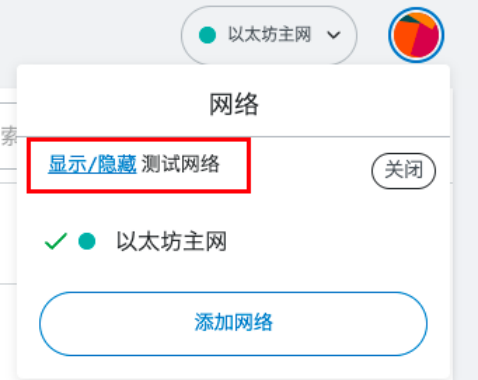
1. Google Chrome 浏览器安装 METAMASK 插件：<https://metamask.io/>



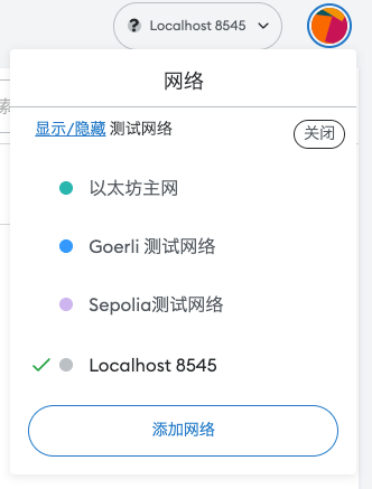
2. 打开 METAMASK 插件，选择创建新钱包



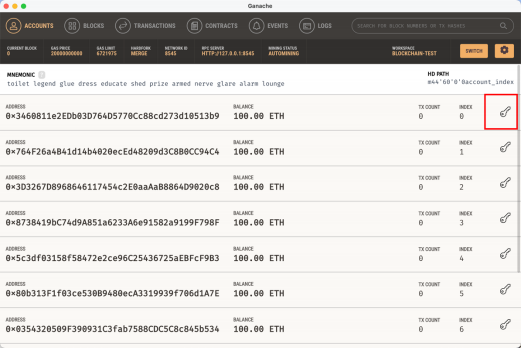
3. 创建完成之后， 点击浏览器右上角的 METAMASK 图标，显示测试网络：

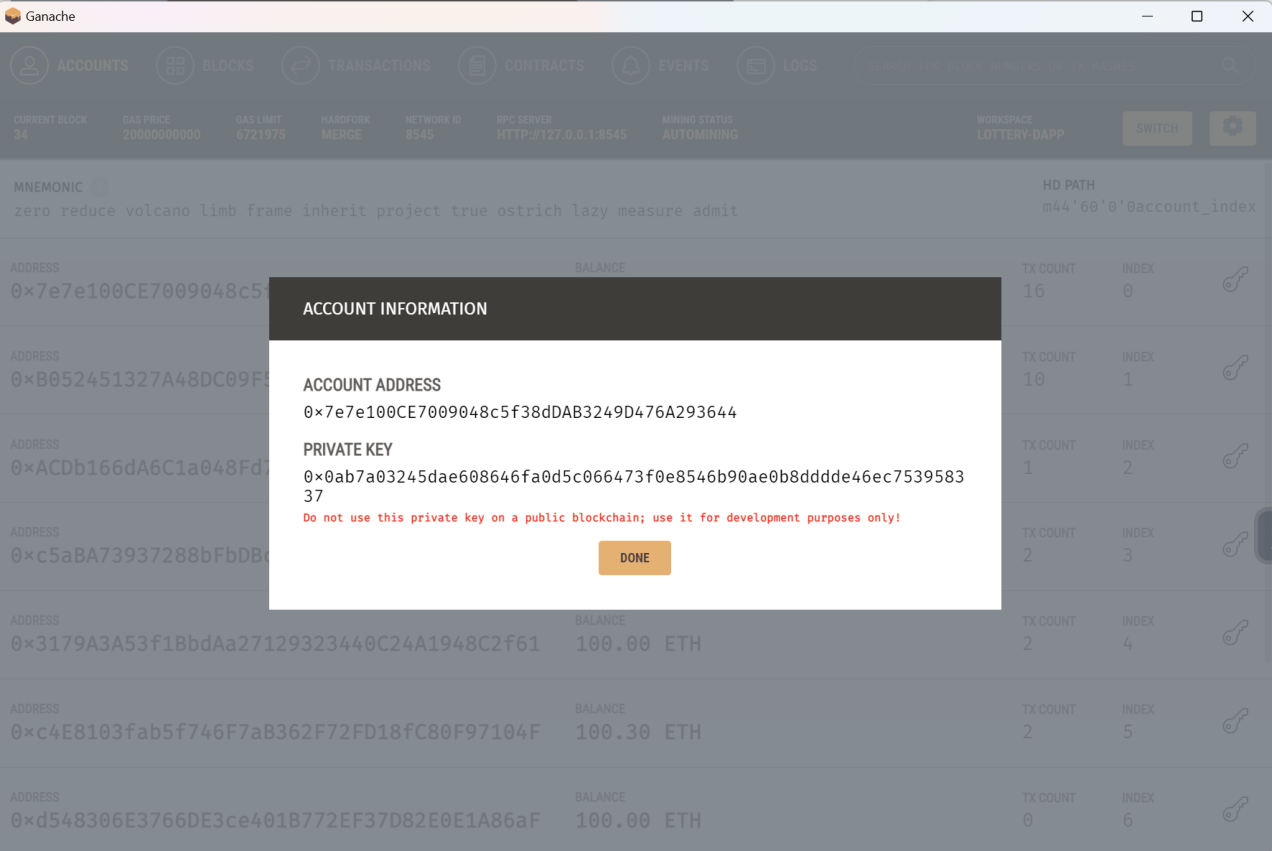


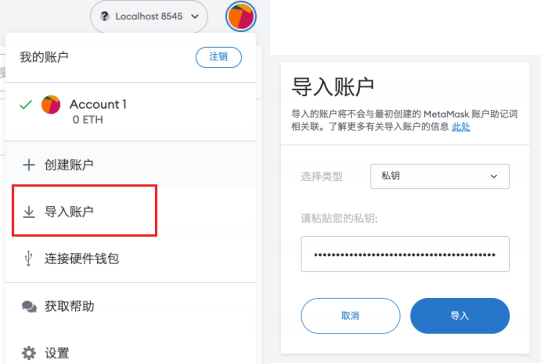
4. 显示测试网络之后，将网络设置为 Localhost 8545



5. 复制 Ganache 生成的 10 个账户中的多个私钥，这里选择第一个作为演示：



6. 打开 METAMASK 插件，选择导入账户， 粘贴刚刚复制的私钥：



7. 导入成功后可以看到下图：



**5.2** 智能合约的编译与部署

5.2.1 编译智能合约

什么是 Solidity？Solidity 是一门面向合约的、为实现智能合约而创建的高级编程语言。这 门语言受到了 C++ ，Python 和 Javascript 语言的影响，能在以太坊虚拟机 (EVM)上运行。

1. 在本实验中， 我们使用 solidity 实现智能合约的编写。以下是具体的代码:

//SPDX-License-Identifier:GPL-3.0

pragma solidity ^0.8.11;

contract Lottery {

address public owner;

address payable[] public players;//used for any addresses or functions that can receive Ether

uint public lotteryId;

mapping (uint => address payable)public lotteryHistory;

//构造函数

constructor(){

owner=msg.sender;

lotteryId=1;

//将部署人的地址设置为我们的所有者状态变量

}

function getWinnerByLottery(uint lottery) public view returns (address payable){

return lotteryHistory[lottery];

}

//输入函数

function getBalance() public view returns (uint){//数据可视化

return address(this).balance;

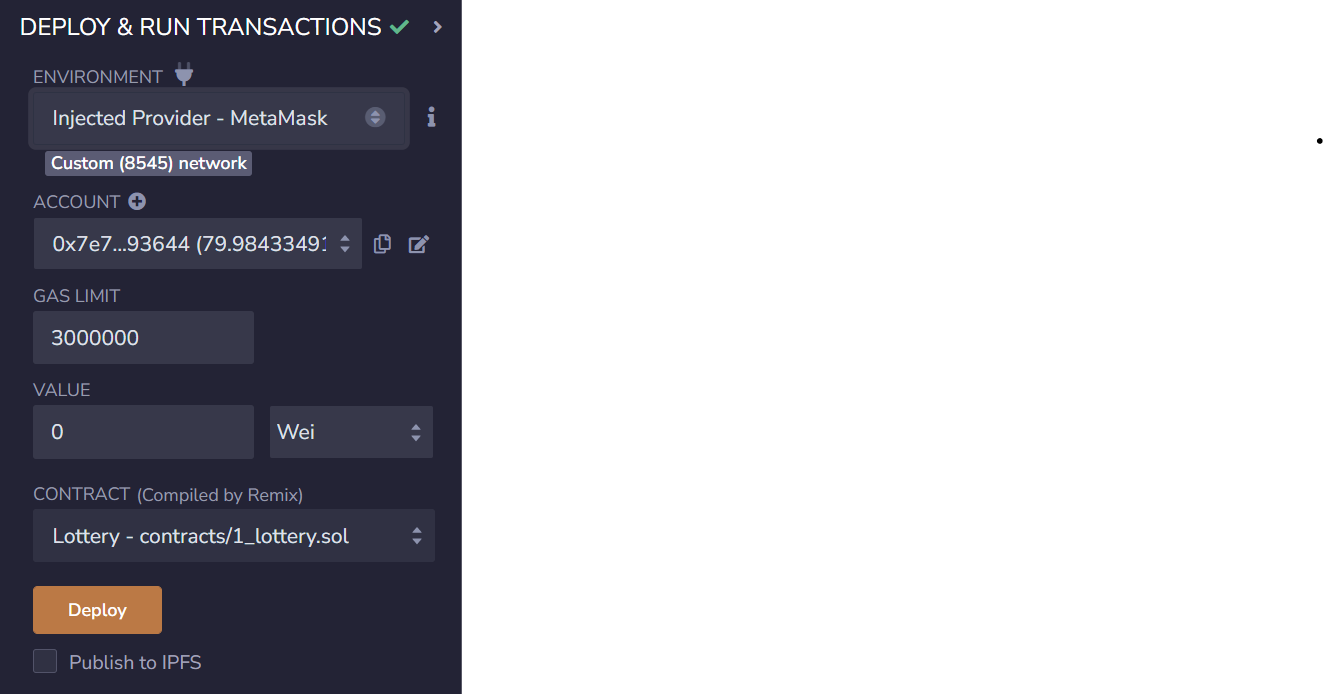
}

function getPlayers() public view returns (address payable[] memory){

return players;

}

|  |
| --- |
| function enter() public payable{  //需要输入值  require(msg.value >= .01 ether);  //把进入抽奖的玩家的地址加入玩家数组  players.push (payable(msg.sender));  }  //伪随机数，利用散列算法  function getRandomNumber() public view returns(uint){  return uint(keccak256(abi.encodePacked(owner,block.timestamp)));  }  function pickWinner() public onlyOwner {  //get an index from the array above    uint index = getRandomNumber() % players.length;  players[index].transfer(address(this).balance);  //address(this)gives us a reference to the current smart contract  //balance gives us a reference to the balance of that contract(the value sent by players)  //重入攻击,先转移再更新状态  lotteryHistory[lotteryId]=players[index];  lotteryId++;    players=new address payable[](0);//reset,for next round  }  modifier onlyOwner() {  require(msg.sender==owner);  \_;  }  } |
| 2. 使用 Remix 实现智能合约的编译：<http://remix.ethereum.org/> （Remix 是以太坊的一个在线集成开发环境）  3. 打开 Remix 之后， 在 contracts 文件夹里创建一个后缀为 voting.sol 的文件，将合约代码复制进去， 然后编译，编译成功如下图：    5.2.2 部署智能合约   1. 首先将 Remix 中的 ENVIRONMENT 选成 Injected Provider - MetaMask； 2. 在完成这一步之后，通过切换MetaMask上的Account，Remix中的账户也会随之发生变化； |

3. 点击 Deploy 按钮后， METAMASK 会自动跳出来，点击“确定”：

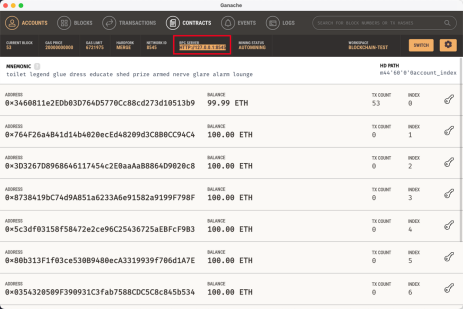


**5.3** 前端页面设计与区块链交互

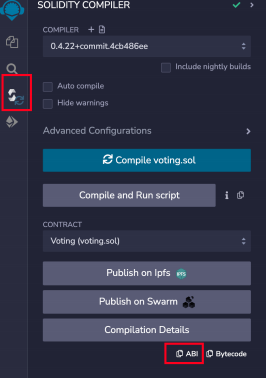
1. 本实验通过 HTML 实现前端页面设计，并与区块链实现交互。 HTML 代码见附录 1； 2. HTML 中通过 Javascript 来实现调用区块链中的方法（上述 HTML 代码引用的是

index.js 文件，因此需要将与区块链的交互逻辑添加到名为 index.js 的文件中， 并与本 HTML 文件放置于同一 目录下，**js** 代码中需要修改的三个变量为 **server**、**abi**、**address** ， js 代码见附录 2；

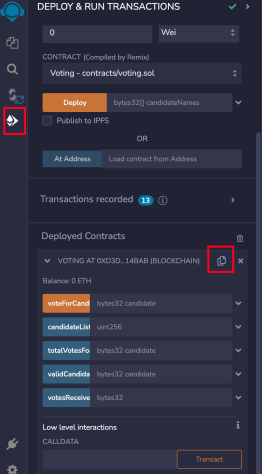
1. 获取 server ：Ganache 的 RPC SERVER 复制过来即可



1. 获取 abi：Remix 中复制

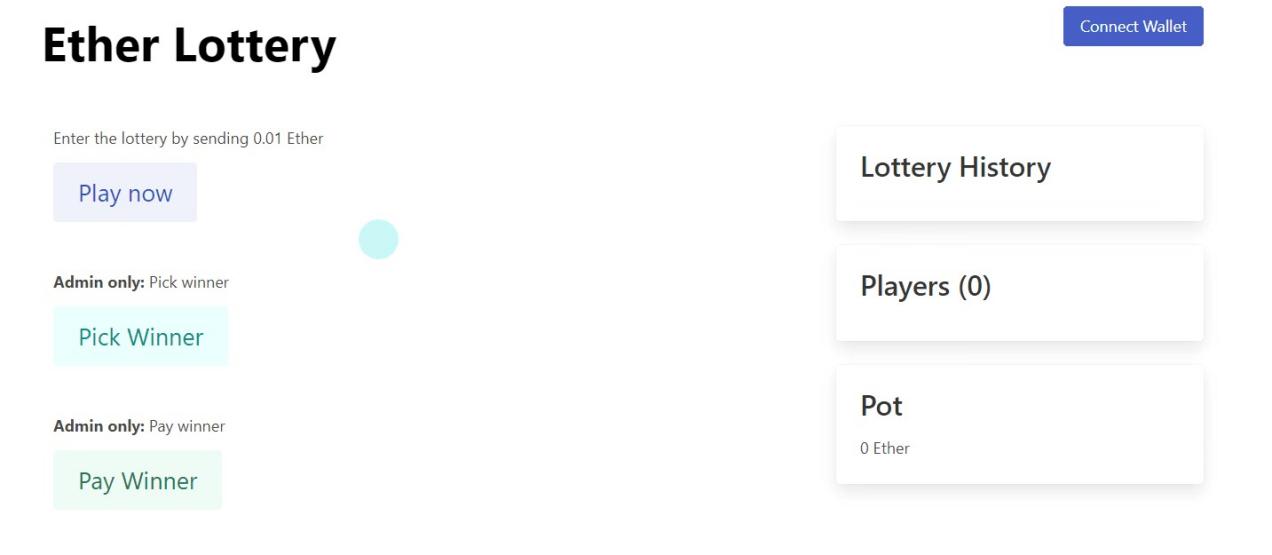


1. 获取 address：Remix 中复制

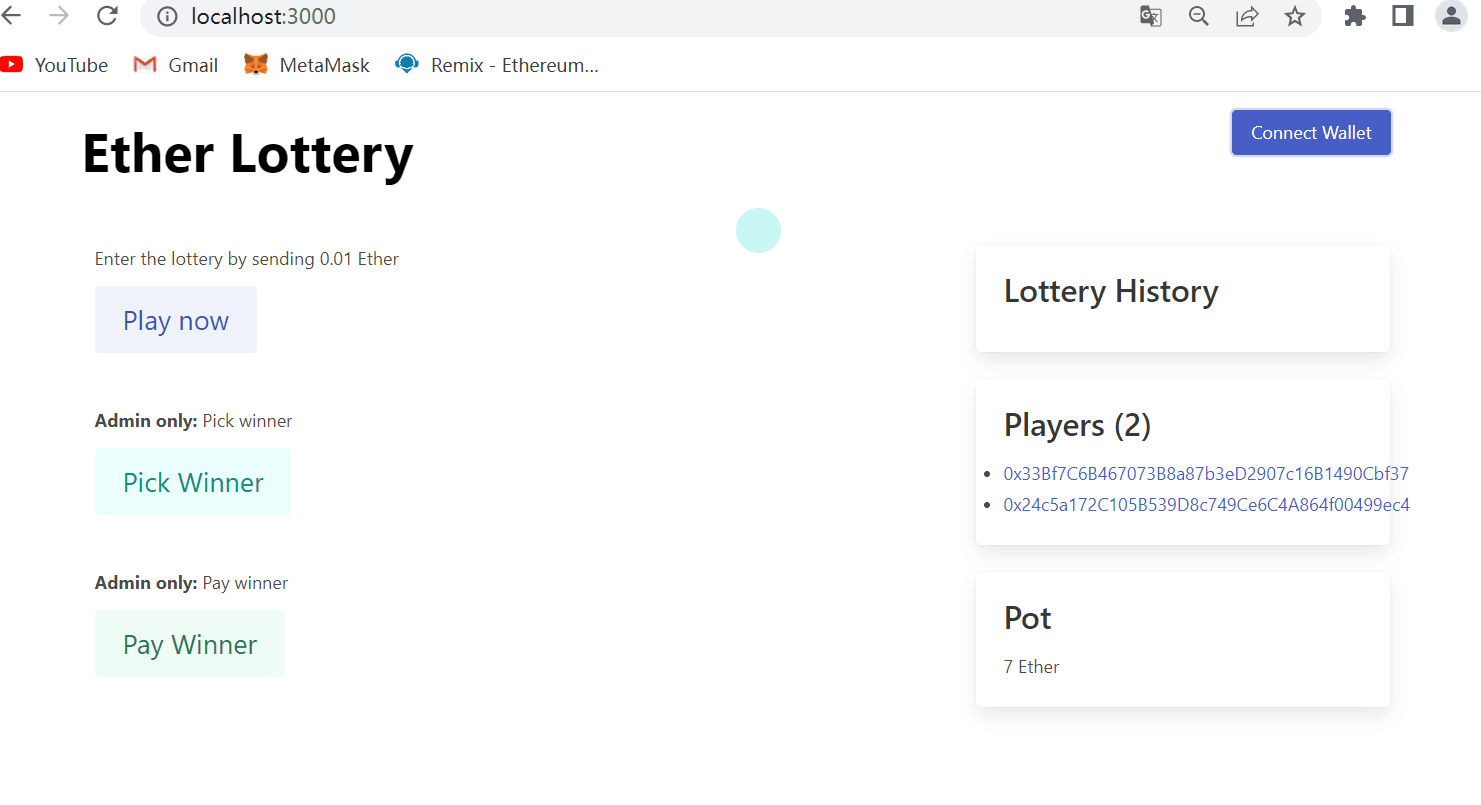


**5.4** 结果展示

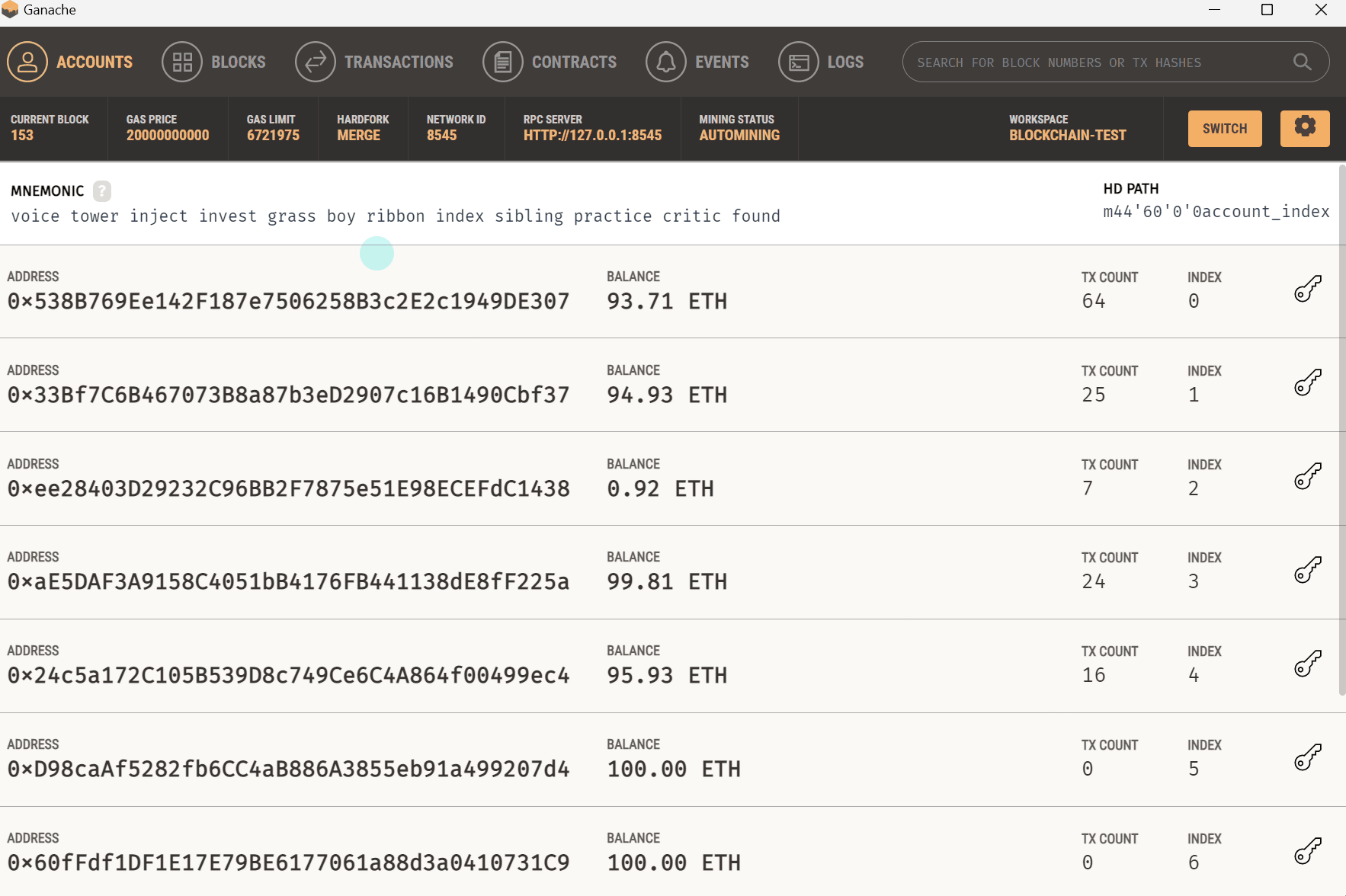
1. 使用谷歌浏览器直接打开 https://localhost:3000，可以看到页面中有玩家操作的按钮，点击Play Now，就可以向奖池中投入固定的钱（0.015Ether），而右边栏中会显示Players



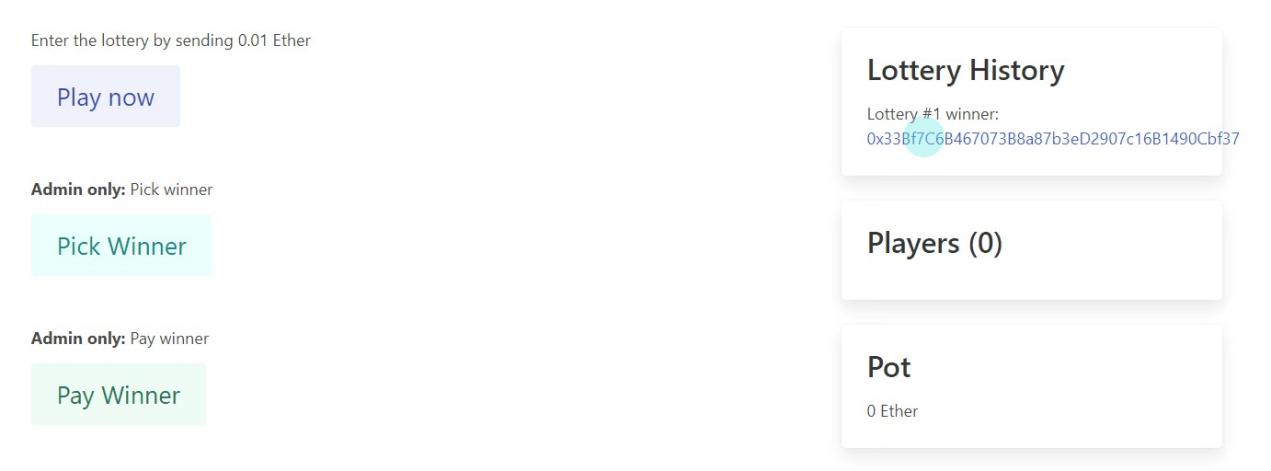
2. 其中一个玩家向Pot中投入Ether，发现Players会增加，会显示其Address，同时Pot中的钱数也会进行增加。



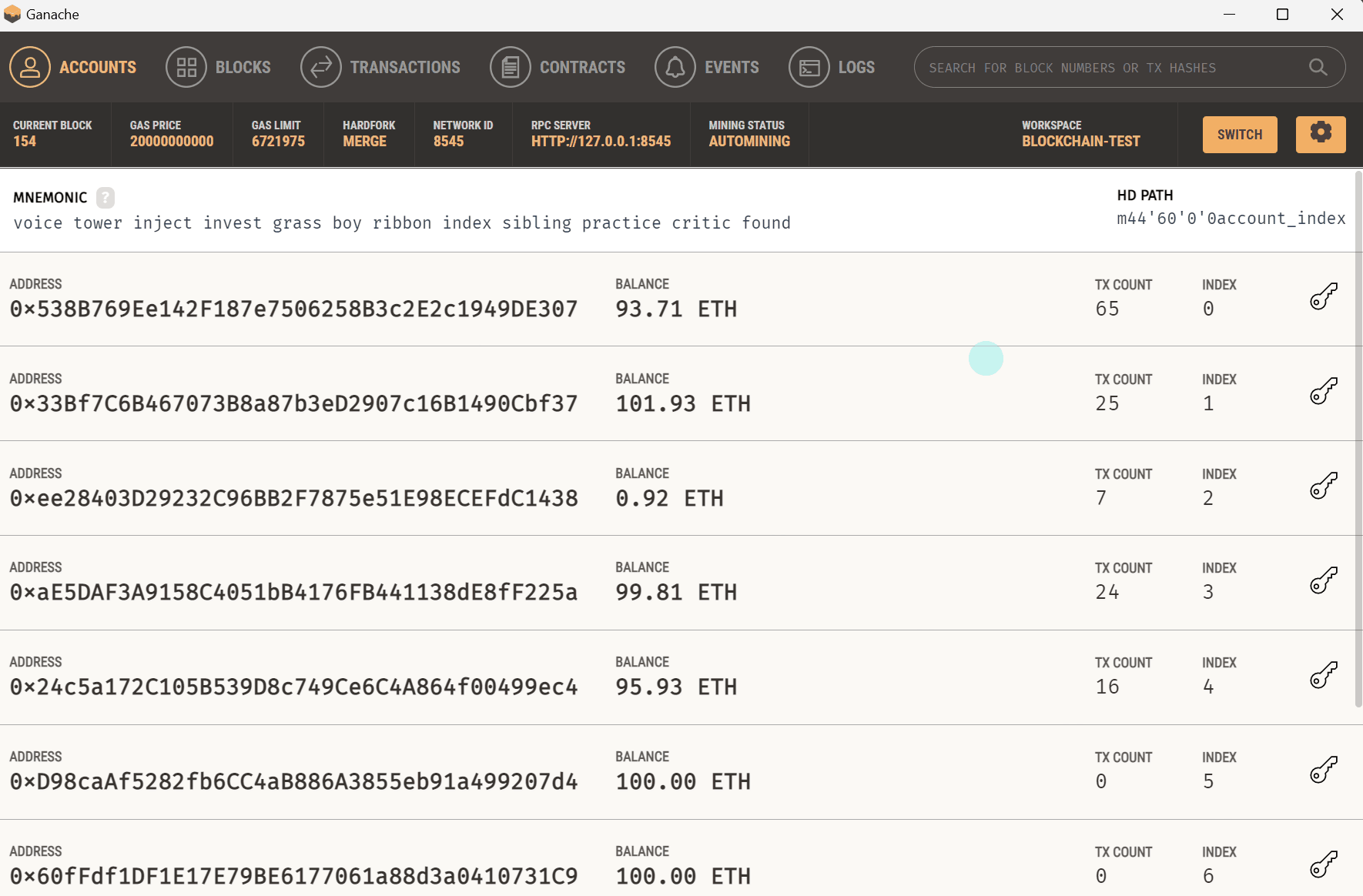
3. 此时去查看 Ganache，可以看到 TX COUNT 也增加，玩家的钱数相应减少：



4. 时间截止后，我们回到主办方，由主办方进行赢家的挑选



5.可以看到Lottery History中已经有了更新，Account2中的账户的钱数就会增加，增加量即为Pot中的数字。



1. 实验总结
2. 关于博彩细节的处理

为了保证在博彩过程中没有出现多次无效投票攻击的情况，因此我们设定了一定的投票成本（0.015Ether），所有的玩家想要加入博彩，就必须要投入大于等于该钱数的成本；

注意到，博彩的过程中，所有玩家都可以向其中投入资金，但是赢家的挑选则必须要主办方去实现，以保证合约的公信力。

1. 博彩项目尚需补充的细节：

我们想到的更完整的博彩项目，需要一个截止时间，截止时间到后自动由主办方挑选玩家；同时，还可以在博彩中设置更完善的奖励机制，例如投入成本越多的玩家，获得的奖励也会随之增加。

**附录**

附录 **1: css** 代码

LotteryDApp.css

html,

body {

  padding: 0;

  margin: 0;

  font-family: -apple-system, BlinkMacSystemFont, Segoe UI, Roboto, Oxygen,

    Ubuntu, Cantarell, Fira Sans, Droid Sans, Helvetica Neue, sans-serif;

}

a {

  color: inherit;

  text-decoration: none;

}

\* {

  box-sizing: border-box;

}

.main {

  min-height: 100vh;

}

.main h1 {

  font-size: 3em;

  font-weight: bold;

  color: black;

}

.description {

  line-height: 1.5;

  font-size: 1.2em;

}

.footer {

  display: flex;

  flex: 1;

  margin-top: 2rem;

  padding: 2rem 0;

  border-top: 1px solid #eaeaea;

  justify-content: center;

  align-items: center;

}

.footer a {

  display: flex;

  justify-content: center;

  align-items: center;

  flex-grow: 1;

}

附录 **2:js** 代码

import { useState, useEffect } from 'react'

import Head from 'next/head'

import Web3 from 'web3'

import lotteryContract from '../blockchain/lottery'

import styles from '../styles/Home.module.css'

import 'bulma/css/bulma.css'

export default function Home() {

  const [web3, setWeb3] = useState()

  const [address, setAddress] = useState()

  const [lcContract, setLcContract] = useState()

  const [lotteryPot, setLotteryPot] = useState()

  const [lotteryPlayers, setPlayers] = useState([])

  const [lotteryHistory, setLotteryHistory] = useState([])

  const [lotteryId, setLotteryId] = useState()

  const [error, setError] = useState('')

  const [successMsg, setSuccessMsg] = useState('')

  useEffect(() => {

    updateState()

  }, [lcContract])

  const updateState = () => {

    if (lcContract) getPot()

    if (lcContract) getPlayers()

    if (lcContract) getLotteryId()

  }

  const getPot = async () => {

    const pot = await lcContract.methods.getBalance().call()

    setLotteryPot(web3.utils.fromWei(pot, 'ether'))

  }

  const getPlayers = async () => {

    const players = await lcContract.methods.getPlayers().call()

    setPlayers(players)

  }

  const getHistory = async (id) => {

    setLotteryHistory([])

    for (let i = parseInt(id); i > 0; i--) {

      const winnerAddress = await lcContract.methods.lotteryHistory(i).call()

      const historyObj = {}

      historyObj.id = i

      historyObj.address = winnerAddress

      setLotteryHistory(lotteryHistory => [...lotteryHistory, historyObj])

    }

  }

  const getLotteryId = async () => {

    const lotteryId = await lcContract.methods.lotteryId().call()

    setLotteryId(lotteryId)

    await getHistory(lotteryId)

  }

  const enterLotteryHandler = async () => {

    setError('')

    setSuccessMsg('')

    try {

      await lcContract.methods.enter().send({

        from: address,

        value: '15000000000000000',

        gas: 300000,

        gasPrice: null

      })

      updateState()

    } catch(err) {

      setError(err.message)

    }

  }

  const pickWinnerHandler = async () => {

    setError('')

    setSuccessMsg('')

    console.log(`address from pick winner :: ${address}`)

    try {

      await lcContract.methods.pickWinner().send({

        from: address,

        gas: 300000,

        gasPrice: null

      })

    } catch(err) {

      setError(err.message)

    }

  }

  const payWinnerHandler = async () => {

    setError('')

    setSuccessMsg('')

    try {

      await lcContract.methods.payWinner().send({

        from: address,

        gas: 300000,

        gasPrice: null

      })

      console.log(`lottery id :: ${lotteryId}`)

      const winnerAddress = await lcContract.methods.lotteryHistory(lotteryId).call()

      setSuccessMsg(`The winner is ${winnerAddress}`)

      updateState()

    } catch(err) {

      setError(err.message)

    }

  }

  const connectWalletHandler = async () => {

    setError('')

    setSuccessMsg('')

    /\* check if MetaMask is installed \*/

    if (typeof window !== "undefined" && typeof window.ethereum !== "undefined") {

      try {

        /\* request wallet connection \*/

        await window.ethereum.request({ method: "eth\_requestAccounts"})

        /\* create web3 instance & set to state \*/

        const web3 = new Web3(window.ethereum)

        /\* set web3 instance in React state \*/

        setWeb3(web3)

        /\* get list of accounts \*/

        const accounts = await web3.eth.getAccounts()

        /\* set account 1 to React state \*/

        setAddress(accounts[0])

        /\* create local contract copy \*/

        const lc = lotteryContract(web3)

        setLcContract(lc)

        window.ethereum.on('accountsChanged', async () => {

          const accounts = await web3.eth.getAccounts()

          console.log(accounts[0])

          /\* set account 1 to React state \*/

          setAddress(accounts[0])

        })

      } catch(err) {

        setError(err.message)

      }

    } else {

      /\* MetaMask is not installed \*/

      console.log("Please install MetaMask")

    }

  }

  return (

    <div>

      <Head>

        <title>Ether Lottery</title>

        <meta name="description" content="An Ethereum Lottery dApp" />

        <link rel="icon" href="/favicon.ico" />

      </Head>

      <main className={styles.main}>

        <nav className="navbar mt-4 mb-4">

          <div className="container">

            <div className="navbar-brand">

              <h1>Ether Lottery</h1>

            </div>

            <div className="navbar-end">

              <button onClick={connectWalletHandler} className="button is-link">Connect Wallet</button>

            </div>

          </div>

        </nav>

        <div className="container">

          <section className="mt-5">

            <div className="columns">

              <div className="column is-two-thirds">

                <section className="mt-5">

                  <p>Enter the lottery by sending 0.01 Ether</p>

                  <button onClick={enterLotteryHandler} className="button is-link is-large is-light mt-3">Play now</button>

                </section>

                <section className="mt-6">

                  <p><b>Admin only:</b> Pick winner</p>

                  <button onClick={pickWinnerHandler} className="button is-primary is-large is-light mt-3">Pick Winner</button>

                </section>

                <section className="mt-6">

                  <p><b>Admin only:</b> Pay winner</p>

                  <button onClick={payWinnerHandler} className="button is-success is-large is-light mt-3">Pay Winner</button>

                </section>

                <section>

                  <div className="container has-text-danger mt-6">

                    <p>{error}</p>

                  </div>

                </section>

                <section>

                  <div className="container has-text-success mt-6">

                    <p>{successMsg}</p>

                  </div>

                </section>

              </div>

              <div className={`${styles.lotteryinfo} column is-one-third`}>

                <section className="mt-5">

                  <div className="card">

                    <div className="card-content">

                      <div className="content">

                        <h2>Lottery History</h2>

                        {

                          (lotteryHistory && lotteryHistory.length > 0) && lotteryHistory.map(item => {

                            if (lotteryId != item.id) {

                              return <div className="history-entry mt-3" key={item.id}>

                                <div>Lottery #{item.id} winner:</div>

                                <div>

                                  <a href={`https://etherscan.io/address/${item.address}`} target="\_blank">

                                    {item.address}

                                  </a>

                                </div>

                              </div>

                            }

                          })

                        }

                      </div>

                    </div>

                  </div>

                </section>

                <section className="mt-5">

                  <div className="card">

                    <div className="card-content">

                      <div className="content">

                        <h2>Players ({lotteryPlayers.length})</h2>

                        <ul className="ml-0">

                          {

                            (lotteryPlayers && lotteryPlayers.length > 0) && lotteryPlayers.map((player, index) => {

                              return <li key={`${player}-${index}`}>

                                <a href={`https://etherscan.io/address/${player}`} target="\_blank">

                                  {player}

                                </a>

                              </li>

                            })

                          }

                        </ul>

                      </div>

                    </div>

                  </div>

                </section>

                <section className="mt-5">

                  <div className="card">

                    <div className="card-content">

                      <div className="content">

                        <h2>Pot</h2>

                        <p>{lotteryPot} Ether</p>

                      </div>

                    </div>

                  </div>

                </section>

              </div>

            </div>

          </section>

        </div>

      </main>

      <footer className={styles.footer}>

        <p>&copy; 2022 Block Explorer</p>

      </footer>

    </div>

  )

}

const lotteryAbi = [{"inputs":[],"stateMutability":"nonpayable","type":"constructor"},{"inputs":[],"name":"enter","outputs":[],"stateMutability":"payable","type":"function"},{"inputs":[],"name":"getBalance","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"getPlayers","outputs":[{"internalType":"address payable[]","name":"","type":"address[]"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"getRandomNumber","outputs":[{"internalType":"bytes32","name":"requestId","type":"bytes32"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"uint256","name":"lottery","type":"uint256"}],"name":"getWinnerByLottery","outputs":[{"internalType":"address payable","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"uint256","name":"","type":"uint256"}],"name":"lotteryHistory","outputs":[{"internalType":"address payable","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"lotteryId","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"owner","outputs":[{"internalType":"address","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"payWinner","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"pickWinner","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"uint256","name":"","type":"uint256"}],"name":"players","outputs":[{"internalType":"address payable","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"randomResult","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"bytes32","name":"requestId","type":"bytes32"},{"internalType":"uint256","name":"randomness","type":"uint256"}],"name":"rawFulfillRandomness","outputs":[],"stateMutability":"nonpayable","type":"function"}]

const lotteryContract = web3 => {

    return new web3.eth.Contract(

        lotteryAbi,

        ""

    )

}

export default lotteryContract