NEO smart economy

IT大咖说 知识分享平台

NEO smart economy

Blockchain 编程日

2017年8月20日

NEO开发者 陈志同

Blockchain 编程日

1、如何用 C# 编写 NEO 智能合约

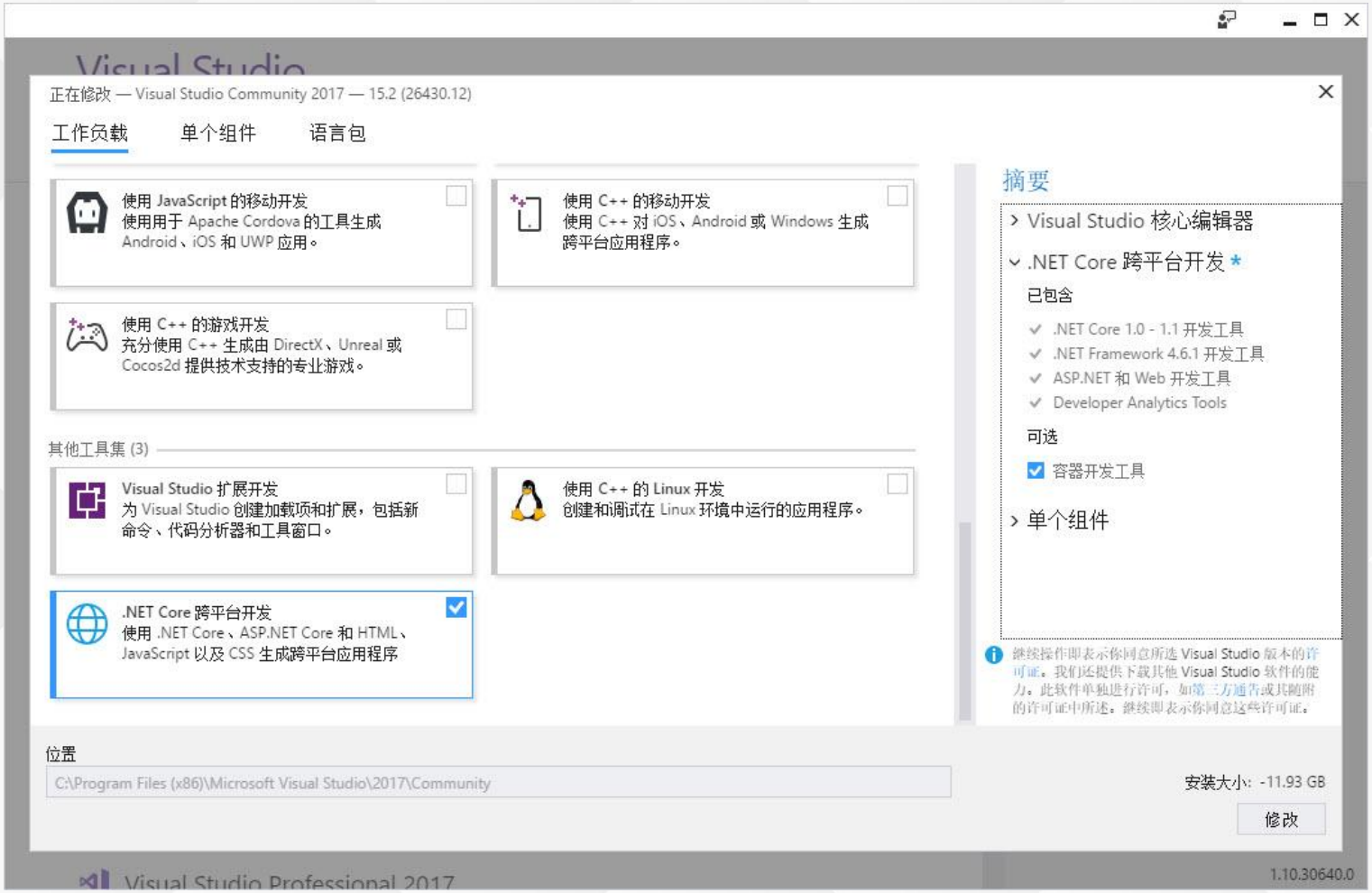
2、在NEO智能合约中使用区块链API

3、在客户端中部署NEO鉴权合约

4、在Azure上一键部署NEO节点

如何用C#编写NEO智能合约

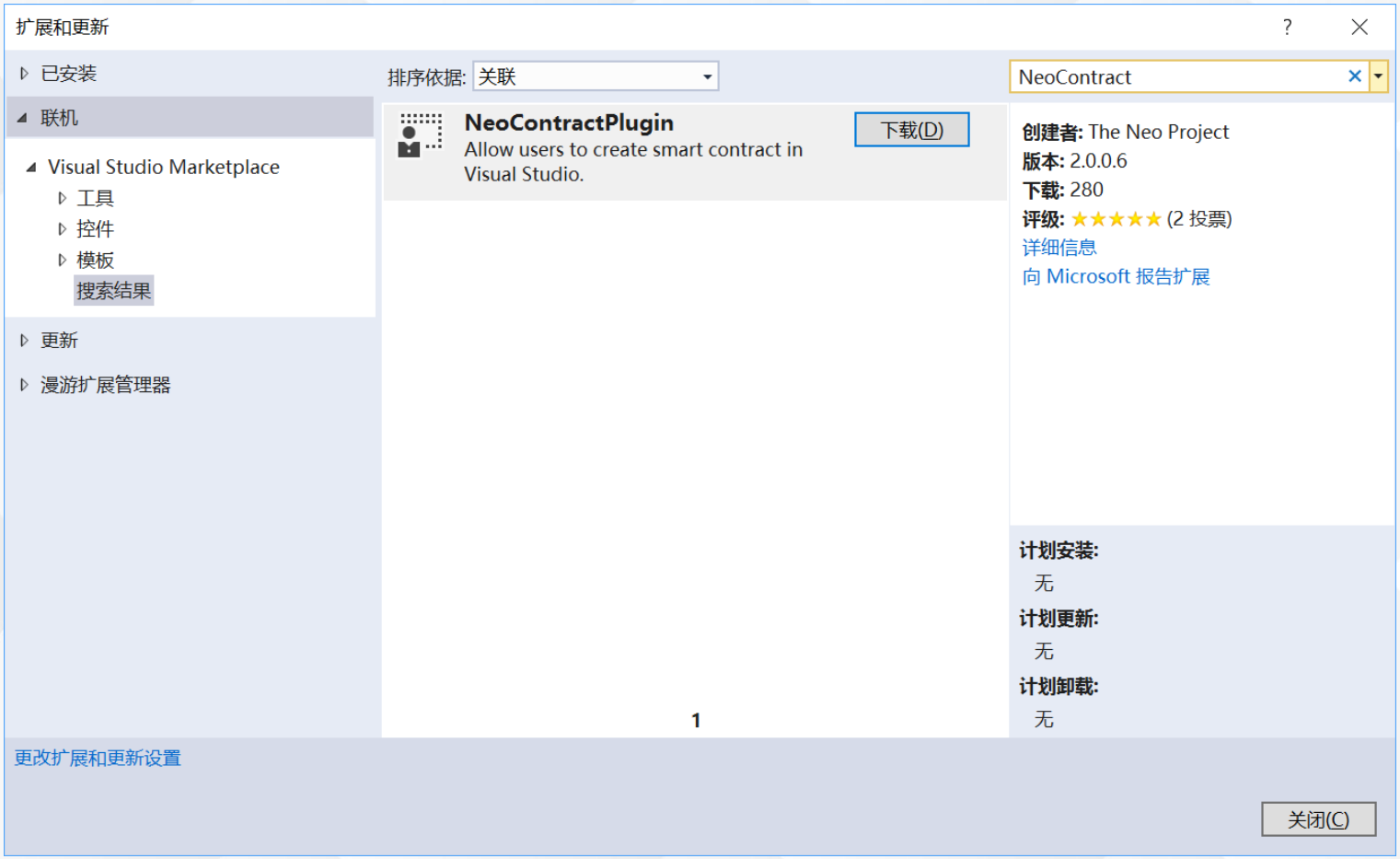
开发工具 → 安装插件 → 安装编译器 → 创建项目 → 编译 → 运行



安装 Visual Studio 2017，在安装时选中.NET Core 跨平台开发

如何用C#编写NEO智能合约

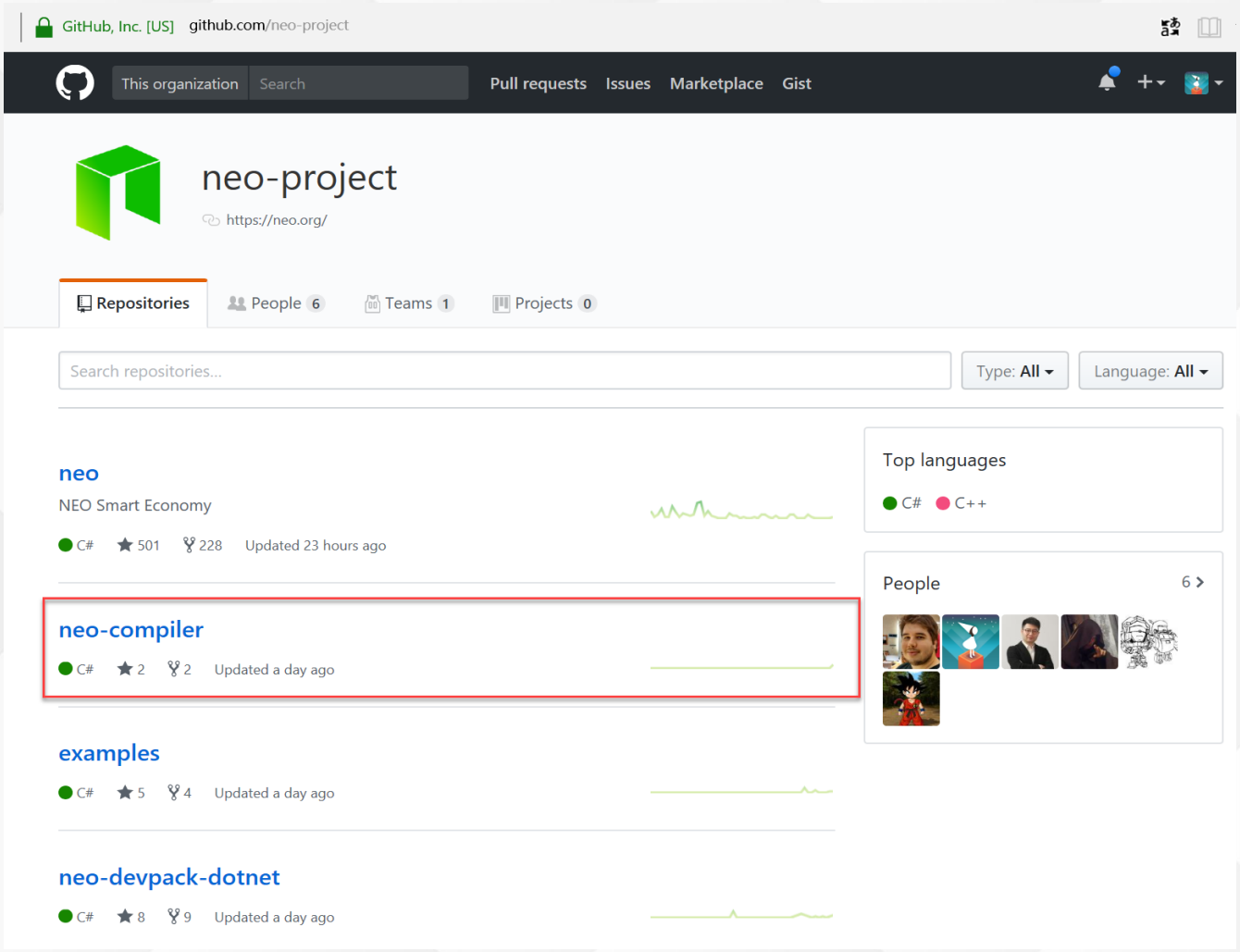
开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



启动 Visual Studio 2017，在"扩展和更新"中安装 NeoContractPlugin

如何用C#编写NEO智能合约

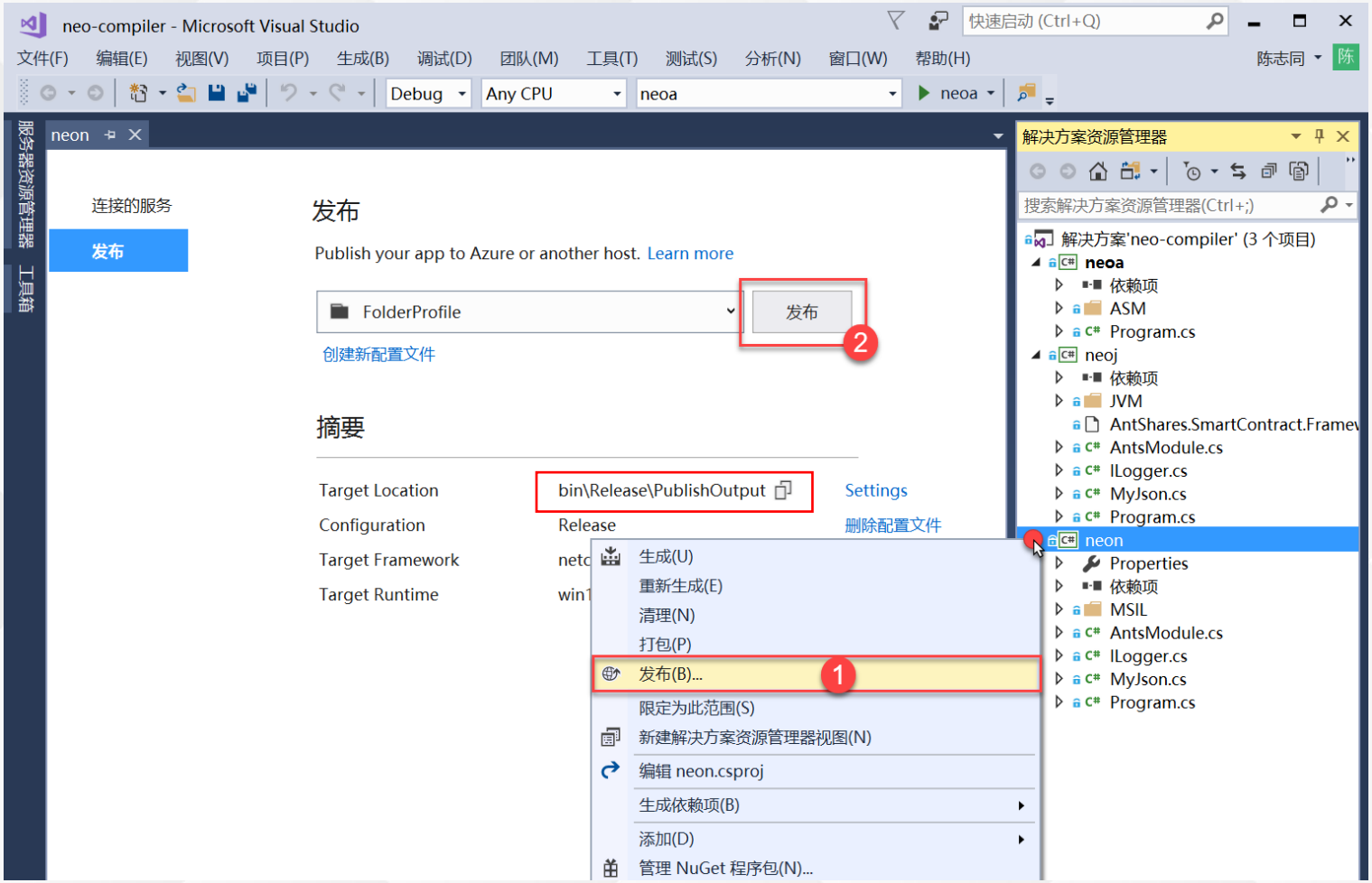
开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



在 GitHub 中下载智能合约编译器 neo-compiler 项目

如何用C#编写NEO智能合约

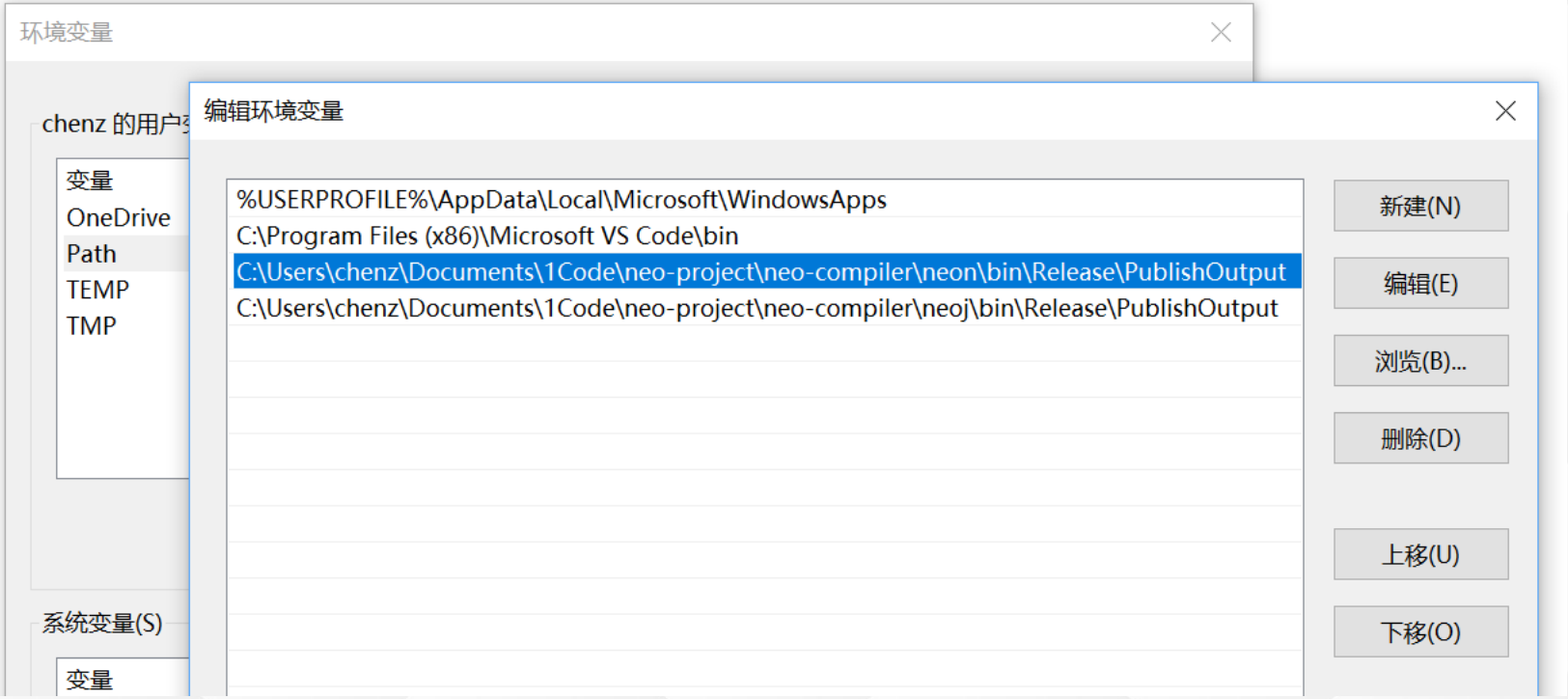
开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



发布 neon 项目

如何用C#编写NEO智能合约

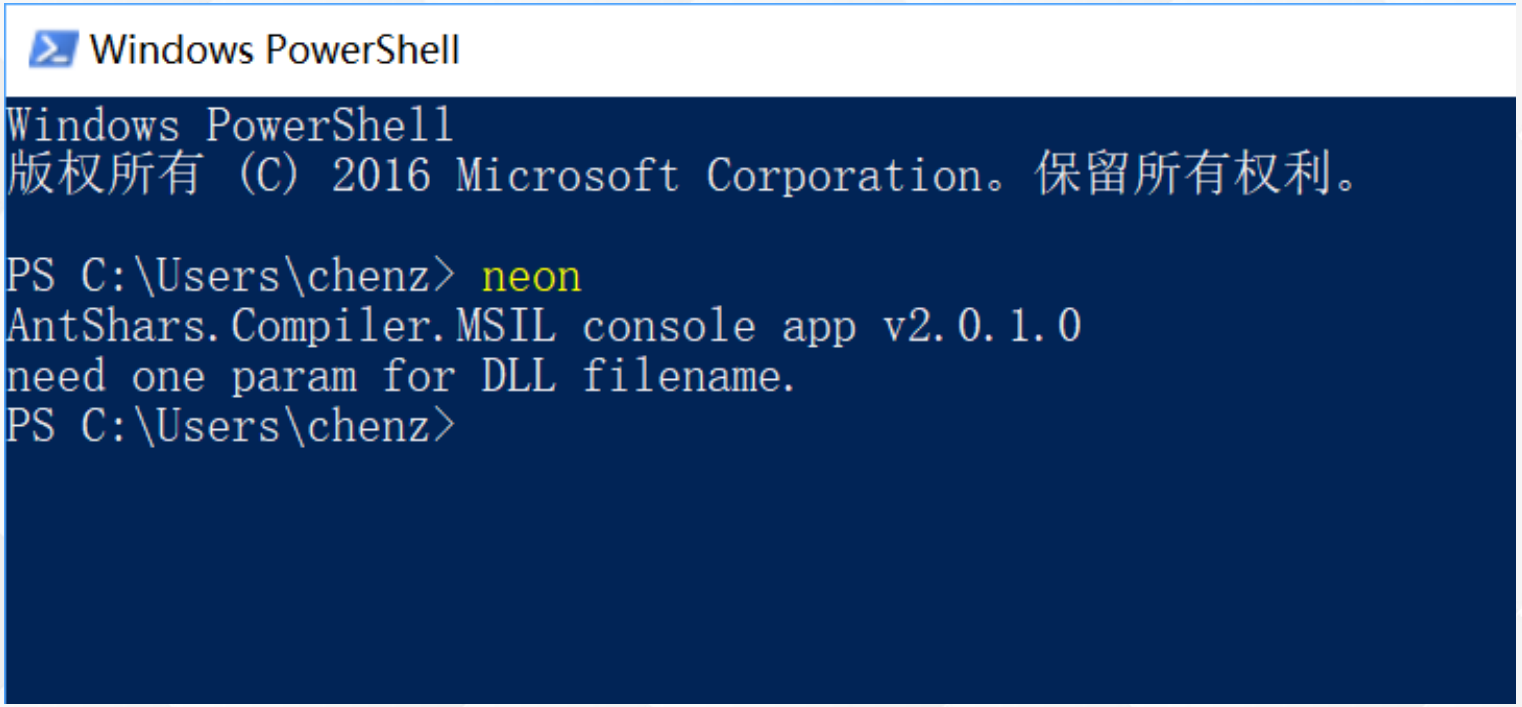
开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



将发布的文件夹添加到环境变量 Path 中

如何用C#编写NEO智能合约

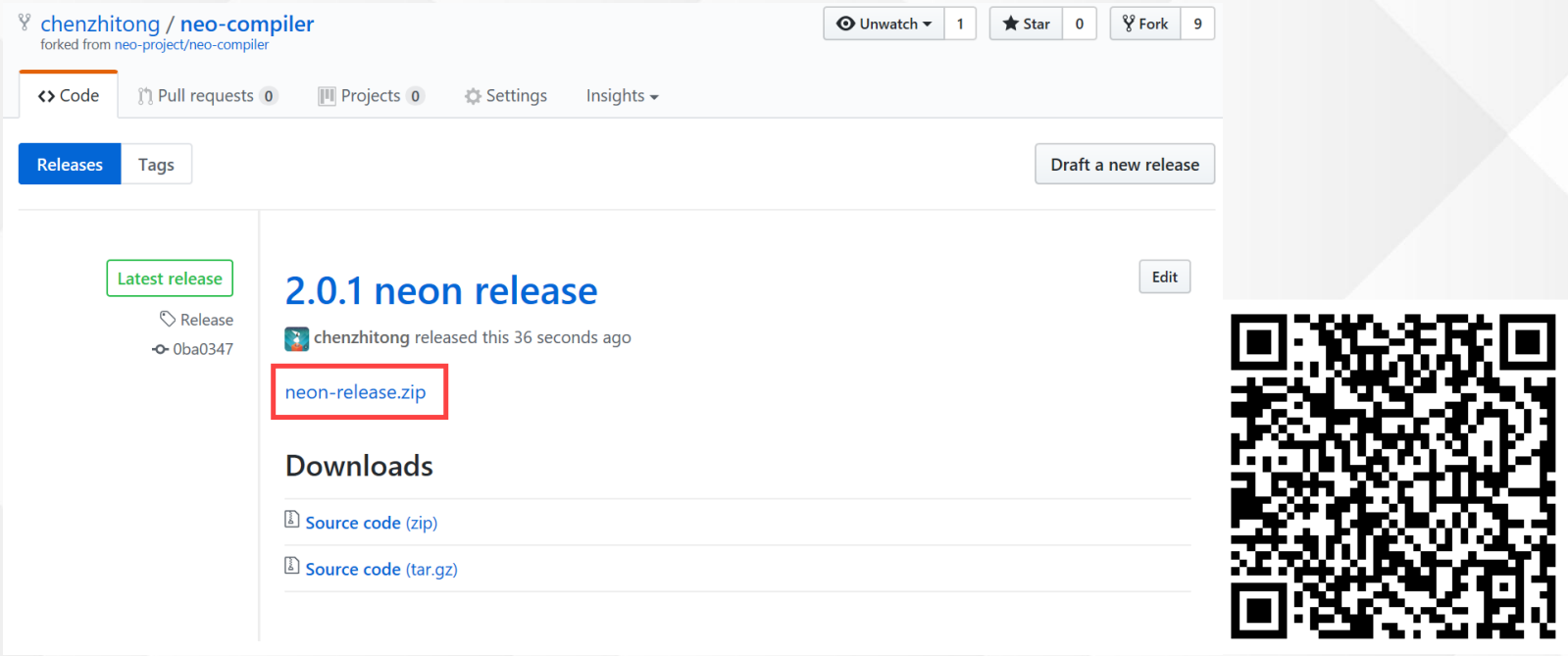
开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



打开 PowerShell，输入neon，显示如图所示表示编译器安装成功

如何用C#编写NEO智能合约

开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行

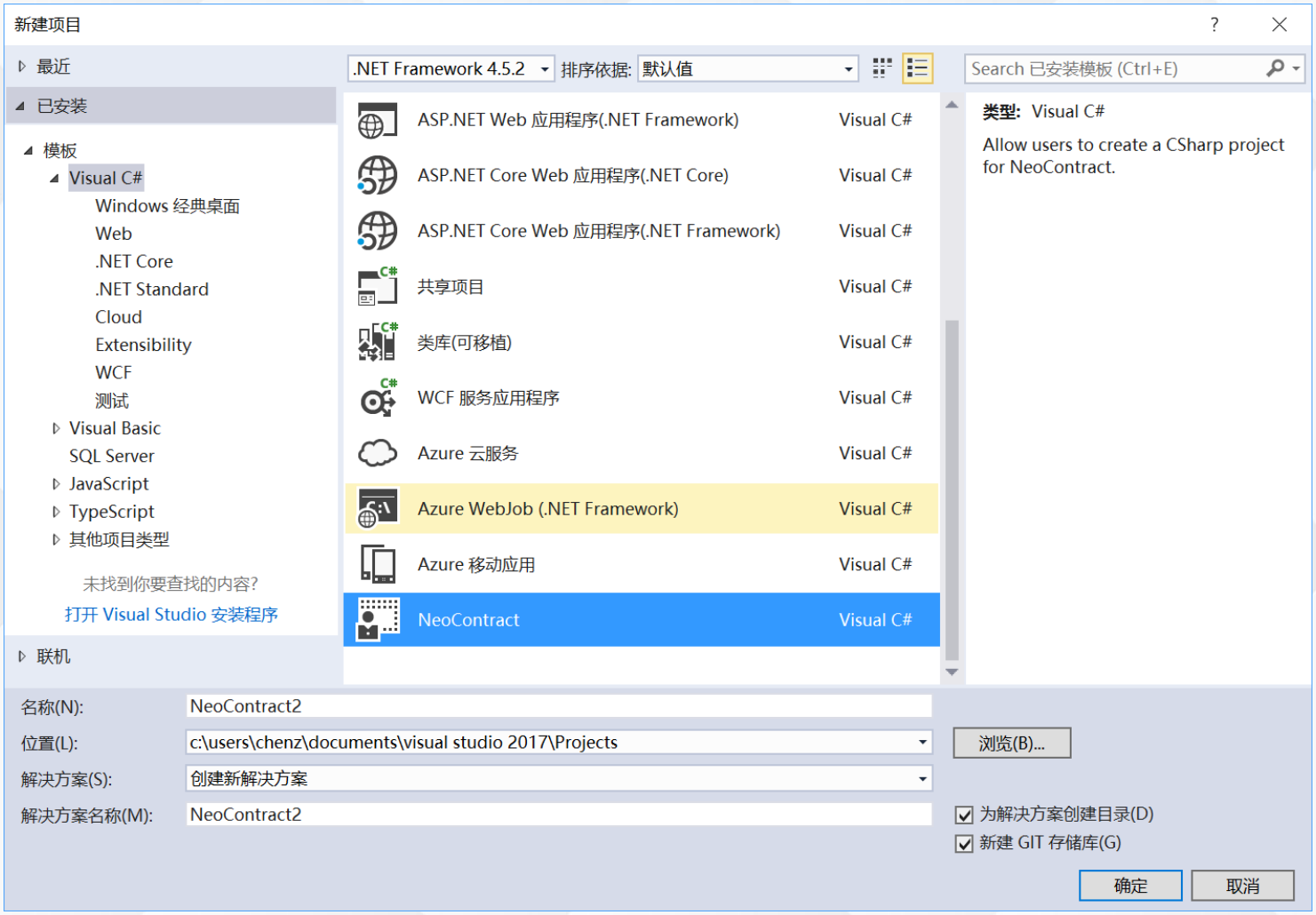


https://github.com/chenzhitong/neo-compiler/releases/

如果 neon 项目还原 NuGet 程序包特别慢的话，可以在这里下载发布后的文件

如何用C#编写NEO智能合约

开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



重新启动 Visual Studio 2017，新建项目，选择 NEO智能合约

如何用C#编写NEO智能合约

开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行

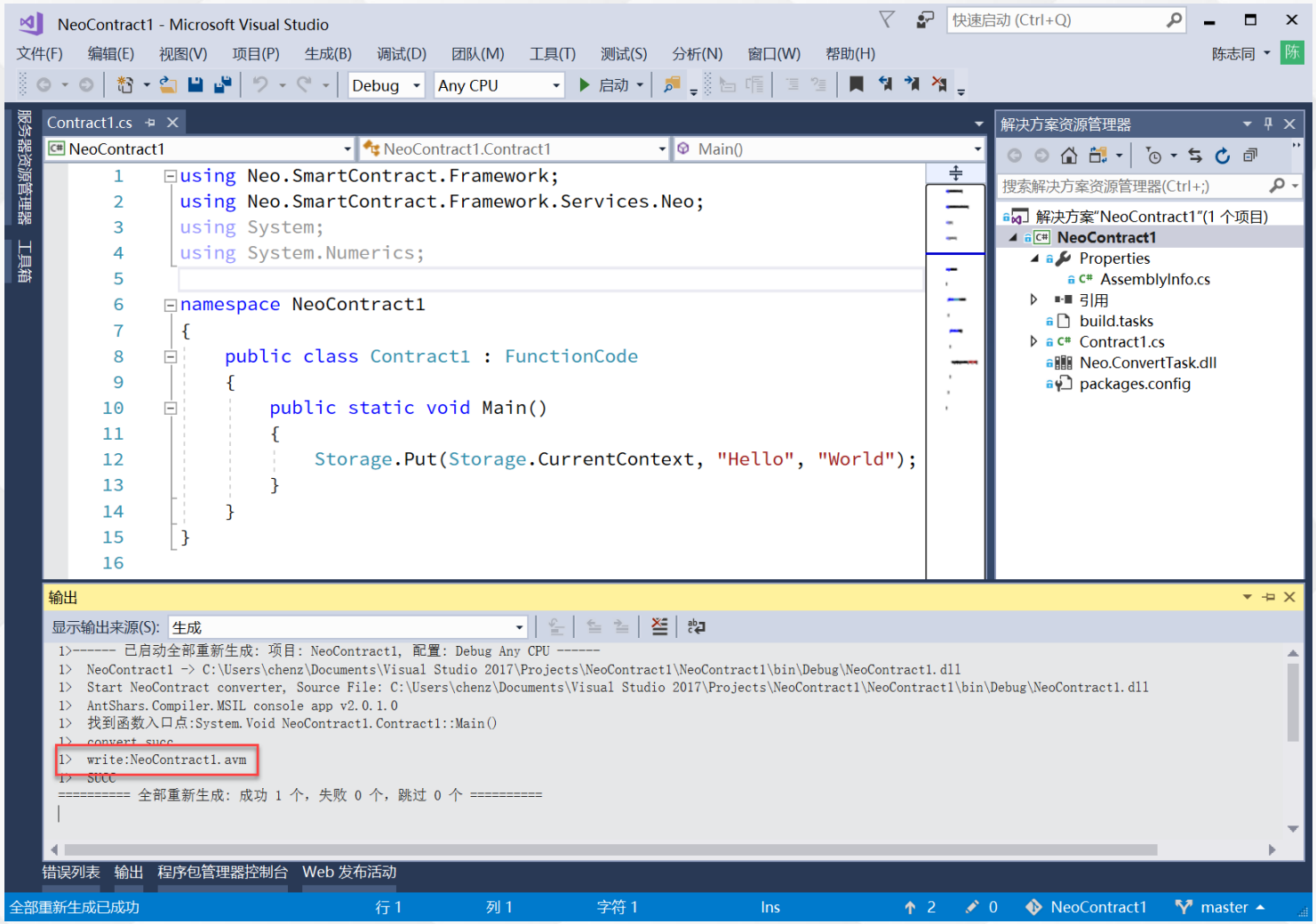
Contract1.cs

NeoContract1 | NeoContract1.Contract1

```
1   using Neo.SmartContract.Framework;
2   using Neo.SmartContract.Framework.Services.Neo;
3   using System;
4   using System.Numerics;
5
6   namespace NeoContract1
7   {
8       public class Contract1 : FunctionCode
9       {
10          public static void Main()
11          {
12              Storage.Put(Storage.CurrentContext, "Hello", "World");
13          }
14      }
15  }
16
```

如何用C#编写NEO智能合约

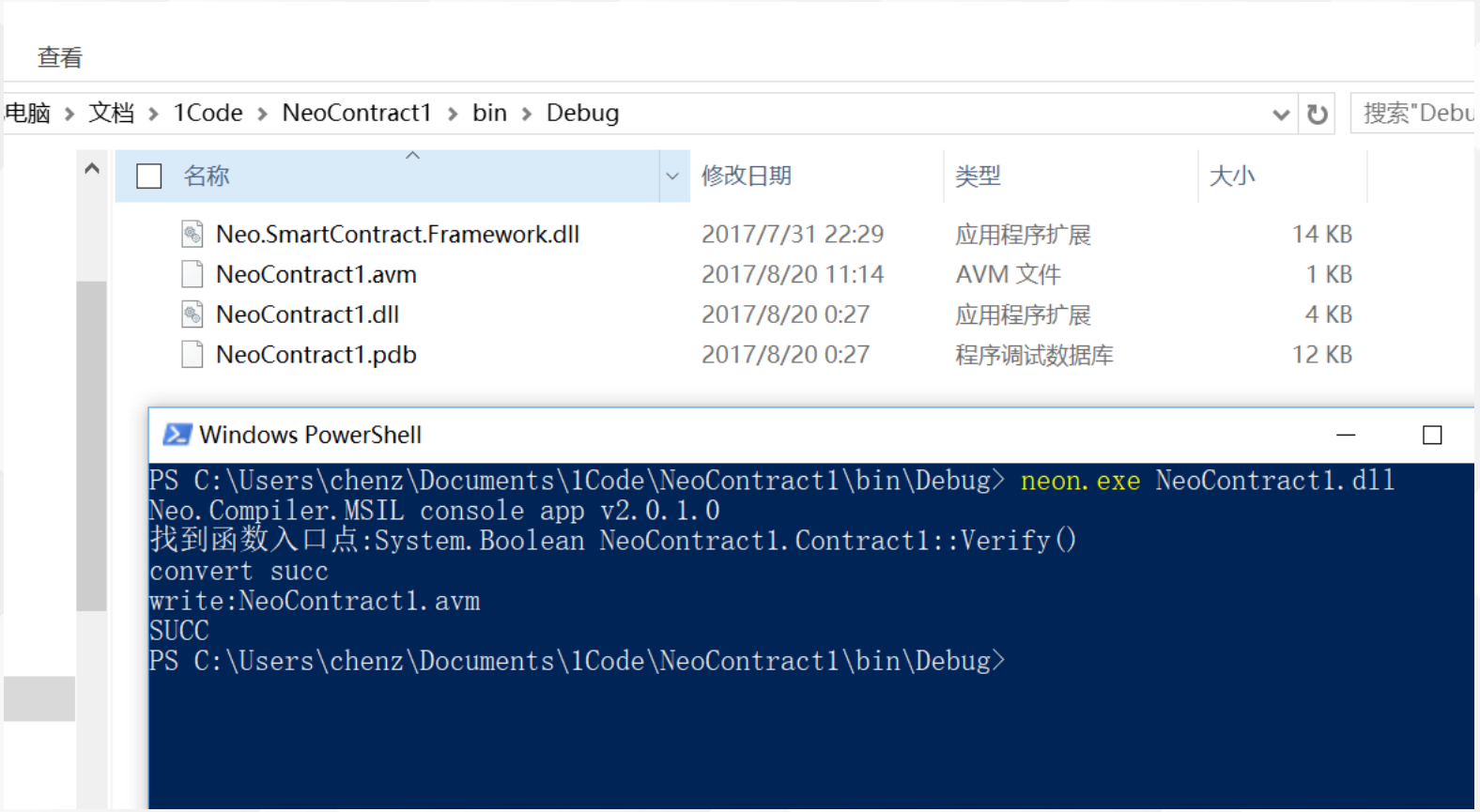
开发工具 → 安装插件 → 安装编译器 → 创建项目 → **编译** → 运行



生成项目，得到 .avm 文件

如何用C#编写NEO智能合约

开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行



也可以用 neon.exe <path> 命令来编译

如何用C#编写NEO智能合约

开发工具 > 安装插件 > 安装编译器 > 创建项目 > 编译 > 运行

智能合约的触发：

触发器 Trigger	鉴权合约 Verification	应用合约 Application
合约代码位置	本地	本地或区块链
继承的基类	VerificationCode	FunctionCode
触发方式	从该合约地址转账会自动触发合约	1、发送交易来触发合约 2、从该合约地址转账会自动触发合约（需额外编程）
可发布到区块链上	否	是
可被其它合约调用	否	是
学习/开发难度	简单	正常

如何用C#编写NEO智能合约

开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行

智能合约的触发：

触发器	鉴权合约	应用合约

Contract1.cs

NeoContract1 | NeoContract1.Contract1

```
1   using Neo.SmartContract.Framework;
2   using Neo.SmartContract.Framework.Services.Neo;
3   using System;
4   using System.Numerics;
5
6   namespace NeoContract1
7   {
8       public class Contract1 : FunctionCode
9       {
10          public static void Main()
11          {
12              Storage.Put(Storage.CurrentContext, "Hello", "World");
13          }
14      }
15  }
16
```

学习/开发难度	简单	正常

如何用C#编写NEO智能合约

开发工具 > 安装插件 > 安装编译器 > 创建项目 > 编译 > 运行

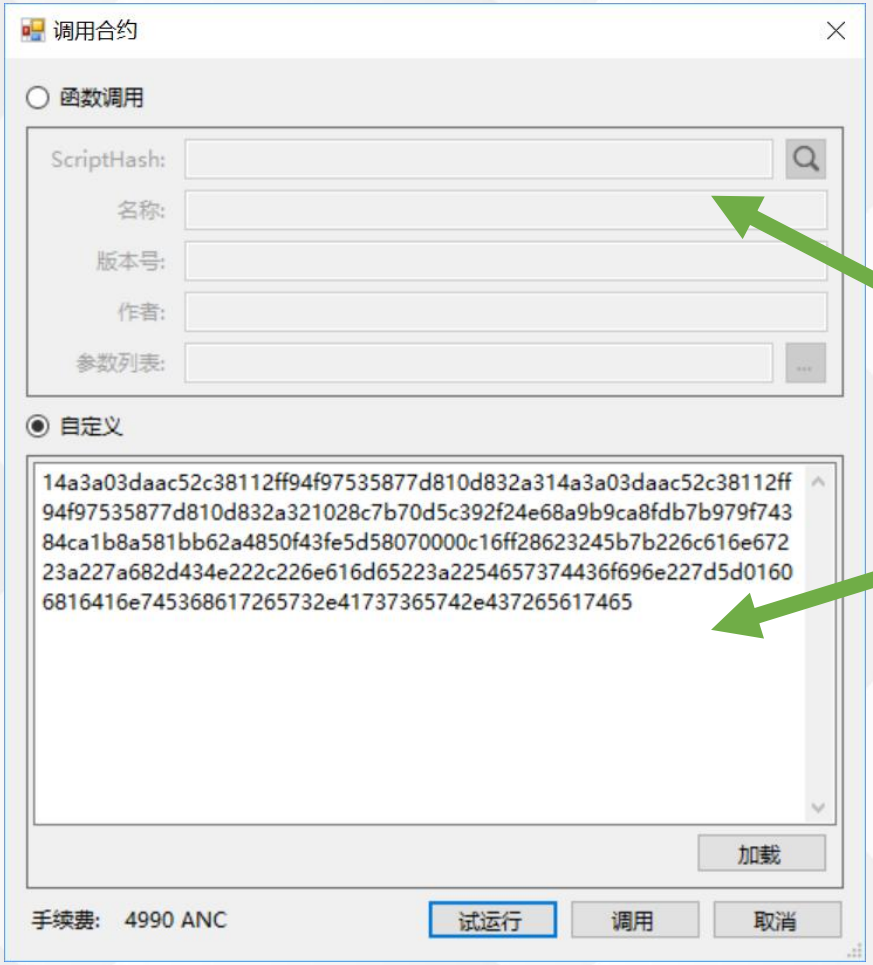
智能合约的触发：

触发器 Trigger	鉴权合约 Verification	应用合约 Application
合约代码位置	本地	本地或区块链
继承的基类	VerificationCode	FunctionCode
触发方式	从该合约地址转账会自动触发合约	1、发送交易来触发合约 2、从该合约地址转账会自动触发合约（需额外编程）
可发布到区块链上	否	是
可被其它合约调用	否	是
学习/开发难度	简单	正常

如何用C#编写NEO智能合约

开发工具 | 安装插件 | 安装编译器 | 创建项目 | 编译 | 运行

应用合约（Application）的一种常见的触发方式



调用区块链上的智能合约

调用本地的智能合约

Blockchain 编程日

1、如何用 C# 编写 NEO 智能合约

2、在NEO智能合约中使用区块链API

3、在客户端中部署NEO鉴权合约

4、在Azure上一键部署NEO节点

在NEO智能合约中使用区块链API

	类	说明
	Account	表示账户的类，提供了查询余额的方法
	Asset	用来表示资产的数据结构
	Block	表示区块的类，提供了查询区块中交易的方法
	Blockchain	该类提供了访问区块链数据的一系列方法
	Contract	表示合约的类
	Header	用来表示区块头的数据结构
	Runtime	new 提供智能合约运行时的一些方法
	Storage	提供了持久化存储区的插入、查询、删除的方法
	StorageContex	new 用来表示私有存储区存储上下文的类
	Transaction	用来表示交易的基类
	TransactionAttribute	用来表示交易特性的数据结构
	TransactionInput	用来表示交易输入的数据结构
	TransactionOutput	用来表示交易输出的数据结构
	Validator	new 提供共识节点的一些方法

http://docs.neo.org/zh-cn/sc/fw/dotnet/neo.html

在NEO智能合约中使用区块链API

Account.GetBalance 方法 (byte[])

通过资产ID获得该账户中这种资产的余额。

命名空间：Neo.SmartContract.Framework.Services.Neo

程序集：Neo.SmartContract.Framework

语法

```
public extern long GetBalance(byte[] asset_id)
```

参数：资产ID，即注册资产时的 RegistTransaction 的交易ID，32字节的byte数组。

返回值：账户中该资产余额，长整形，等于实际金额 × 10^8。

示例

```
public class Contract1 : FunctionCode
{
    public static void Main()
    {
        byte[] scriptHash = { 36, 23, 241, 177, 228, 54, 109, 223, 27, 237, 139, 54, 20
7, 38, 132, 101, 172, 3, 10, 73 };
        Account account = Blockchain.GetAccount(scriptHash);
        //以NEO资产为例
        byte[] asset = { 197, 111, 51, 252, 110, 207, 205, 12, 34, 92, 74, 179, 86, 25
4, 229, 147, 144, 175, 133, 96, 190, 14, 147, 15, 174, 190, 116, 166, 218, 255, 124, 15
5 };
        long balance = account.GetBalance(asset);
    }
}
```

语法

```
public extern long GetBalance(byte[] asset_id)
```

参数：资产ID，即注册资产时的 RegistTransaction 的交易ID，32字节的byte数组。

返回值：账户中该资产余额，长整形，等于实际金额 × 10^8。

示例

```
public class Contract1 : FunctionCode
{
    public static void Main()
    {
        byte[] scriptHash = { 36, 23, 241, 177, 228, 54, 109, 223, 27, 237, 139, 54, 20
7, 38, 132, 101, 172, 3, 10, 73 };
        Account account = Blockchain.GetAccount(scriptHash);
        //以NEO资产为例
        byte[] asset = { 197, 111, 51, 252, 110, 207, 205, 12, 34, 92, 74, 179, 86, 25
4, 229, 147, 144, 175, 133, 96, 190, 14, 147, 15, 174, 190, 116, 166, 218, 255, 124, 15
5 };
        long balance = account.GetBalance(asset);
    }
}
```

在NEO智能合约中使用区块链API

现场演示

http://docs.neo.org/zh-cn/sc/fw/dotnet/neo.html

在NEO智能合约中使用区块链API

锁仓合约示例

Program.cs | Contract1.cs

NeoContract1 | NeoContract1.Lock | Verify(byte[] sig

```
1   using Neo.SmartContract.Framework;
2   using Neo.SmartContract.Framework.Services.Neo;
3   using System;
4   using System.Numerics;
5
6   namespace NeoContract1
7   {
8       public class Lock : VerificationCode
9       {
10          public static bool Verify(byte[] signature)
11          {
12              Header header = Blockchain.GetHeader(Blockchain.GetHeight());
13              if (header.Timestamp < 1504195200) // 2017-9-1 0:0:0
14                  return false;
15              // 公钥0285eab65f4a0126e4b85b4e5d8b7e303aff7efb360d595f2e3189bb90487ad5aa
16              return VerifySignature(new byte[] { 2, 133, 234, 182, 95, 74, 1, 38,
17                  228, 184, 91, 78, 93, 139, 126, 48, 58, 255, 126, 251, 54, 13, 89,
18                      95, 46, 49, 137, 187, 144, 72, 122, 213, 170 }, signature);
19          }
20      }
21  }
22
```

参考：http://docs.neo.org/zh-cn/sc/tutorial/Lock2.html

Blockchain 编程日

1、如何用 C# 编写 NEO 智能合约

2、在NEO智能合约中使用区块链API

3、在客户端中部署NEO鉴权合约

4、在Azure上一键部署NEO节点

在客户端中部署NEO鉴权合约

编写合约 | 获得合约脚本 | 创建钱包 | 创建合约地址 | 测试

Program.cs | Contract1.cs

NeoContract1 | NeoContract1.Contract1 | Verify()

```
1   using Neo.SmartContract.Framework;
2   using Neo.SmartContract.Framework.Services.Neo;
3   using System;
4   using System.Numerics;
5
6   namespace NeoContract1
7   {
8       public class Contract1 : VerificationCode
9       {
10          public static bool Verify()
11          {
12              return true;
13          }
14      }
15  }
16
```

在客户端中部署NEO鉴权合约

编写合约 | 获得合约脚本 | 创建钱包 | 创建合约地址 | 测试

Program.cs | Contract1.cs

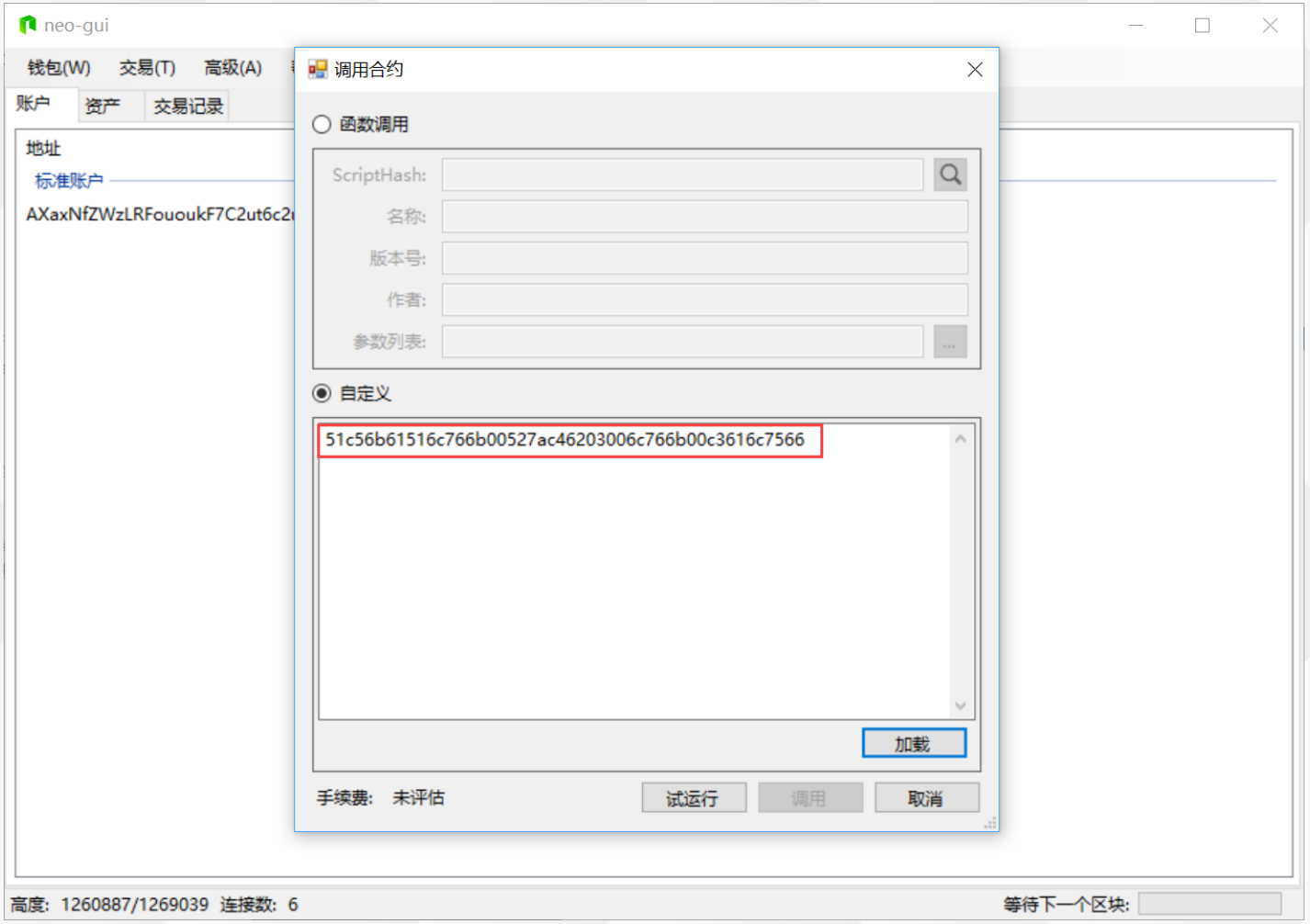
ConsoleApp1 | ConsoleApp1.Program | Main(stri

```
1  using System;
2  using System.IO;
3  using System.Text;
4
5  namespace ConsoleApp1
6  {
7      class Program
8      {
9          static void Main(string[] args)
10         {
11             byte[] bytes = File.ReadAllBytes("Test.avm");
12             string str = Encoding.Default.GetString(bytes);
13             Console.WriteLine(str);
14             Console.ReadLine();
15         }
16     }
17 }
```

自己编写程序，获取到合约的脚本

在客户端中部署NEO鉴权合约

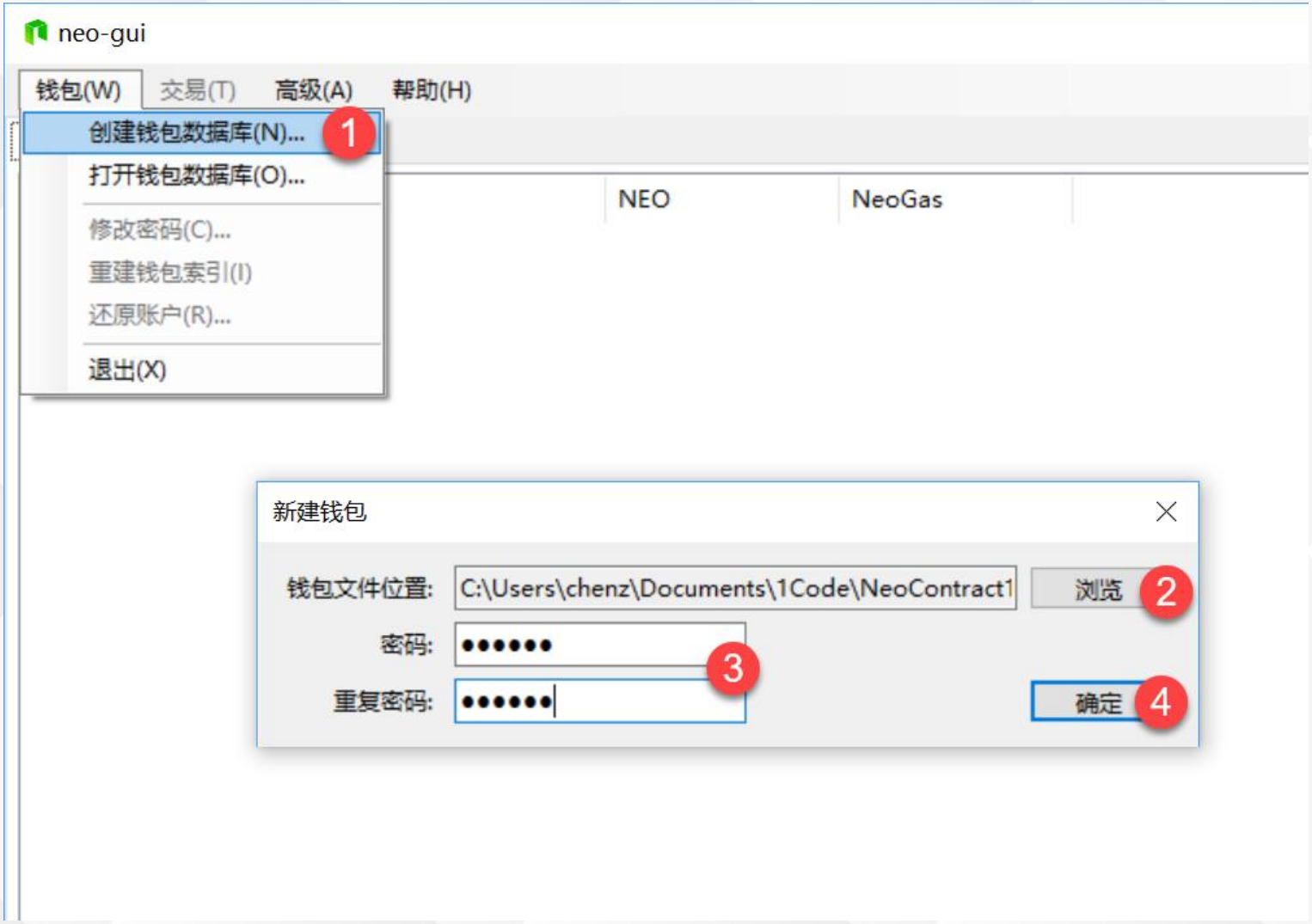
编写合约 | 获得合约脚本 | 创建钱包 | 创建合约地址 | 测试



不编写程序，用客户端的相关功能也能获取到合约脚本

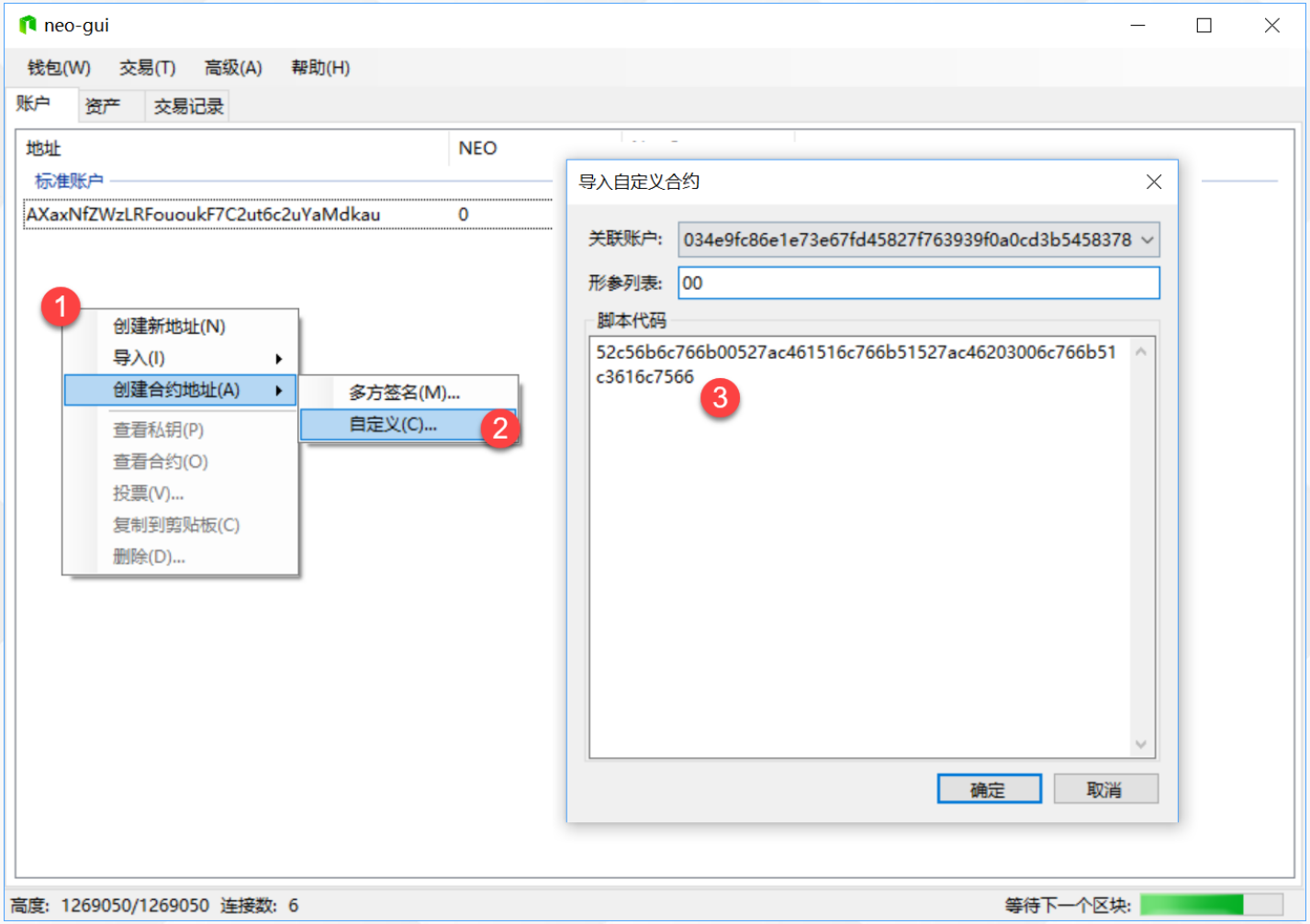
在客户端中部署NEO鉴权合约

编写合约 | 获得合约脚本 | 创建钱包 | 创建合约地址 | 测试



在客户端中部署NEO鉴权合约

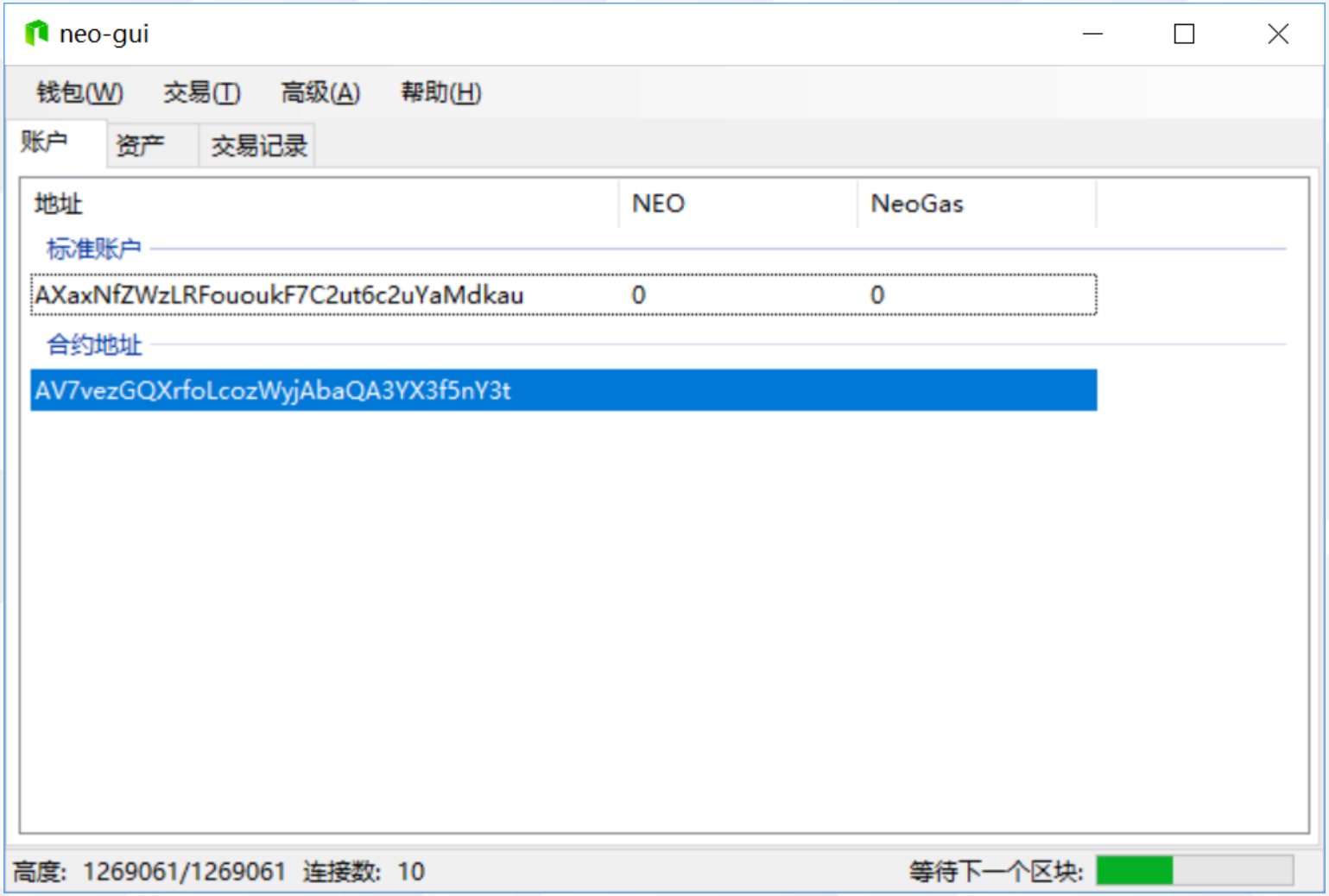
编写合约 | 获得合约脚本 | 创建钱包 | 创建合约地址 | 测试



创建自定义合约，选择关联账户，输入形参列表，和脚本代码

在客户端中部署NEO鉴权合约

编写合约 | 获得合约脚本 | 创建钱包 | 创建合约地址 | 测试



在客户端中部署NEO鉴权合约

编写合约 → 获得合约脚本 → 创建钱包 → 创建合约地址 → 测试

AV7vezGQXrfoLcozWyjAbaQA3YX3f5nY3t
智能合约地址

合约执行通过

return true

转账成功

合约执行失败

return false 或 出现异常

转账失败

AXaxNfZWzLRFououkF7C2ut6c2uYaMdkau
其它地址

Blockchain 编程日

1、如何用 C# 编写 NEO 智能合约

2、在NEO智能合约中使用区块链API

3、在客户端中部署NEO鉴权合约

4、在Azure上一键部署NEO节点

在Azure上一键部署NEO节点

申请账号 | 搜索镜像 | 填写信息 | 开始创建 | 启动

登录 Azure （中国）网站：www.azure.cn 创建一个 Azure 账号。

注：该账号为世纪互联运营的 Azure（中国）账号，与 Azure（全球）账号及微软账号不通用。

新用户在 Azure（中国）中可以申请 1 元试用活动，您只需要缴纳 1 元人民币，就可以获得 1,500 元 Azure 服务使用额度，有效期一个月。

详情请点击 1 元试用订阅详情。

在Azure上一键部署NEO节点

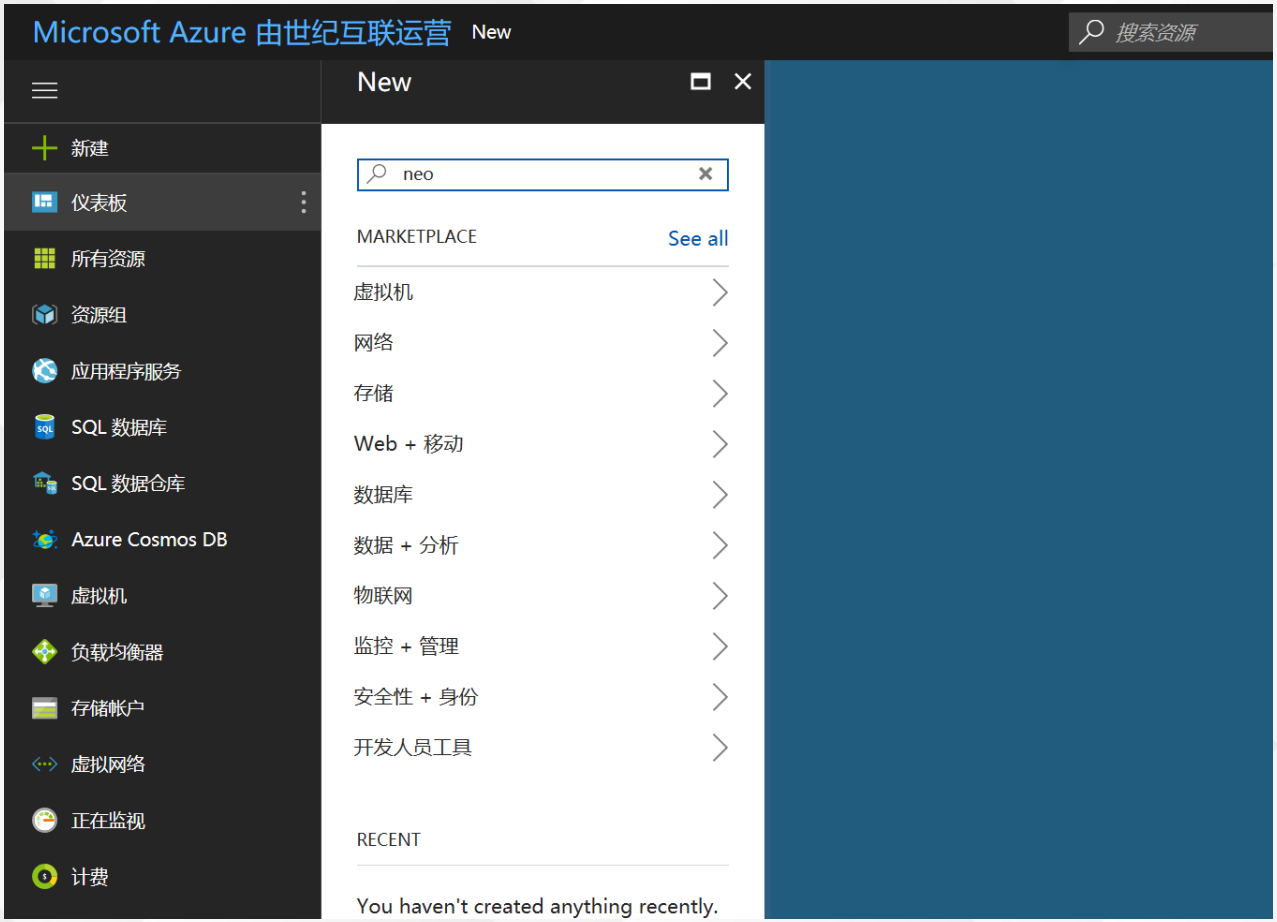
申请账号 | 搜索镜像 | 填写信息 | 开始创建 | 启动

创建好账户后，打开 Azure 镜像市场 在搜索中搜索 **NEO** 即可找到 NEO 的 Azure 镜像。

在Azure上一键部署NEO节点

申请账号 | 搜索镜像 | 填写信息 | 开始创建 | 启动

不久后即可在 MAEKETPLACE 中搜索到 NEO 的 Azure 镜像。



在Azure上一键部署NEO节点

申请账号 | 搜索镜像 | 填写信息 | 开始创建 | 启动

点击进去可以查看使用详情。点击 立即部署 会跳转到 创建 Azure 虚拟机 界面，在该页面中可以配置虚拟机的基础信息。

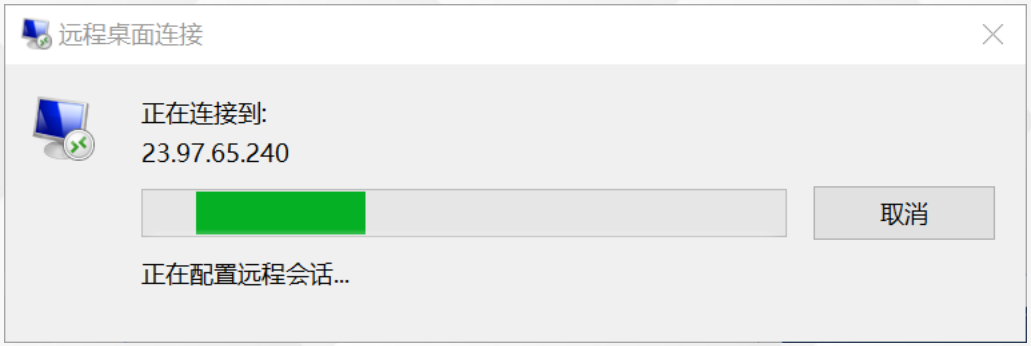
在Azure上一键部署NEO节点

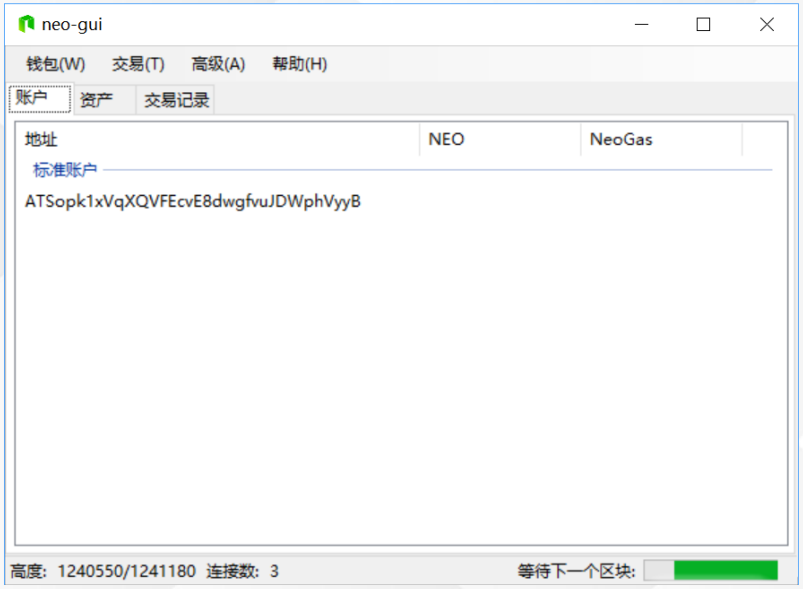
申请账号 > 搜索镜像 > 填写信息 > 开始创建 > 启动

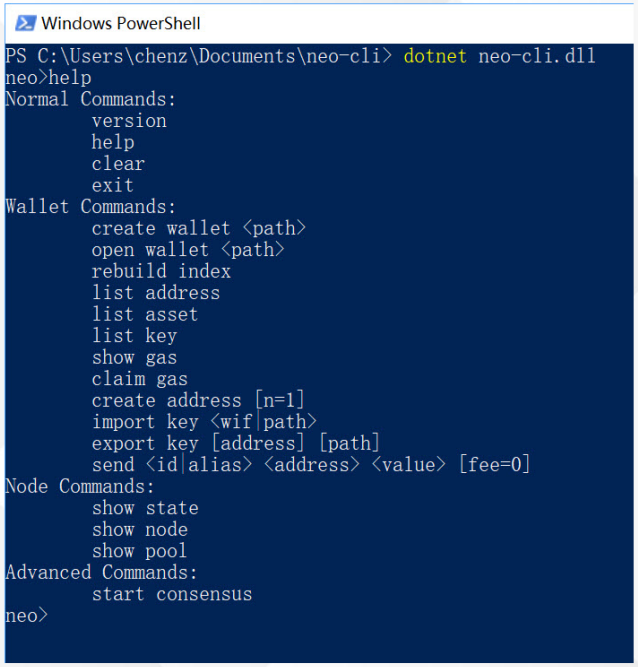
点击页面最下方的立即部署即可开始部署镜像。

在Azure上一键部署NEO节点

申请账号 → 搜索镜像 → 填写信息 → 开始创建 → 启动







在Azure上一键部署NEO节点

部署了NEO节点可以干什么？

1、学习NEO区块链
neo-cli，neo-gui，API

2、开发NEO生态项目
轻钱包的服务器，区块链浏览器，NEO智能合约商店

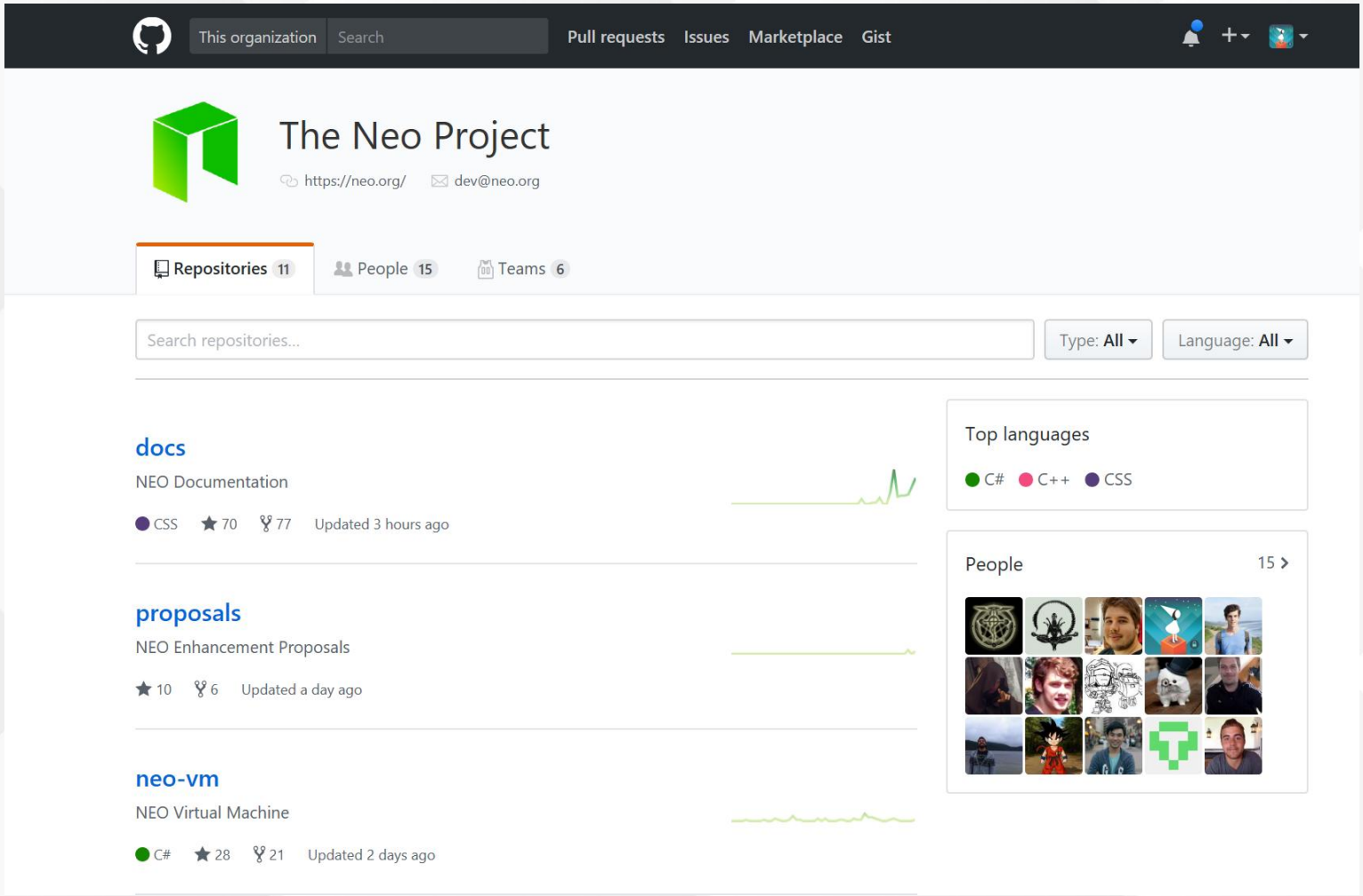
3、开发自己的基于区块链的项目

4、搭建私有链或联盟链

NEO相关资源

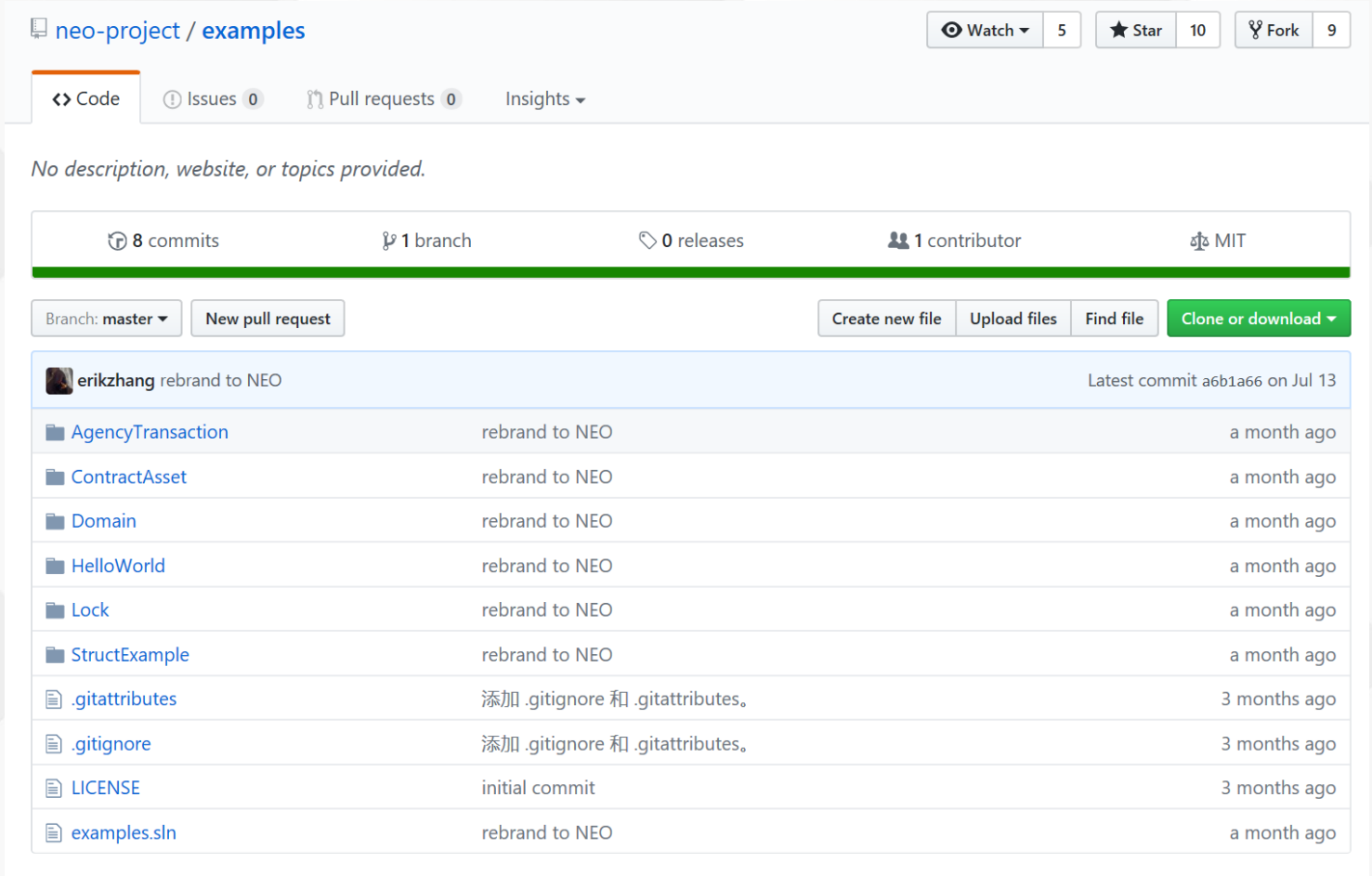
官方网站	neo.org

NEO相关资源



GitHub github.com/neo-project

NEO相关资源



智能合约示例 github.com/neo-project/examples

NEO相关资源

tanZiWen / neo_ico_template

Watch 3 | Star 3 | Fork 2

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights

Branch: master ▾ neo_ico_template / NEO_ICO_Template / ICO_Template.cs

Find file | Copy path

元谈 fix error name — b6c9ed5 2 hours ago

3 contributors

Executable File | 260 lines (246 sloc) | 10.9 KB — Raw | Blame | History

```
using Neo.SmartContract.Framework;
using Neo.SmartContract.Framework.Services.Neo;
using Neo.SmartContract.Framework.Services.System;
using System;
using System.ComponentModel;
using System.Numerics;

namespace ICO_Template
{
    public class ICO_Template : FunctionCode
    {
        //Token Settings
        public static string Name() => "name of the token";
        public static string Symbol() => "SymbolOfTheToken";
        public static readonly byte[] Owner = { 2, 133, 234, 182, 95, 74, 1, 38, 228, 184, 91, 78, 93, 139, 126, 48, 58, 255, 126, 251, 54,
        public static byte Decimals() => 8;
        private const ulong factor = 100000000; //decided by Decimals()

        //ICO Settings
        private static readonly byte[] neo_asset_id = { 197, 111, 51, 252, 110, 207, 205, 12, 34, 92, 74, 179, 86, 254, 229, 147, 144, 175,
        private const ulong basic_rate = 1000 * factor;
        private const ulong total_ico_usd = 12000000;
        private const ulong total_pre_usd = 3000000;
        private const ulong neo_decimals = 100000000;
        private const ulong neo_to_usd = 50;
```

ICO模板 智能合约示例 — github.com/tanZiWen/neo_ico_template

NEO相关资源

官网 | 中文 | English | Español | 日本語 | 한국어 | Deutsche | Nederlandse

中文 / 智能合约 / 如何开始（C#）

Enter here to filter...

- 白皮书
- 开始学习
- + NEO 节点
- - 智能合约
  - 介绍
  - 如何开始（C#）
  - 如何开始（Java）
  - + 教程
  - 测试
  - 白皮书
  - + API 参考
  - + 框架

如何用 C# 编写 NEO 智能合约

目前 NEO 智能合约推荐使用 C# 语言来开发（此外还支持 Java、Kotlin、Go、C/C++、Python、JavaScript 等编程语言）

此部分包含简短的教程，可指导你配置 NEO 智能合约的 C# 开发环境，并使你了解如何创建智能合约项目，以及如何编译。

NOTE
目前 NEO 的所有项目已经升级到了 Visual Studio 2017 版本，如果你电脑中安装的是 Visual Studio 2015，请升级。

Improve this Doc

IN THIS ARTICLE
- 开发工具
- 创建项目
- 编译项目

开发工具

1. Visual Studio 2017

如果你的计算机中已经安装过 Visual Studio 2017，并且在安装时勾选了 `.NET Core 跨平台开发` 可跳过本小节。

下载及安装方法：

Visual Studio 下载地址

安装过程很简单，直接按照提示一步一步操作即可，需要注意的是在安装时需要勾选 `.NET Core 跨平台开发` ，安装大概需要十几分钟或几十分钟。

技术文档 docs.neo.org

THANKS!

NEO开发者陈志同

因为NEO更新迭代速度非常快，本演示文稿仅保障截止到演讲当天的正确性