**2024《区块链技术与应用》**

**（微众银行）**

**实验2：开发Solidity智能合约**

1. **实验背景与目标**

本实验旨在通过Solidity语言实现一个符合ERC20标准的智能合约，并在FISCO BCOS平台上进行部署和调用，使参与者能够深入理解智能合约的开发流程、ERC20标准以及区块链平台的部署机制。

### 实验目标

1. **掌握Solidity开发环境**：通过搭建Solidity开发环境，参与者将能够熟练使用相关工具进行智能合约的开发和测试。
2. **理解ERC20标准**：通过实现一个ERC20标准的智能合约，参与者将能够理解ERC20代币的工作原理和特性。
3. **智能合约开发实践**：通过编写和测试ERC20Test合约，参与者将获得实际的智能合约开发经验。
4. **区块链平台部署经验**：通过在FISCO BCOS平台上部署和调用ERC20Test合约，参与者将了解智能合约在区块链平台上的部署流程和调用方法。

### 实验步骤概述

1. **准备Solidity开发环境**：使用Remix等的成熟Solidity开发工具
2. **实现ERC20标准的合约ERC20Test**：
   * 学习ERC20标准，理解其功能和接口。
   * 使用Solidity编写ERC20Test合约，实现代币的发行、转移、余额查询等功能。
3. **在FISCO BCOS上部署调用合约ERC20Test**：
   * 学习FISCO BCOS平台的基本操作和合约部署流程。
   * 将ERC20Test合约部署到FISCO BCOS平台上，并进行调用测试，验证合约功能的正确性。

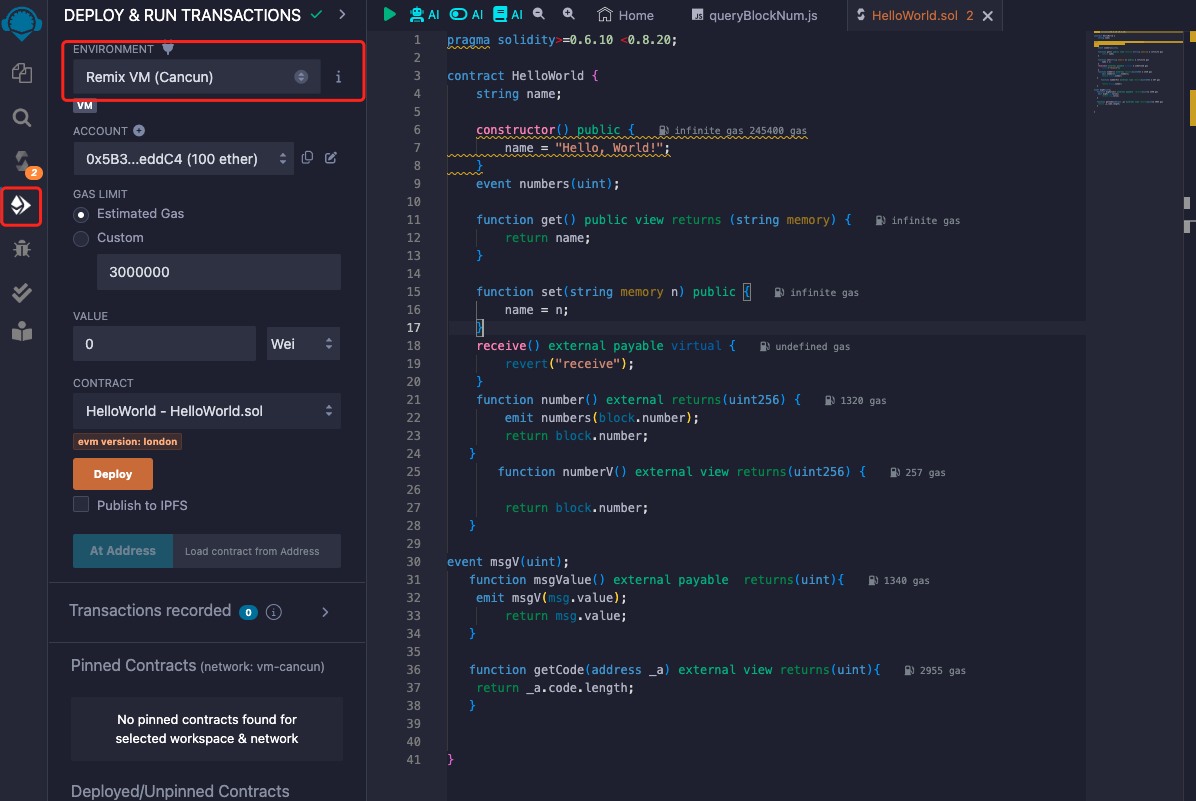
通过本实验，参与者不仅能够获得Solidity智能合约的开发和部署经验，还能够深入理解ERC20代币标准和区块链平台的运作机制，为未来的区块链应用开发打下坚实的基础。

1. **准备Solidity开发环境**

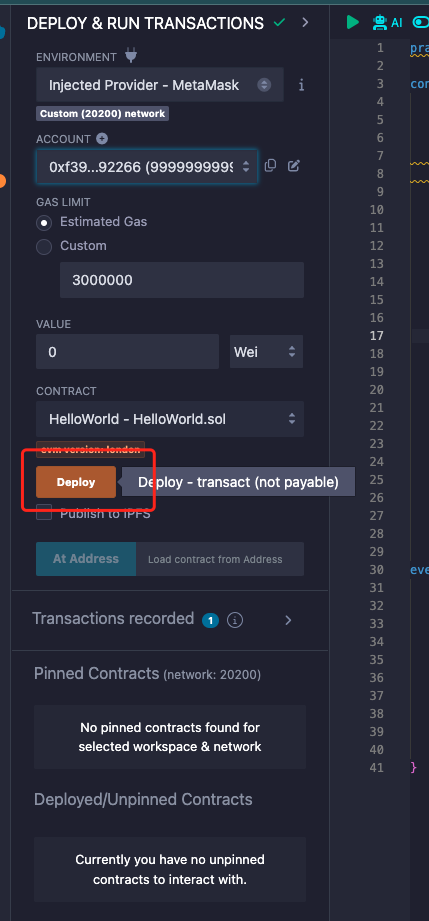
Solidity是目前区块链领域最有影响力的开发语言，搭建Solidity开发环境有以下几种方式：

1. **（推荐）**使用Remix 在线/本地搭建的方式，请参考链接：[在线](https://remix.ethereum.org/)、[本地](https://github.com/ethereum/remix-project)。使用Remix工具可以有语法提示、高亮、错误提示，可以方便地切换不同的Solidity版本。并且可以使用Remix自带的EVM工具对智能合约进行调试。

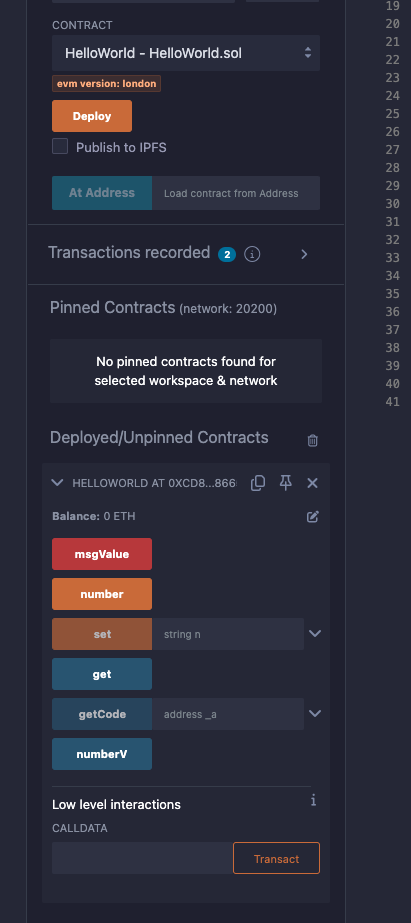
可以选择环境，并进行合约的部署



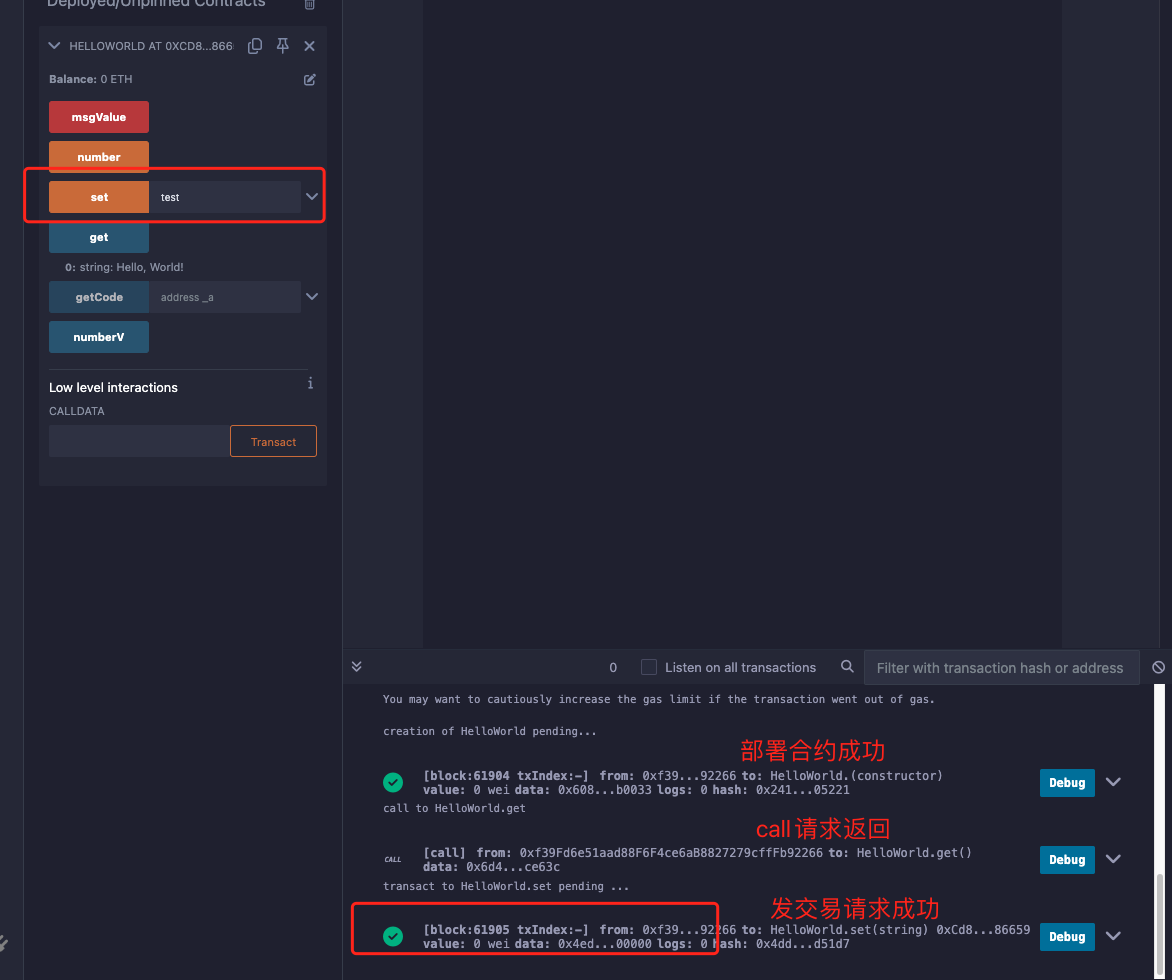
点击Deploy按钮，对合约进行部署操作。



部署合约成功后，会在下面的合约栏中出现对应的合约，以及合约方法。



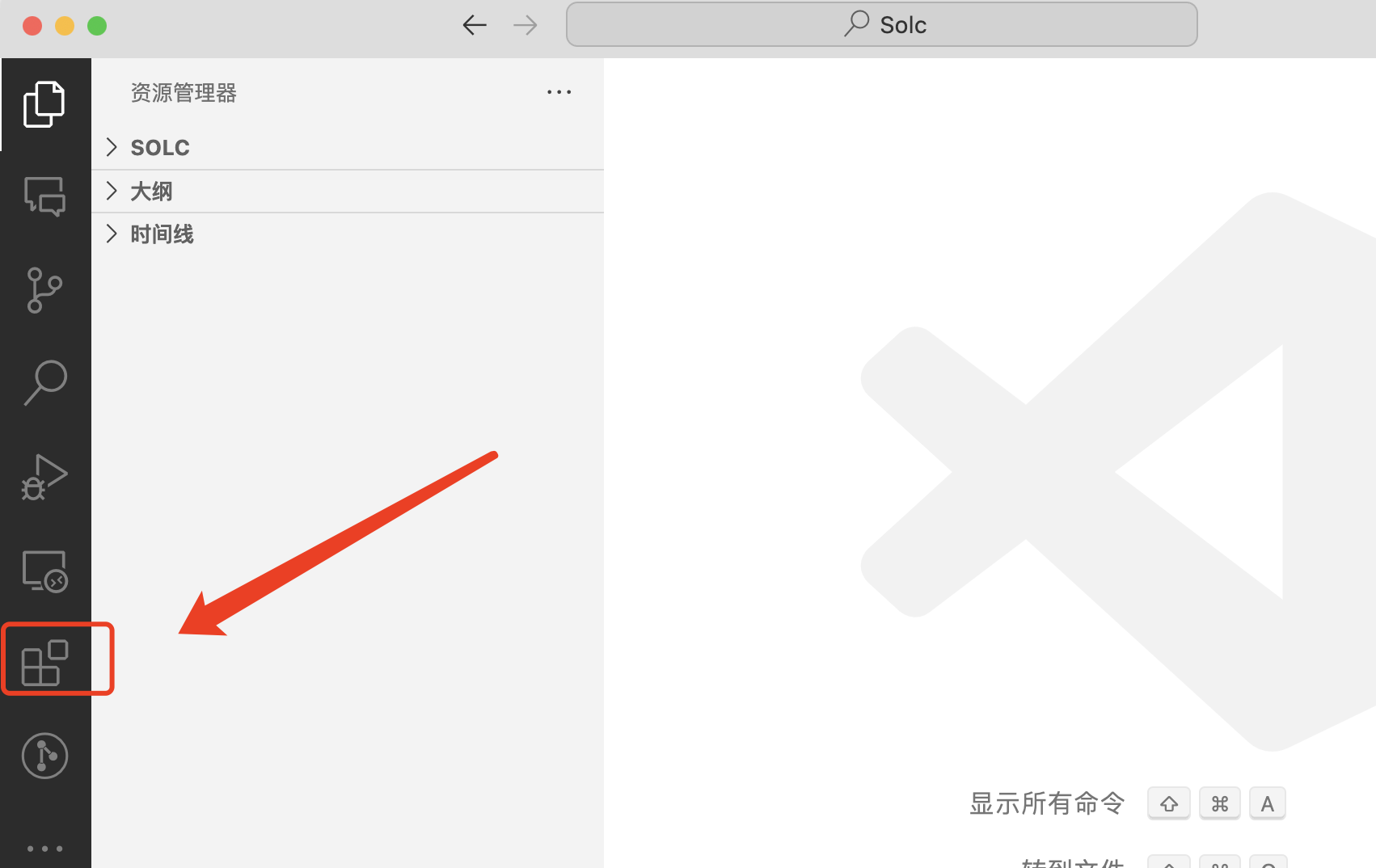
部署合约、发起call请求、发送交易请求，都会在下面框内有返回信息。



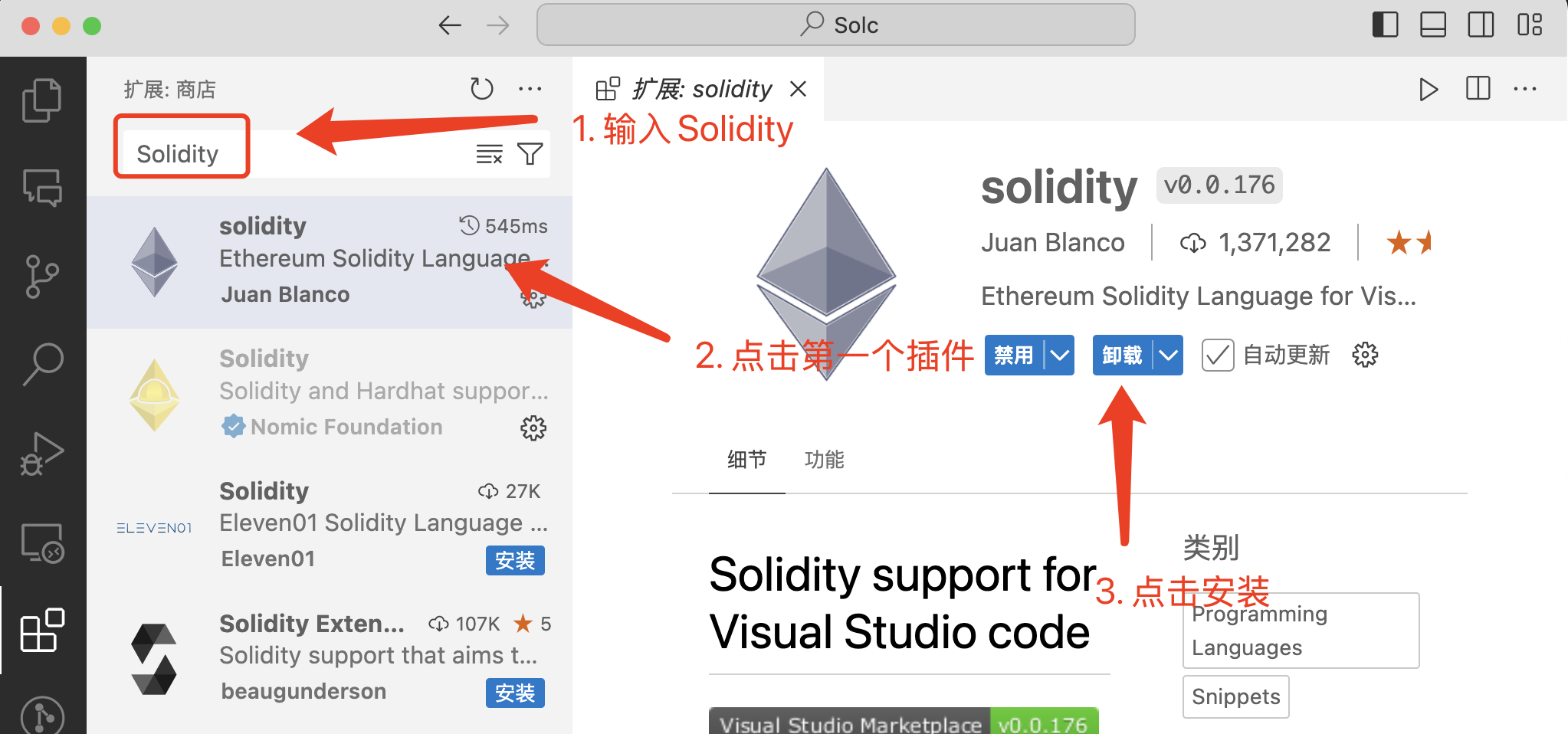
1. 使用VSCode + 插件的方式

可以在VSCode中使用Solidity插件，提供Solidity语法提示、高亮、错误提示等功能。

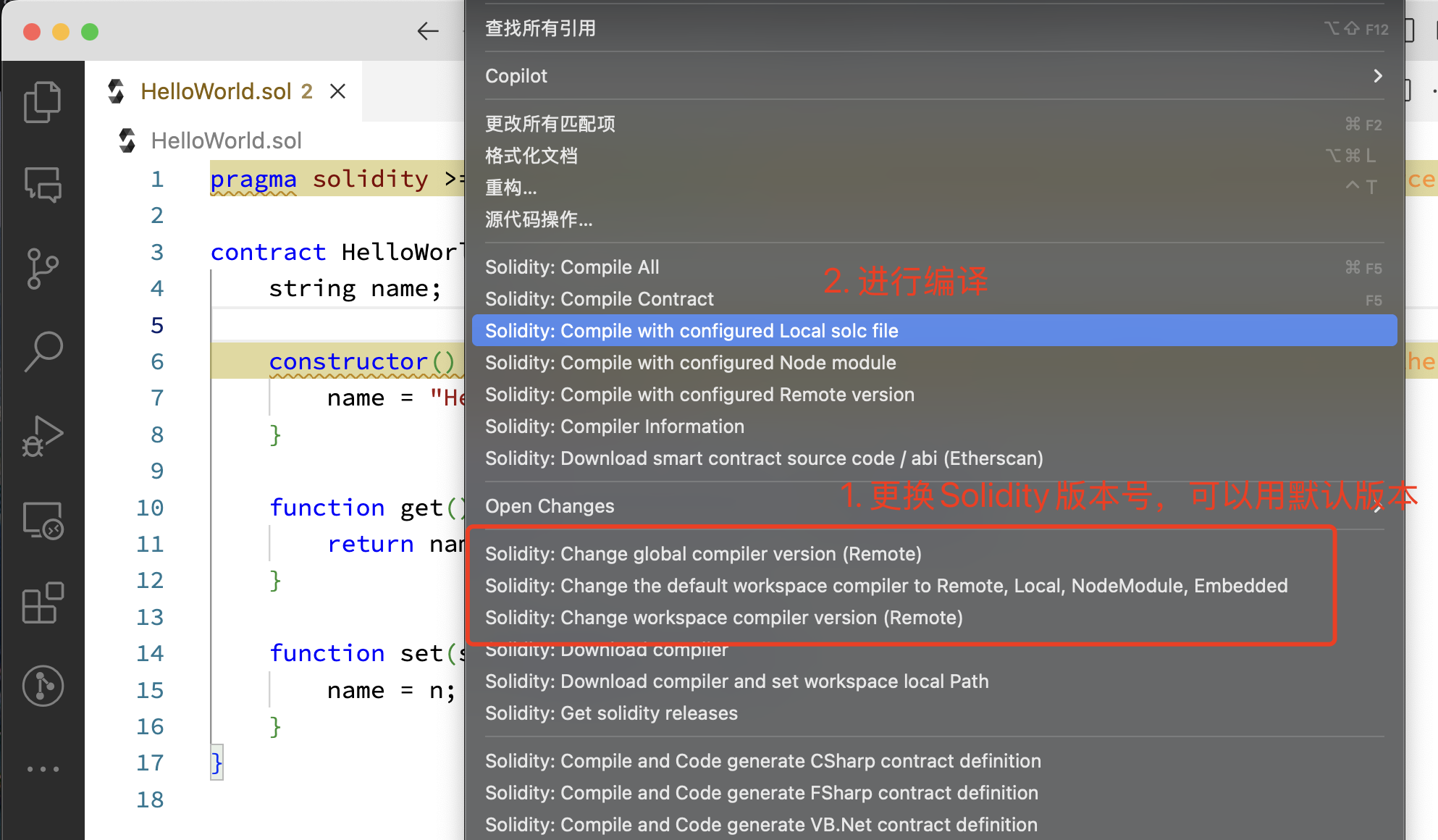
点击VSCode插件



搜索Solidity插件，并安装



鼠标右键，将会出现Solidity相关的配置以及编译选项。



1. 使用IntelliJ IDEA + 插件的方式，效果同上。
2. **实现ERC20标准的合约ERC20Test**
3. **ERC20定义**

ERC-20是一种同质化代币合约的标准，这是一个能实现智能合约中代币的应用程序接口标准。

ERC-20 的功能示例包括：

* 将代币从一个帐户转到另一个帐户
* 获取帐户的当前代币余额
* 获取网络上可用代币的总供应量
* 批准一个帐户中一定的代币金额由第三方帐户使用

参考内容链接：<https://ethereum.org/zh/developers/docs/standards/tokens/erc-20/#body>

如果智能合约实施了下列方法和事件，它可以被称为 ERC-20 代币合约。

1. **ERC20接口定义**

具体的接口定义如下：

|  |
| --- |
| // 接口定义  function name() public view returns (string) // 代币的名称  function symbol() public view returns (string) // 代币的缩写符号  function decimals() public view returns (uint8) // 代币精确位数，默认位18  function totalSupply() public view returns (uint256) // 代币总发行数量  function balanceOf(address \_owner) public view returns (uint256 balance) // 查询某个地址下的代币数量  function transfer(address \_to, uint256 \_value) public returns (bool success) // 向某个地址转账代币，代币数量为value  function transferFrom(address \_from, address \_to, uint256 \_value) public returns (bool success) // 从from地址向to地址转账，代币数量位value  function approve(address \_spender, uint256 \_value) public returns (bool success) // 允许某个地址的账号使用自己的代币，允许数量为value  function allowance(address \_owner, address \_spender) public view returns (uint256 remaining) // 查询spender地址可以使用多少属于owner的代币  // 事件  event Transfer(address indexed \_from, address indexed \_to, uint256 \_value)  event Approval(address indexed \_owner, address indexed \_spender, uint256 \_value) |

1. **ERC20Test实现**

对于ERC20接口最基础的实现可以参考：[openzeppelin-contracts/contracts/token/ERC20/ERC20.sol at master · OpenZeppelin/openzeppelin-contracts (github.com)](https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol)

1. **在FISCO BCOS中部署ERC20Test**

可以通过控制台向FISCO BCOS区块链进行合约的部署。控制台的部署调用可以参考实验1的步骤，此处不再赘述。

在实现好合约ERC20Test之后，将合约文件导出到控制台所在的文件夹，此处举例控制台的绝对路径在fisco目录下。使用命令：

cp ERC20Test.sol ~/fisco/console/contracts/solidity/

之后在控制台进行部署操作即可。若合约的构造函数中有参数输入需求，则将参数放置合约名之后即可。例如，举例实现的ERC20Test的构造函数为：

constructor (string memory name\_, string memory symbol\_) public;

那么在部署合约时需要输入参数name和symbol，下面我举例name的参数值为TestCoin，symbol的参数值为TEC进行部署：

[group0]: /> deploy ERC20Test TestCoin TEC

transaction hash: 0x796b573aece250bba891b9251b8fb464d22f41cb36e7cae407b2bd0a870f5b72

contract address: 0x6849F21D1E455e9f0712b1e99Fa4FCD23758E8F1

currentAccount: 0x7b047472a4516e9697446576f8c7fcc064f967fa