hardhat开发编译测试部署智能合约实操

* 项目搭建

首先，新建一个目录名为hardhat-tutorial，进入该目录，初始化一个node项目，执行npm init -y

文本

描述已自动生成

然后安装hardhat作为开发时依赖，执行

npm install -D hardhat

文本

描述已自动生成

接下来创建hardhat项目，执行

npx hardhat

会有两种选择：

1. 创建一个sample项目
2. 仅仅创建一个空的hardhat.config.js文件

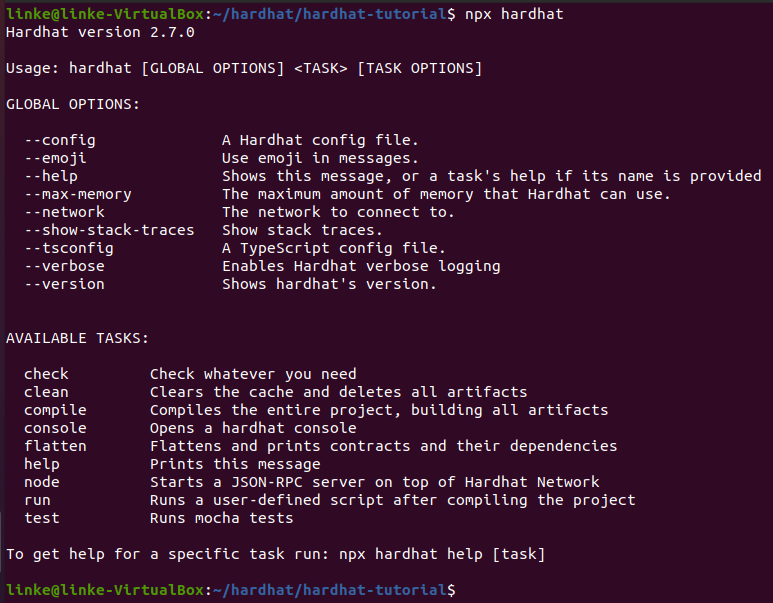
如果选择第二种，一个空的hardhat.config.js文件会被创建，然后我们可以根据自己的需求配置相关的依赖。

文本

描述已自动生成

这里我选择第二种，创建一个空的hardhat.config.js文件。创建hardhat项目成功之后会自动创建hardhat.config.js文件。

接下来如果想看hardhat能够为我们提供什么功能，有些什么命令，可以再次执行命令npx hardhat



在开发智能合约之前，还需要安装好以下几个依赖：waffle, ethers, chai，这些package用于连接以太坊节点以及智能合约的测试，执行以下命令

npm install -D @nomiclabs/hardhat-ethers ethers @nomiclabs/hardhat-waffle ethereum-waffle chai

文本

描述已自动生成

接下来需要在hardhat.config.js配置文件中引入waffle package

图形用户界面, 文本

描述已自动生成

只需要引入waffle，不需要引入ethers，因为waffle依赖于ethers，waffle就会引入ethers

同时，把solidity版本改为0.8.0

* 创建智能合约

当前的目录结构如下

文本

描述已自动生成

创建一个contracts目录用于存放智能合约代码，执行

mkdir contracts

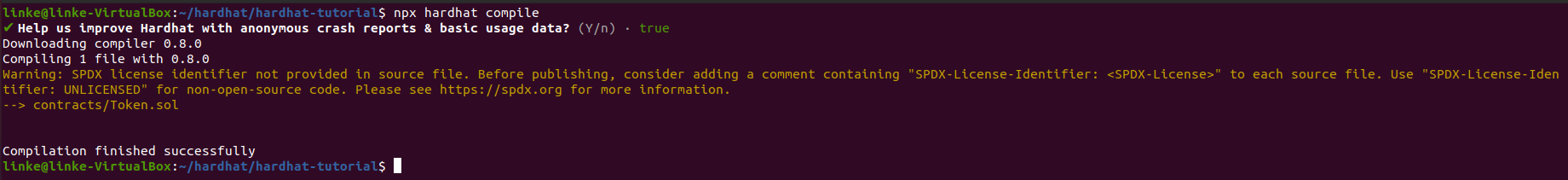
然后再在contracts下创建一个Token.sol智能合约，这是一个非常简单的token合约，内容为

文本

描述已自动生成

合约已经创建好了，首先编译一下，执行

npx hardhat compile



编译成功。

会看到多了两个目录

文本

描述已自动生成

文本

描述已自动生成

artifacts/contracts/Token.sol/Token.json文件包含合约编译后产生的abi和字节码。

接下来创建一个test目录，用于存放测试用例

mkdir test

再创建一个test/Token.js文件，用于存放Token.sol的测试用例，内容如下

文本

描述已自动生成

beforeEach是指在执行每个测试用例之前要执行的操作，describe是一个测试套件（test suite），it是一个测试用例。可以看到，一共有2个测试套件，共4个测试用例。

接下来执行以上的测试用例

执行npx hardhat test

文本

描述已自动生成

全部执行通过。

* 调试智能合约

接下来试一试hardhat提供的debug功能。hardhat提供了一种功能，可以在智能合约中打印出变量的值，有利于debug。智能合约的debug一直都是开发人员的一个痛点，因为不能打断点进行调试，所以hardhat提供的这个功能在一定程度上解决了这个痛点。

接下来在合约中加上debug功能

文本

描述已自动生成

合约修改后，再次执行npx hardhat test

文本

描述已自动生成

可以看到，在测试执行过程中，打印出了变量的信息，非常有利于debug。

* 部署智能合约

创建一个目录scripts，并在下面创建一个deploy.js文件，内容如下

文本

描述已自动生成

然后执行以下命令部署合约

npx hardhat run scripts/deploy.js

文本

描述已自动生成

注意，这个命令并没有跟额外的参数，没有指定网络，意思就是把合约部署在hardhat的本地网络中，即一个内置的Hardhat Network，类似于Ganache，但只存在于内存中，命令执行结束网络就被删除了，也就是说在命令结束之后，无法和合约进行交互。

接下来我们把合约部署到infura接口，infura提供以太坊和IPFS的接入接口，简介如下

文本

描述已自动生成

首先进入infura.io，然后在infura中创建一个project，如下

图形用户界面, 应用程序

描述已自动生成

网络选择ROPSTEN

图形用户界面, 文本, 应用程序

描述已自动生成

把连接ropsten网络的链接<https://ropsten.infura.io/v3/e4f75d737fcb49b2a0384ca325edb001拷贝到hardhat.config.js>文件中，定义为一个常量

并且把本地metamask里面账户的私钥导出来，也放到hardhat.config.js文件中，定义为常量

同时，添加networks

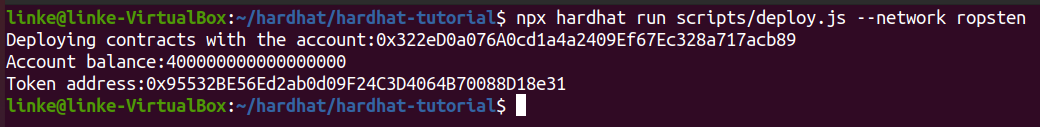
文本

描述已自动生成

从上图可以看到，在hardhat.config.js中定义了网络URL以及对应的私钥，就可以用这个私钥在这个网络中部署合约。

接下来执行

npx hardhat run scripts/deploy.js –network ropsten



把合约地址0x95532BE56Ed2ab0d09F24C3D4064B70088D18e31拿到<https://ropsten.etherscan.io/>去查

电脑网站的截图

描述已自动生成

可以看到，的确已经部署到ropsten测试网上去了。