技术文档

前提：

第一章：

1. 熟悉Solidity 语法，会编写智能合约。

（Solidity官方文档：<https://solidity-cn.readthedocs.io/zh/develop/>）

1. 熟悉Linux操作环境和命令。

（菜鸟教程: <https://www.runoob.com/linux/linux-tutorial.html>）

第二章：

1. 会java开发（spring，spring boot等）

（视频连接：

Spring:<https://www.bilibili.com/video/BV1WE411d7Dv?from=search&seid=16837389262841391006>

Spring MVC:<https://www.bilibili.com/video/BV1aE41167Tu>

Mybatis:<https://www.bilibili.com/video/BV1NE411Q7Nx?from=search&seid=16837389262841391006>

Spring boot：<https://www.bilibili.com/video/BV1PE411i7CV?from=search&seid=13065534905438519564>）

1. 会使用gradle构建工具。

（视频连接：<https://www.bilibili.com/video/BV1iW411C7CV?from=search&seid=7672268708494278128>

）

第三章：

1. 会使用vue框架。

（视频连接：

<https://www.bilibili.com/video/BV18E411a7mC?from=search&seid=17130356317148803657>）

目录

[一 区块链网络节点搭建 4](#_Toc69996495)

[1 下载脚本build\_chian.sh 5](#_Toc69996496)

[2 下载控制台 6](#_Toc69996497)

[二 开发第一个区块链应用 7](#_Toc69996498)

[1 .编写一个简单的智能合约 7](#_Toc69996499)

[2.创建项目环境 8](#_Toc69996500)

[2.1 . 修改application.yml 8](#_Toc69996501)

[2.2．拷贝证书 8](#_Toc69996502)

[2.3. 将remix里面的智能合约拷贝到该项目 9](#_Toc69996503)

[2.4. 编译智能合约 9](#_Toc69996504)

[2.5. 合约层-contract层 11](#_Toc69996505)

[2.6. 控制层-Controller层 13](#_Toc69996506)

[三 开发前端 15](#_Toc69996507)

[1.jar包引入 15](#_Toc69996508)

[2.Vue框架的使用 16](#_Toc69996509)

[2.1安装vue-cli脚手架 16](#_Toc69996510)

[2.2编写前端页面： 19](#_Toc69996511)

[总结：项目开发整体流程 25](#_Toc69996512)

[附录： 26](#_Toc69996513)

[WINSCP教程: 26](#_Toc69996514)

[1.在云服务器上设置一个密钥文件，保存到本地。按红框走 26](#_Toc69996515)

[2.打开WINSCP软件，按着红框往下走 28](#_Toc69996516)

# 一 区块链网络节点搭建

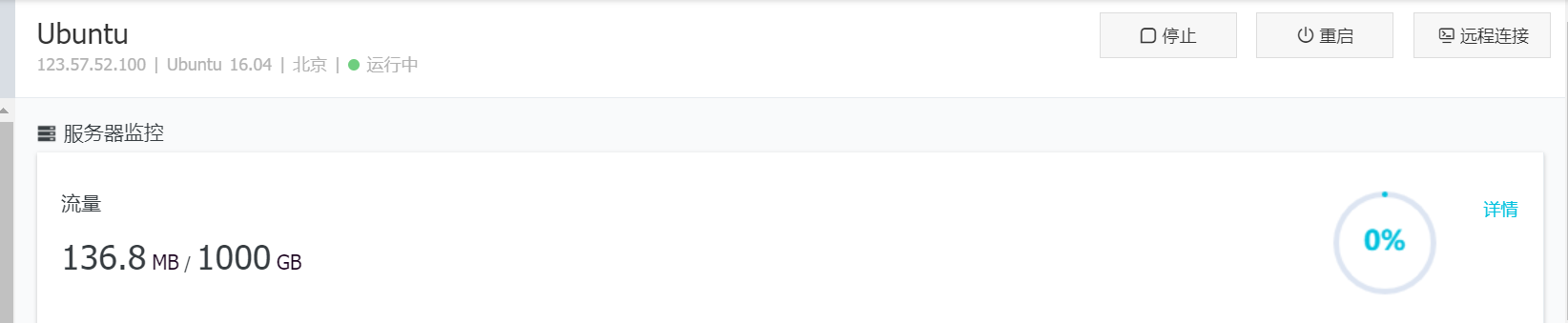
前提：熟悉linux操作环境

按照官方文档“搭建第一个区块链网络”的步骤就可以。（<https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/installation.html>）

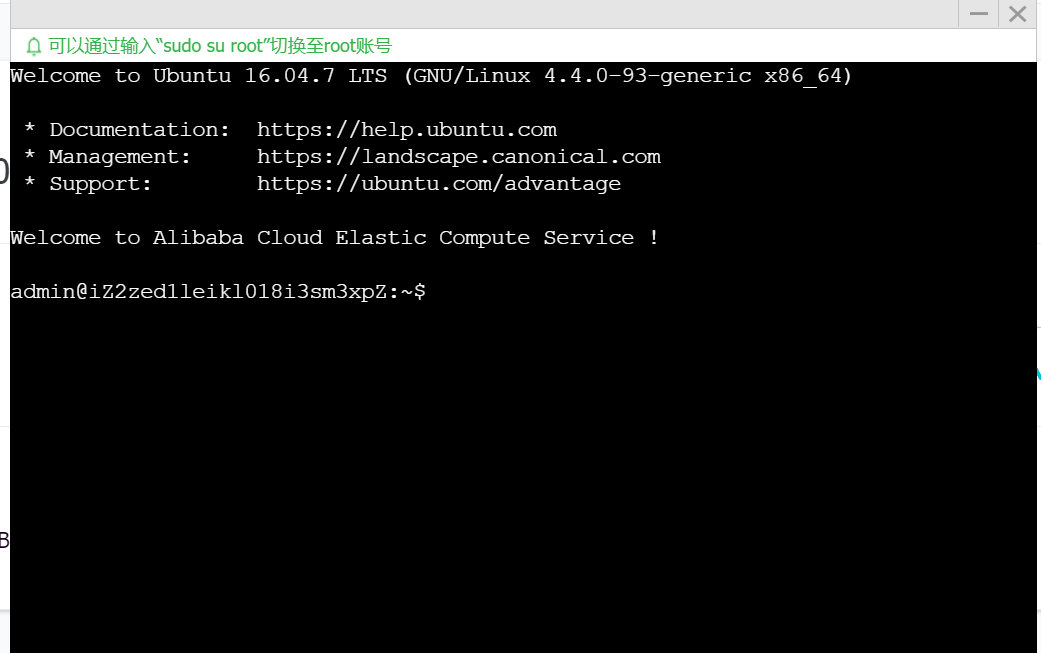
可以选择在虚拟机里搭建，也可以选择在云服务器上

我这里在阿里云服务器里面搭建。（以Ubuntu 16.04为例）

云服务器IP地址：123.57.52.100



点击远程连接：



进入root模式：sudo su root



安装依赖：sudo apt install -y openssl curl（build\_chain.sh依赖）



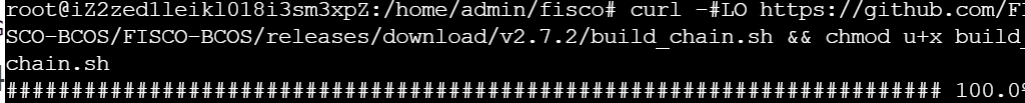
## 1 下载脚本build\_chian.sh

我现在是在目录/home/admin下进行操作

* 建立目录fisco：mkdir fisco 切换目录：cd fisco



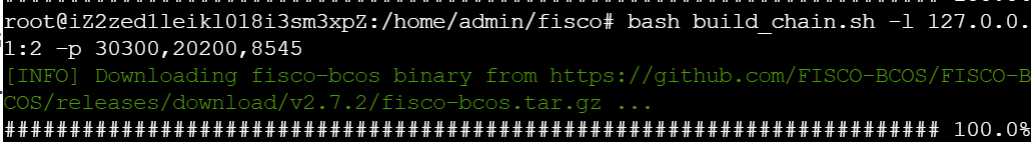
* 下载脚本build\_chian.sh: curl ……



* 搭建单群组2节点区块链：bash build\_chain.sh -l 127.0.0.1:2 -p 30300,20200,8545

（-l 指定ip地址，群组个数，节点数

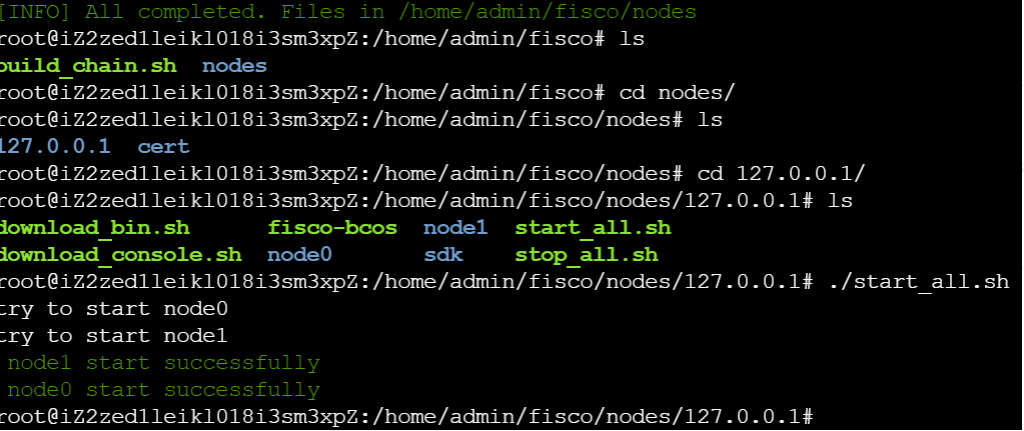
-p 指定起始端口，分别是p2p\_port,channel\_port,jsonrpc\_port）



* 启动节点：

cd nodes/127.0.0.1

./start\_all.sh



现在有一个单群组2节点的区块链启动在IP地址为123.57.52.100的云服务器上，2个节点的端口分别为20200,20201

利用build\_chain.sh搭建一个区块链网络的所有命令

* sudo apt install -y openssl curl
* mkdir fisco
* cd fisco
* curl -#LO https://github.com/FISCO-BCOS/FISCO-BCOS/releases/download/v2.7.2/build\_chain.sh && chmod u+x build\_chain.sh
* bash build\_chain.sh -l 127.0.0.1:2 -p 30300,20200,8545
* bash nodes/127.0.0.1/start\_all.sh
* ps -ef | grep -v grep | grep fisco-bcos

## 2 下载控制台

这里可以直接参考官方文档

# 二 开发第一个区块链应用

前提：熟悉solidity语法，熟悉java开发（spring，spring boot），会使用gradle构建工具。

这里我就不按照官方文档的例子了。

## 1 .编写一个简单的智能合约

（可以先在remix里编译成功）

**pragma** solidity 0.4.25;  
  
**contract** MintCoin{  
 **address public** minter;  
 **mapping** (**address** => **uint**) **public** balances;  
  
 **event** Sent(**address** from,**address** to, **uint** amount);  
 **event** Mint(**address** from,**address** to, **uint** amount);  
 **constructor**() **public** {  
 minter = msg.sender;  
 }  
 **modifier** onlyBank{  
 require(msg.sender == minter);  
 \_;  
 }  
 **function** tokenMint(**address** receiver, **uint** amount) onlyBank **public** {  
 balances[receiver] = amount;  
 }  
 **function** tokenSend(**address** payer,**address** payee,**uint** amount) **public** {  
 balances[payer] -= amount;  
 balances[payee] += amount;  
 **emit** Sent(payer,payee, amount);  
 }  
 **function** checkBalance(**address** receiver) **public returns**(**uint** amount){  
 amount = balances[receiver];  
 }  
}

## 2.创建项目环境

先从github下载一个FISCO BCOS提供的Spring Boot的初始化项目。在这个项目里，它已经编写好了一些配置文件和配置类，我们通过修改它的配置文件来连接上我们在云服务器里搭建的区块链。

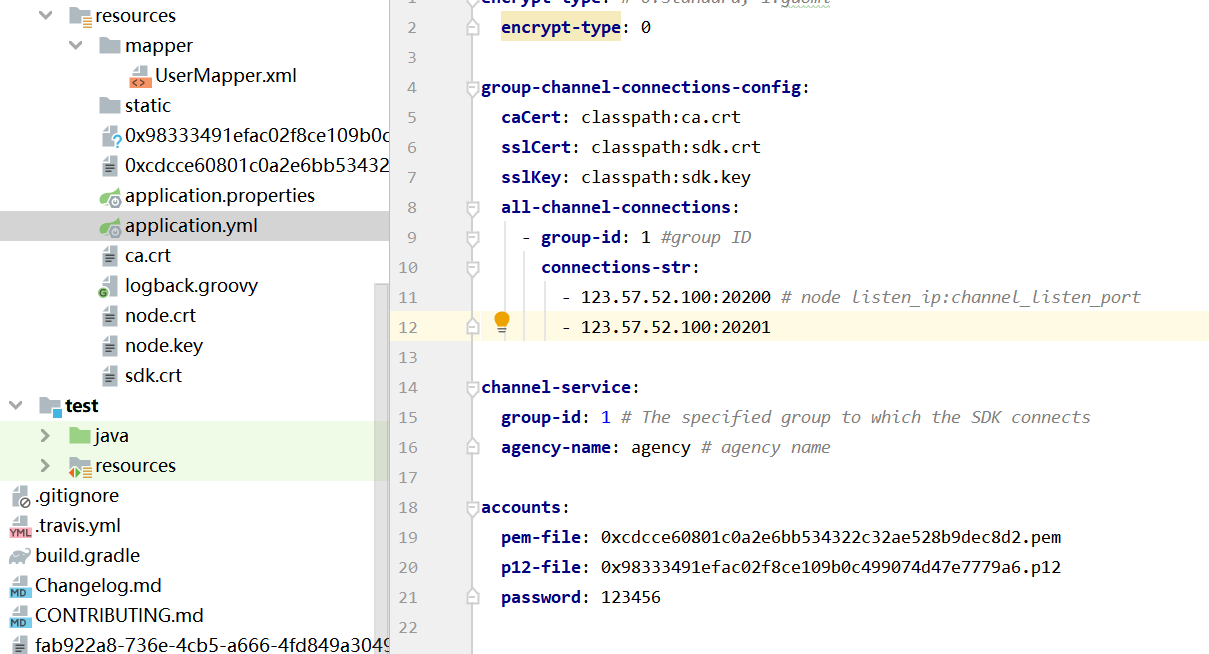
（github的初始化项目连接：https://github.com/FISCO-BCOS/spring-boot-starter）



这里我使用的是idea打开这个项目。（idea可以从idea官网下载，学生邮箱可以免费试用一年）

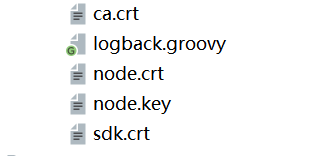
### 2.1 . 修改application.yml

根据云服务器的IP地址以及相关节点的端口号，机构名等，修改application.yml。也可以修改账户地址，黄框对应的是账户私钥文件。

我的阿里云服务器的IP地址：123.57.52.100，对应的节点端口号为：20200,20201，机构名是agency 

### 2.2．拷贝证书

将云服务器上搭建的区块链的相关证书拷到/src/main/resources这个目录下。

分别是： 这些证书在nodes/${ip}/sdk目录下

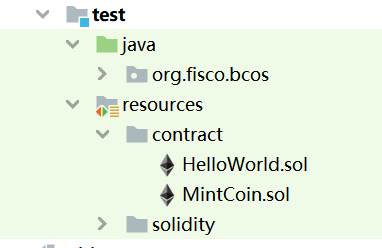


可以安装一个WINSCP软件，便于从云服务器下载和上传文件。

（教程见附录，以从云服务器上下载区块链的相关证书为例）

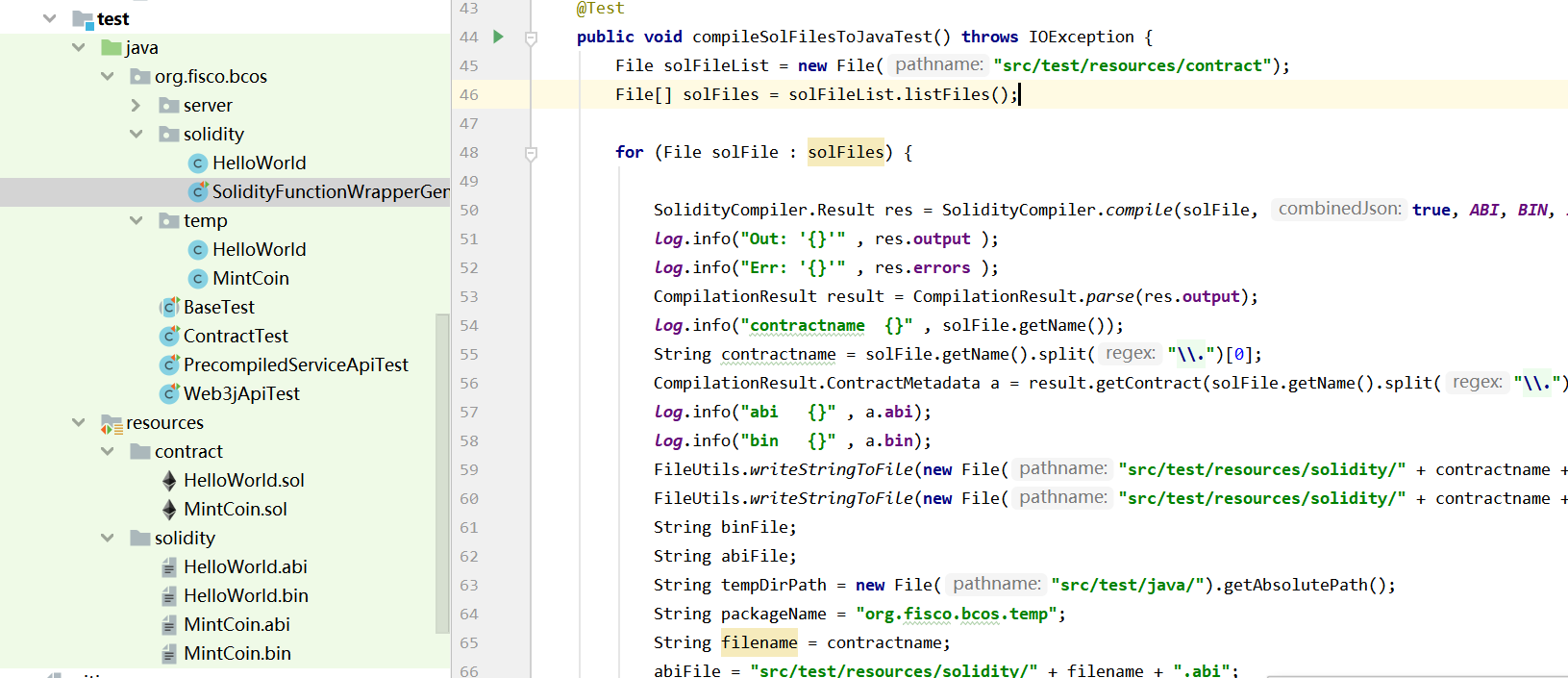
### 2.3. 将remix里面的智能合约拷贝到该项目

在/test/resources/contract新建文件MintCoin.Sol，将remix里面编译好的智能合约拷贝到该文件。



### 2.4. 编译智能合约

#### 将智能合约编译成java代码

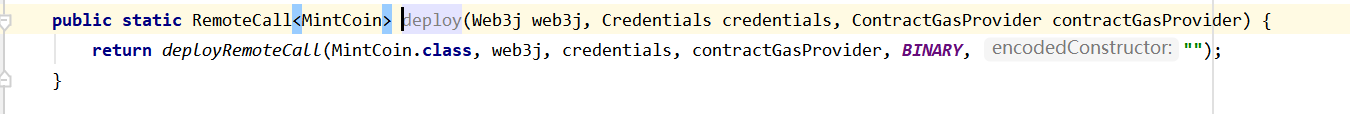


MintCoin.sol放在/test/resources/contract目录下，打开test/java/org/fisco/bcos/solidity目录下的SolidityFunctionWrapperGeneratorTest这个类，运行这个测试函数，会把对应src/test/resources/contract目录下的智能合约编译成java代码，放在src/test/java/org/fisco/bcos/temp目录下。相应的abi文件和bin文件在src/test/resources/soldity的目录下。当然这些目录都可以修改。

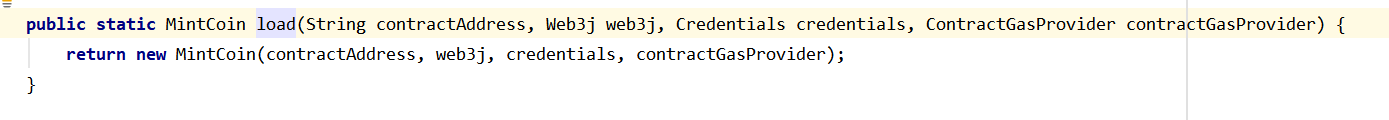
#### 分析编译成的java代码

挑几个重点函数说一下是干什么的

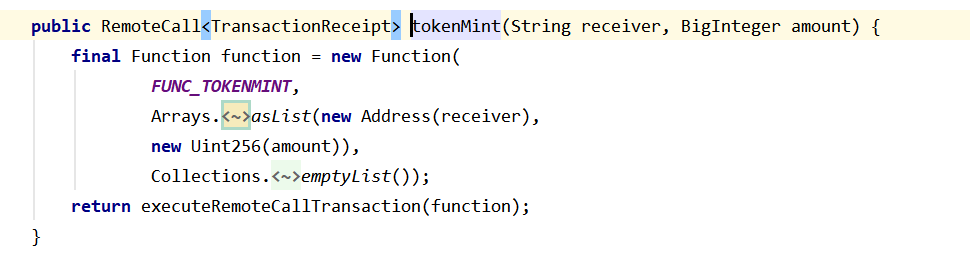
1. deploy部署合约，合约没有部署的情况下，需要调用这个函数部署



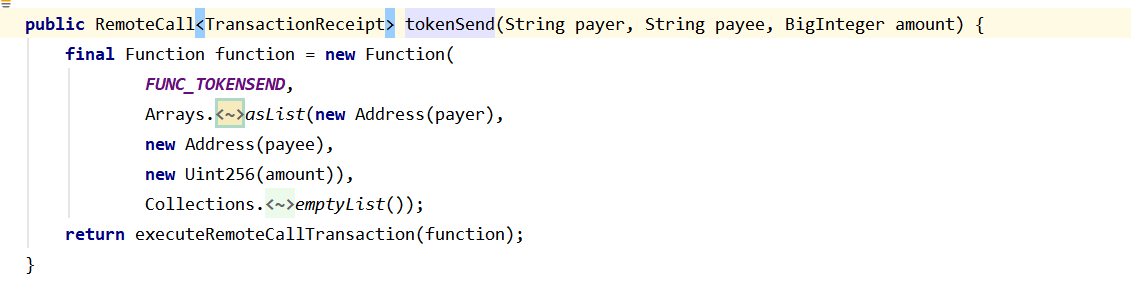
1. load加载合约，合约已部署的情况下，需要调用这个函数加载合约。



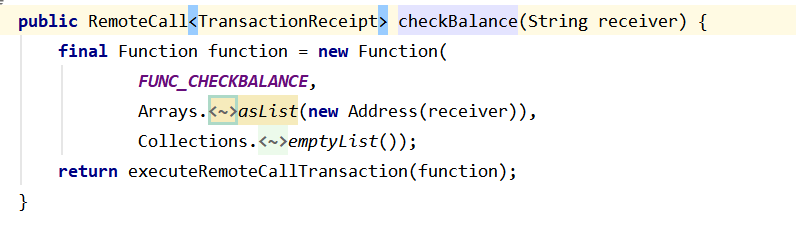
1. 对应MintCoin.sol中的tokenMint函数



1. 对应MintCoin.sol中的tokenSend函数



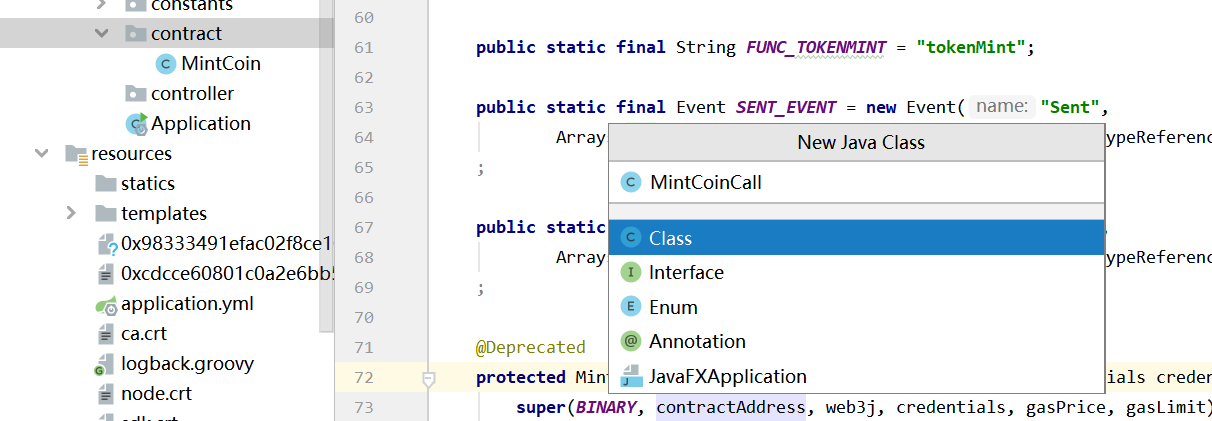
1. 对应MintCoin.sol中的checkBalance函数



### 2.5. 合约层-contract层

在/src/main/java/org/fisco/bcos/下创建contract目录，将之前编译好的MintCoin智能合约的java代码复制到这个目录下。

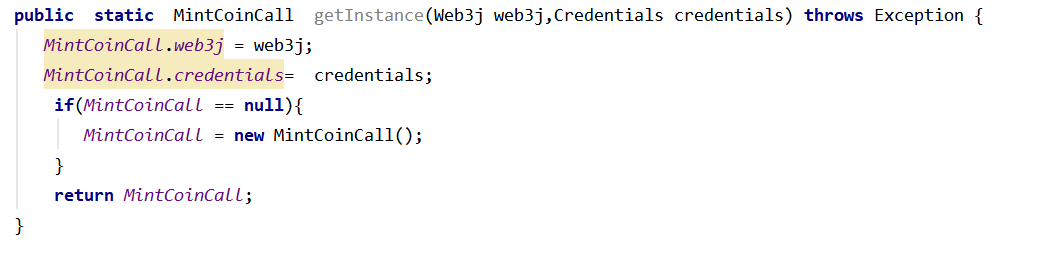
###### 新建MintCoinCall类，部署和调用MintCoin



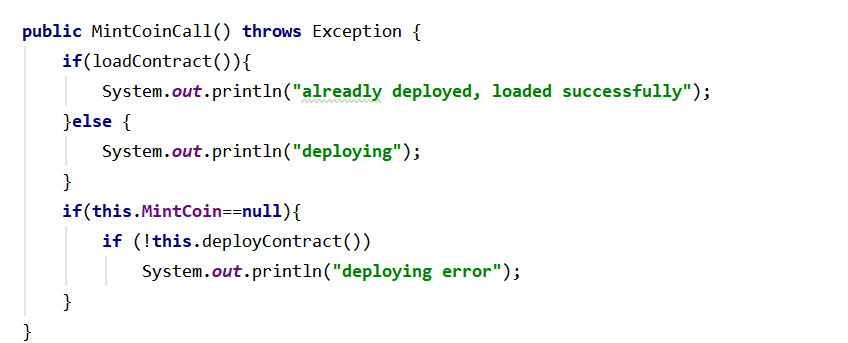
这里我创建一个类MintCoinCall，通过getInstance函数实例化这个类，它可以部署和调用智能合约MintCoin。

Ps：从官网下载的这个项目，web3j与credentials已经通过配置类进入ICO容器中，可以通过@Autowired注解实现自动装配注入。稍后，我会在Controller层利用注解@Autowired进行web3j与credentials的注入，然后利用getInstance函数将可以两个参数传给MintCoinCall这个类，它便可以利用web3j和credentials部署调用合约（这里需要一些spring boot的知识，所以我不详细说了，到时候我发一下b站关于spring boot的视频）





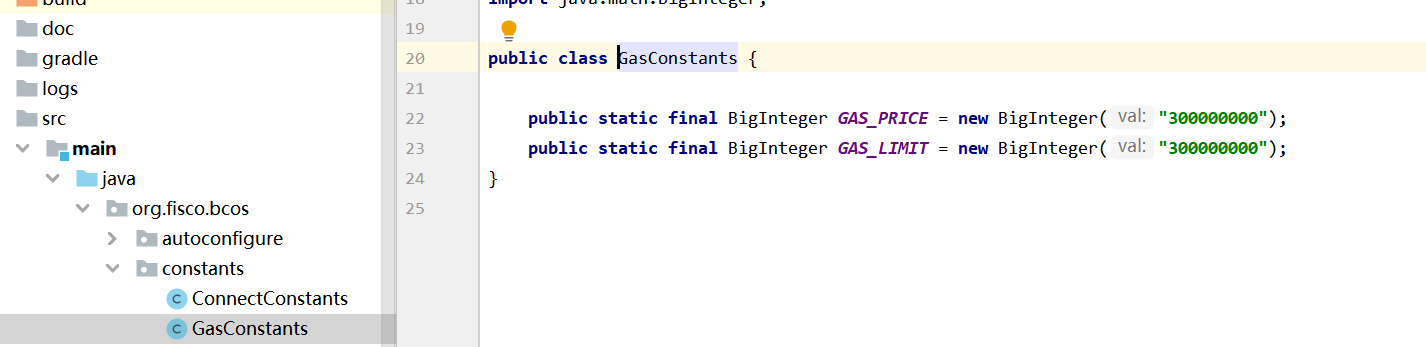
如果合约已经部署的话，就调用loadContract()加载合约，没有部署的话调用deployContract()部署合约。



黄框画的是加载合约和部署合约需要的参数，web3和credentials是一会在controller层传递过来的，GasConstants.GAS\_PRICE和，GasConstants.GAS\_LIMIT是常数，已经通过配置类设置好了。



配置类



### 2.6. 控制层-Controller层

在这一层，一方面和前端界面交互，一方面和合约端交互。（红框是调用合约端的代码，绿框是与数据库的交互，蓝框是映射一个前端页面的http请求，比如http://localhost:9090/mintCoin/registerBalances， <http://localhost:9090>前一部分是后端项目启动的IP地址和端口，后一部分与controller层的mapping对应，像这个url就会进入register这个函数，进行相关的业务处理。）

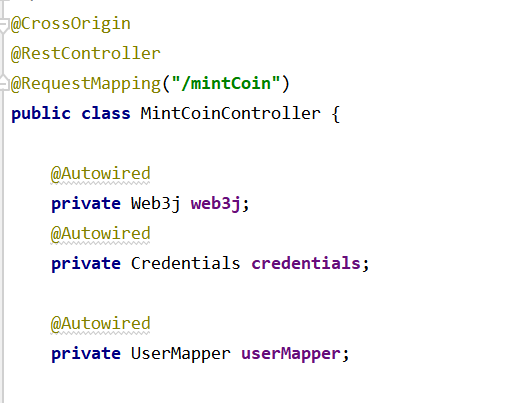
@CrossOrigin 允许跨域请求（因为采用的是前后端分离的形式）

@RestController注解，用来表示某个类的是否可以接收HTTP请求。

@PostMapping注解，映射一个POST请求。

MintCoinCall这个类与智能合约端交互。

紫框是利用request获取前端传过来的参数，黄框是创建账户地址







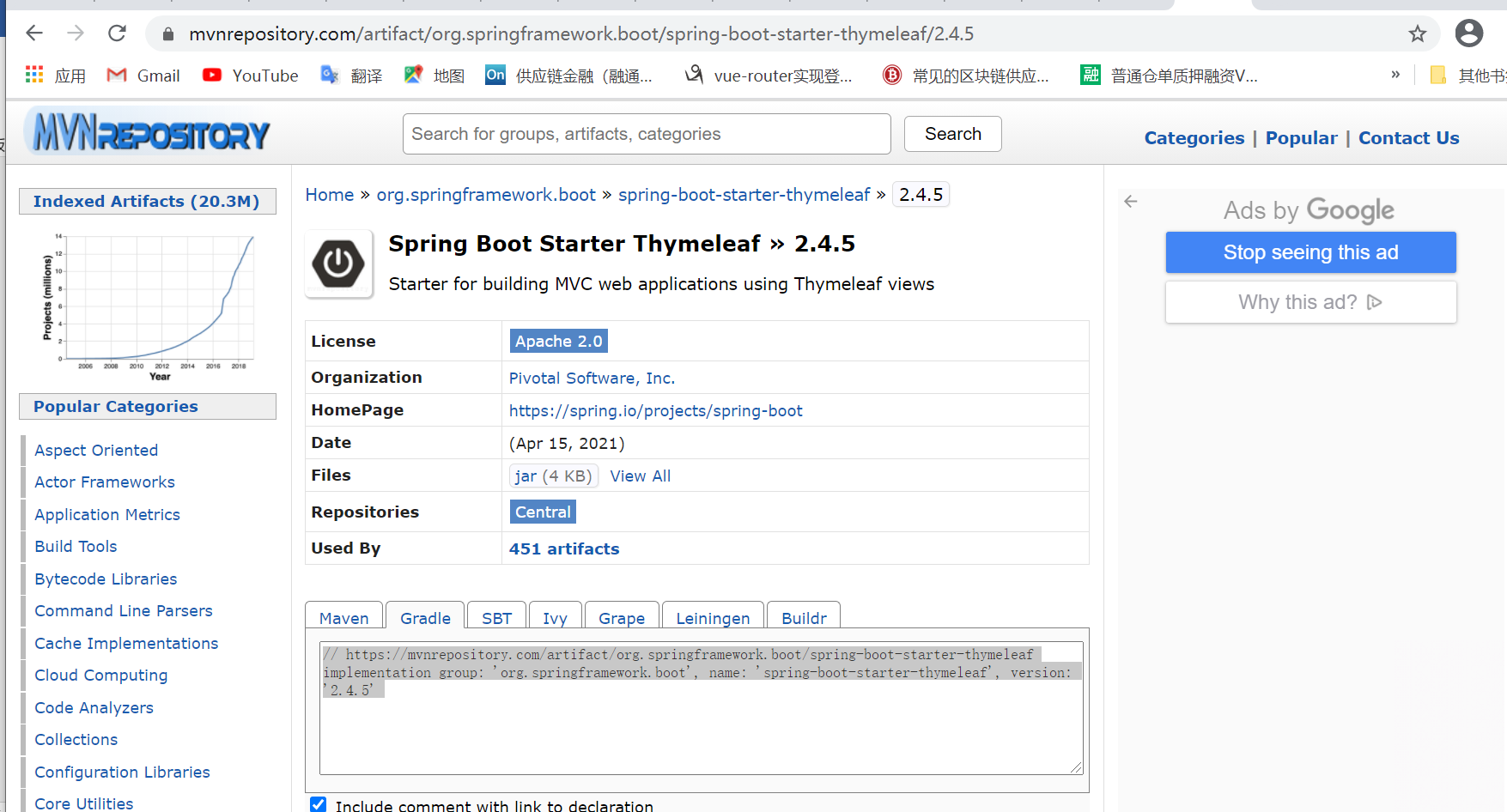
# 三 开发前端

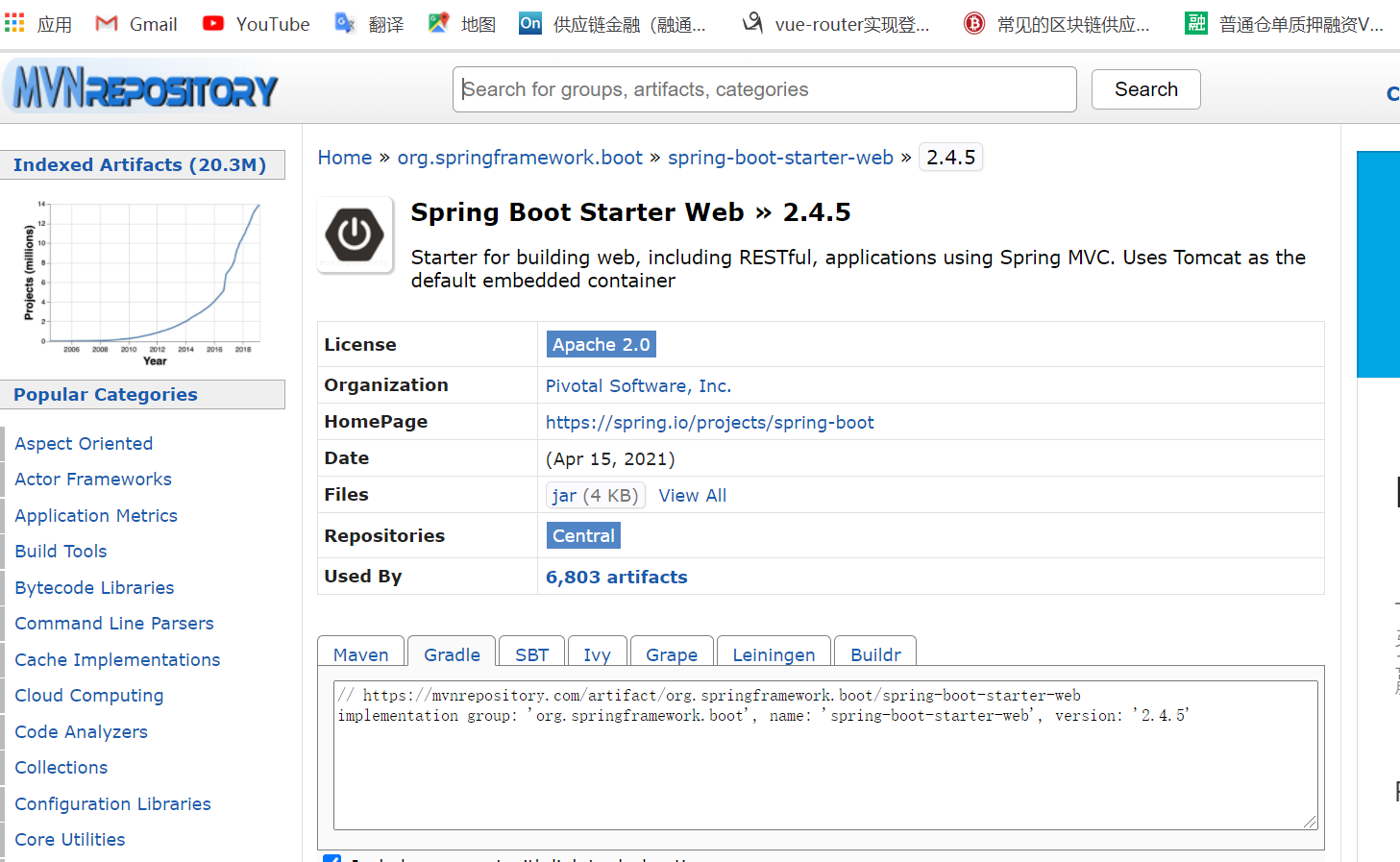
这里我使用的是前后端分离的框架，其实也可以试试springboot推荐的thymeleaf模板，进行web前端的开发。

前提：需要学习Vue前端框架

## 1.jar包引入

* 1. 先从maven仓库拿到对应jar包，选择复制gradle下对应的spring-boot-starter-web和spring-boot-starter-thymeleaf的jar包坐标。（如果不用thymeleaf模板，就不用引入这个jar包了）





* 1. 将其粘贴到build.gradle下的dependencies中



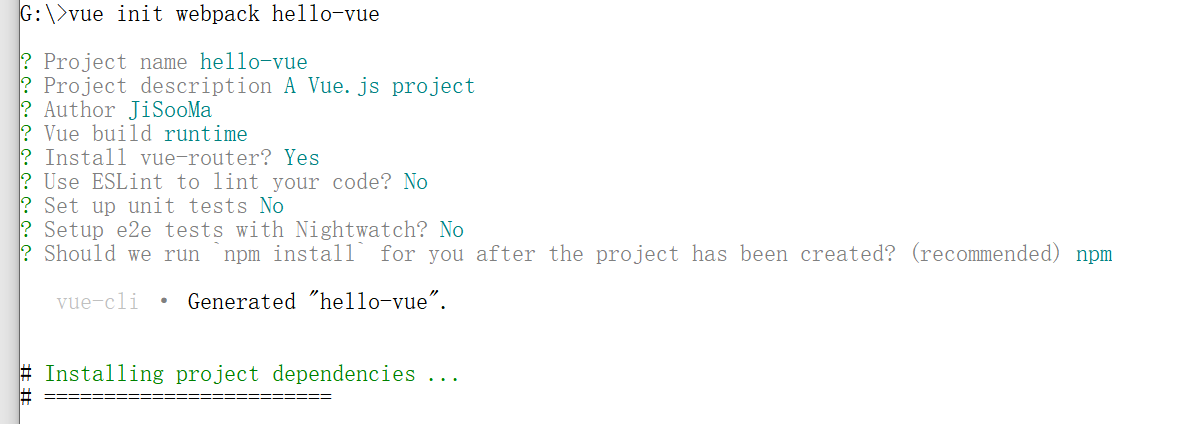
## 2.Vue框架的使用

### 2.1安装vue-cli脚手架

参考博客:https://www.cnblogs.com/fengzhenxiong/p/10213198.html

#### 2.1.1创建Vue前端工程

以管理员模式打开cmd，切换到对应的工程目录下，这里我直接选定G盘。



#### 2.1.2下载element-ui，axios等

element-ui：界面显示，前端显示的组件库。

npm i element-ui -S

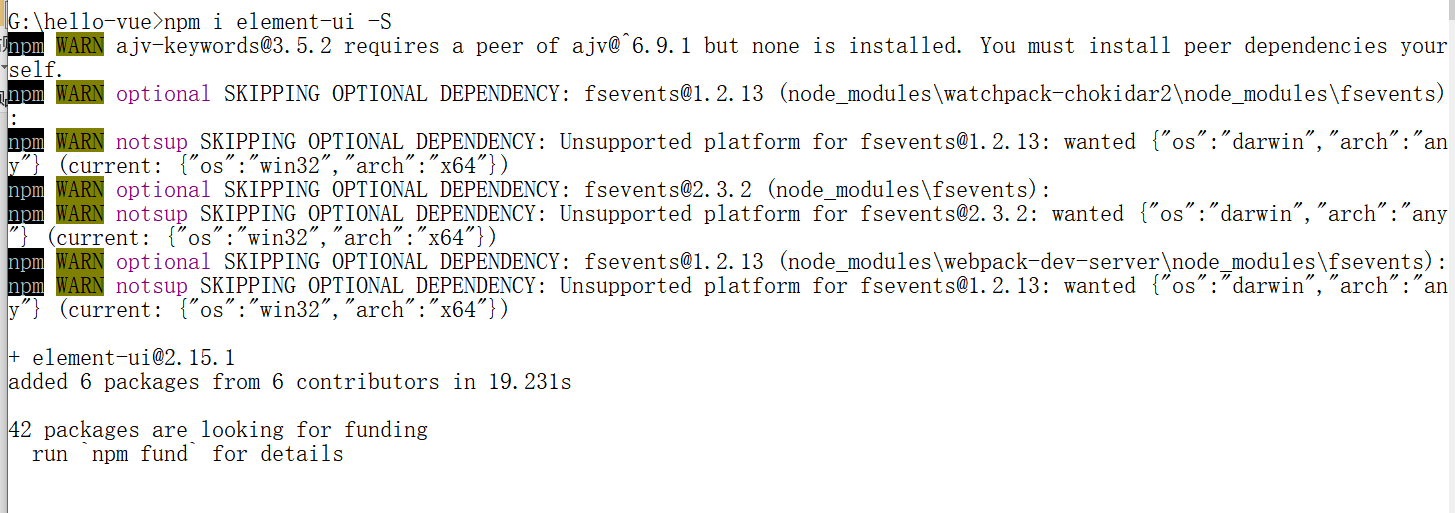
npm install sass-loader node-sass --save-dev

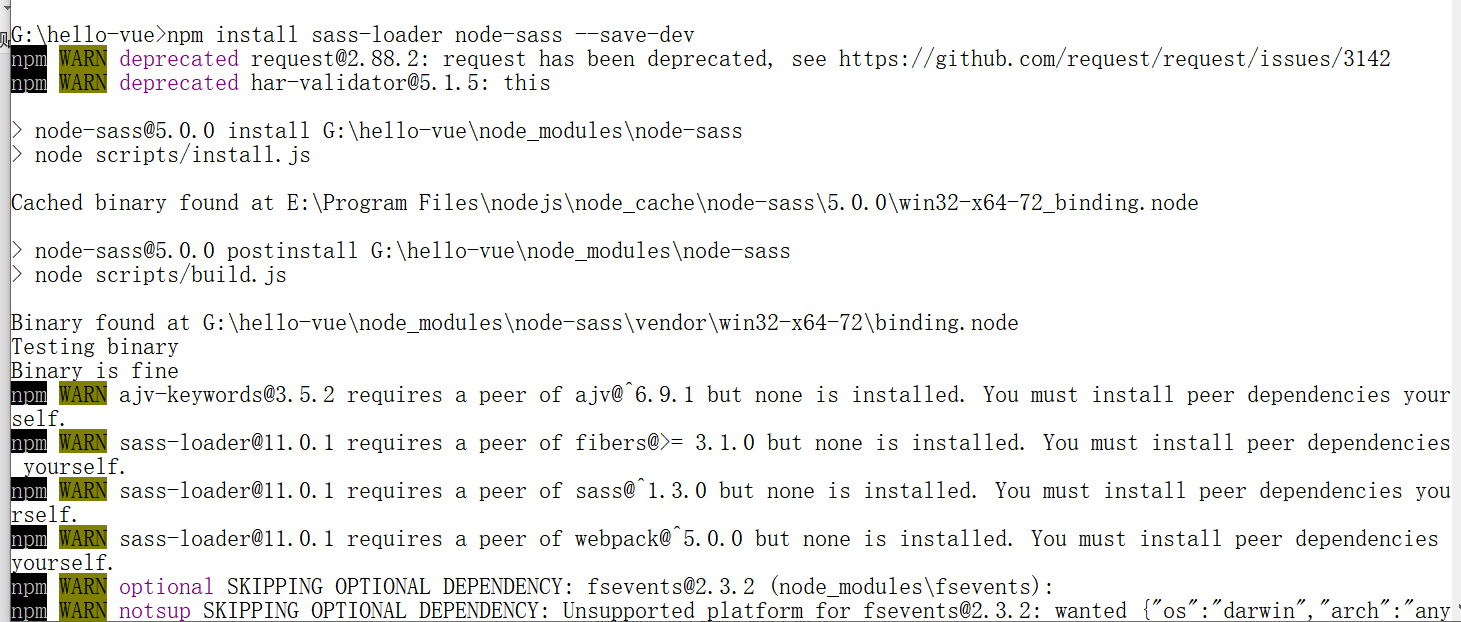
axios：通信框架，与后端通信。

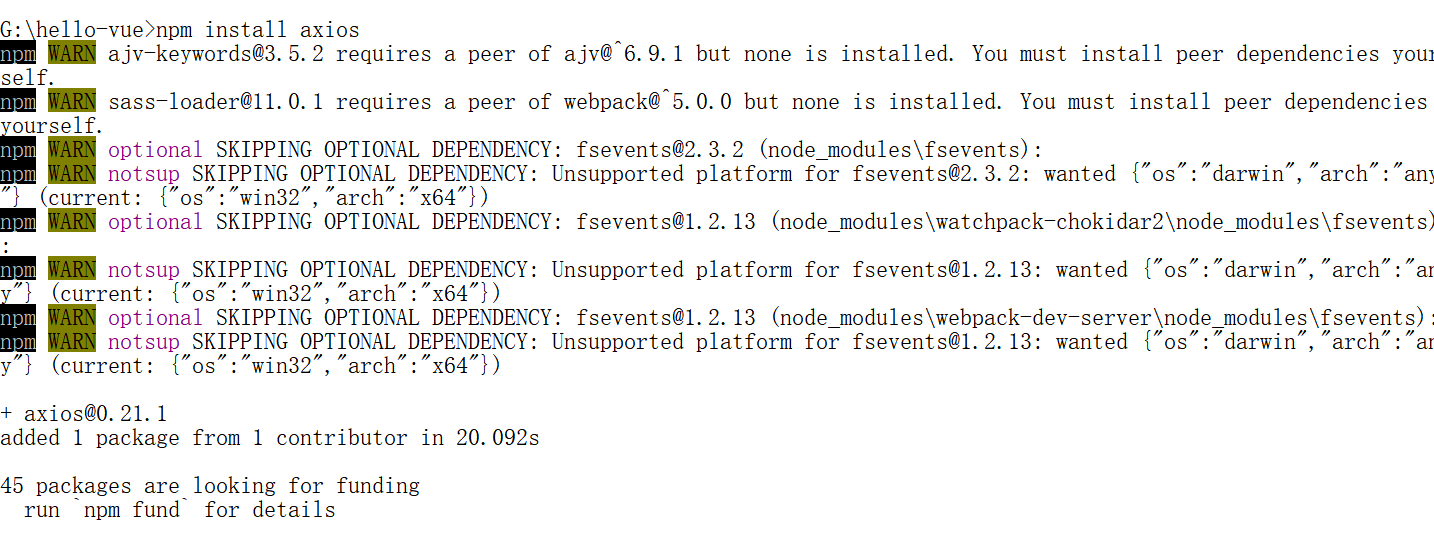
npm install axios

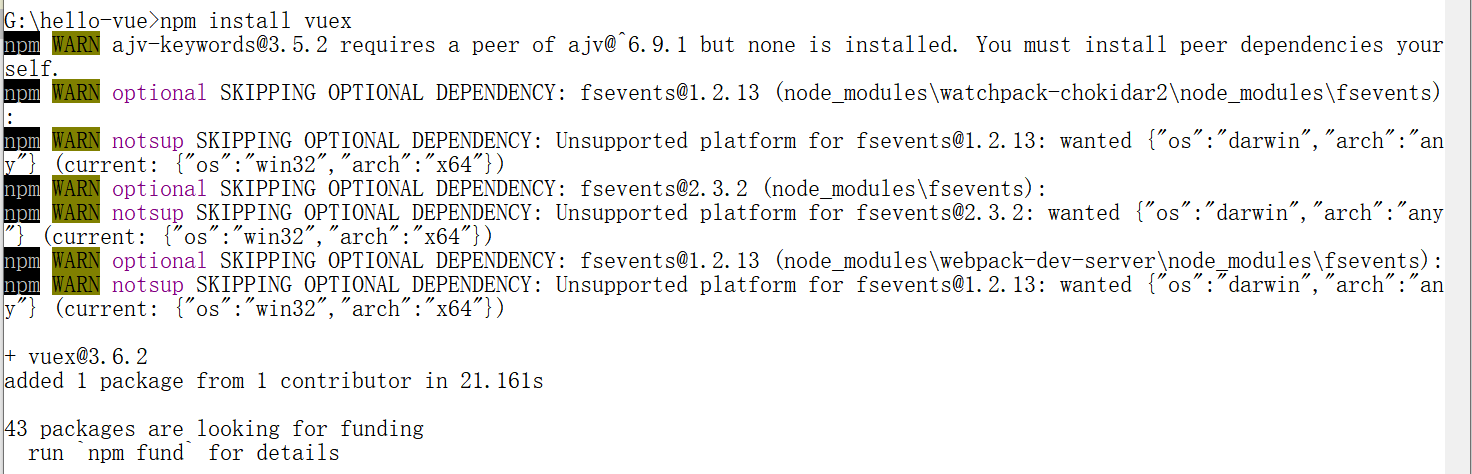
vuex：安装状态管理库，npm install vuex

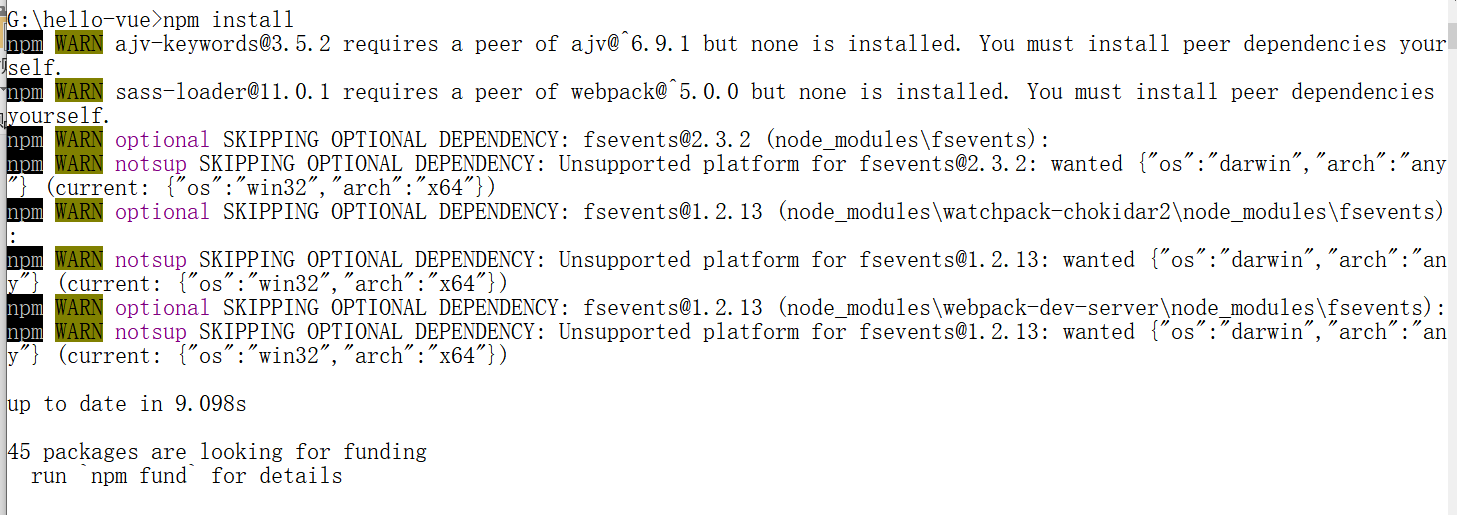
安装依赖：npm install



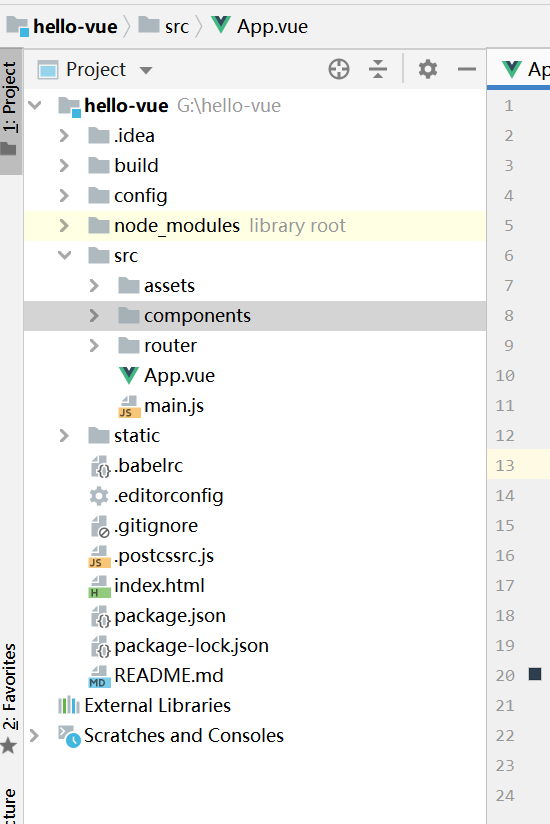








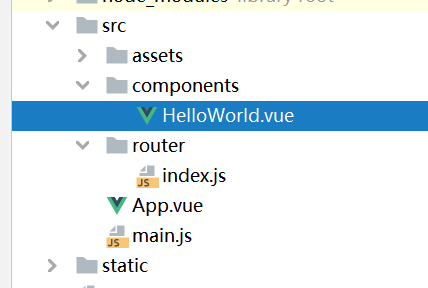
#### 2.1.3利用idea打开项目hello-vue

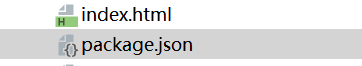


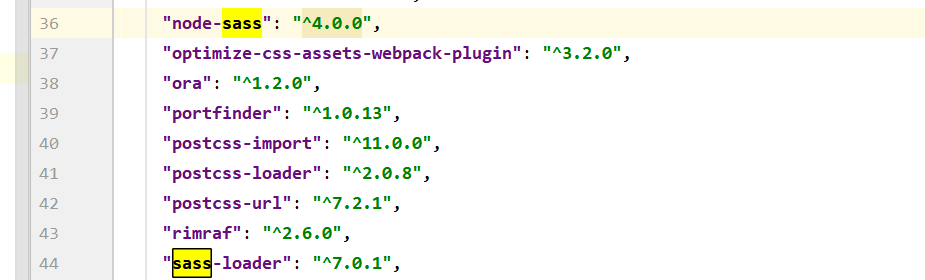
assets:用于存放资源文件

compoents:用于存放Vue组件

router：用于存放vue-router配置

删除所有helloworld.vue组件相关的内容



****

在package.json中修改node-sass和sass-loader的版本为上面的版本

### 2.2编写前端页面：

**registerBal.vue:注册资金**

**LoginTest.vue：登录**

**Menu.vue：导航栏**

**showBalances.vue：显示资金**

**TokenTransfer.vue：转账**

1.编写注册资金页面

详见registerbal.vue组件。

* 1. 界面：



* 1. **与后端通信交互的模块：**

**主要依靠的是axios通信框架。用的是链式的编程形式**

**url：填写后端服务器的地址。http://localhost:9090/mintCoin/registerBalances**

**postData：内容为注册内容。**

**之后进入回调函数，路由转入loginTest组件**

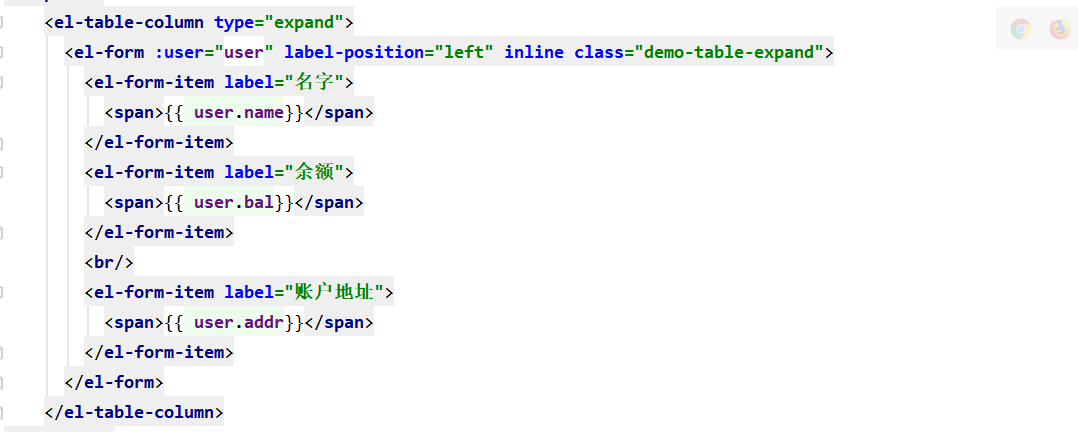


* 1. 注册成功跳到登录页面：LoginTest.vue。





1. 编写查看资金页面
   1. 界面：



* 1. 后端交互：



* 1. 登录之后，点击基本信息，就进入showBalances这个组件,axios发送post请求，会从合约端获取资金数据。



1. 编写转移资金页面
   1. 界面：



* 1. 后端交互：



* 1. 界面显示：

test2注册资金为2000，test1注册资金为1000，

test2向test1转账250

现在test1资金为1250，test2资金为1750









# 总结：项目开发整体流程

1. 在云服务器上利用build\_chian.sh启动一个区块链。
   1. 下载build\_chian.sh
   2. 搭建区块链
   3. 启动节点
2. 在github上下载fisco bcos提供的初始化项目。
   1. idea打开
   2. 根据区块链节点的信息，修改application.yml
   3. 丛云服务器上将相关的证书拷贝到项目的src/main/resources目录下
   4. 在remix里面编写智能合约，编译通过，将其复制到项目中
   5. 利用初始化项目里面的编译函数，将智能合约编译成java代码
   6. 新建目录contract，将编译好的合约复制到这个目录，并编写相关的类去部署和调用合约。
   7. 后面需要与前端交互，与数据库交互，需要在build.gradle引入相关的jar包.
   8. 新建目录controller，编写业务逻辑，一方面与前端交互，另一方面与合约端还有数据库交互。
3. 前端开发，主要利用vue框架，通过axios与后端交互。界面显示用的element-ui组件库。
   1. 以管理员模式打开CMD，下载vue-cli脚手架
   2. 进入前端项目目录，下载aixos，下载element-ui，下载vuex
   3. 在前端项目中，引入axios，elemnet-ui，vuex
   4. 编写前端页面，利用axios与后端交互。

# 附录：

## WINSCP教程:

这里有一个通过SSH连接云服务器的教程。https://blog.csdn.net/swallowblank/article/details/106208941

由于我之前一直用的是密钥文件连接，所以这里我就以之前的密钥文件连接来演示。以从云服务器上下载相关的证书文件为例。

### 1.在云服务器上设置一个密钥文件，保存到本地。按红框走



t-1

t-2

（这里我已经生成过一次，我把之前的删除，然后再创建密钥）



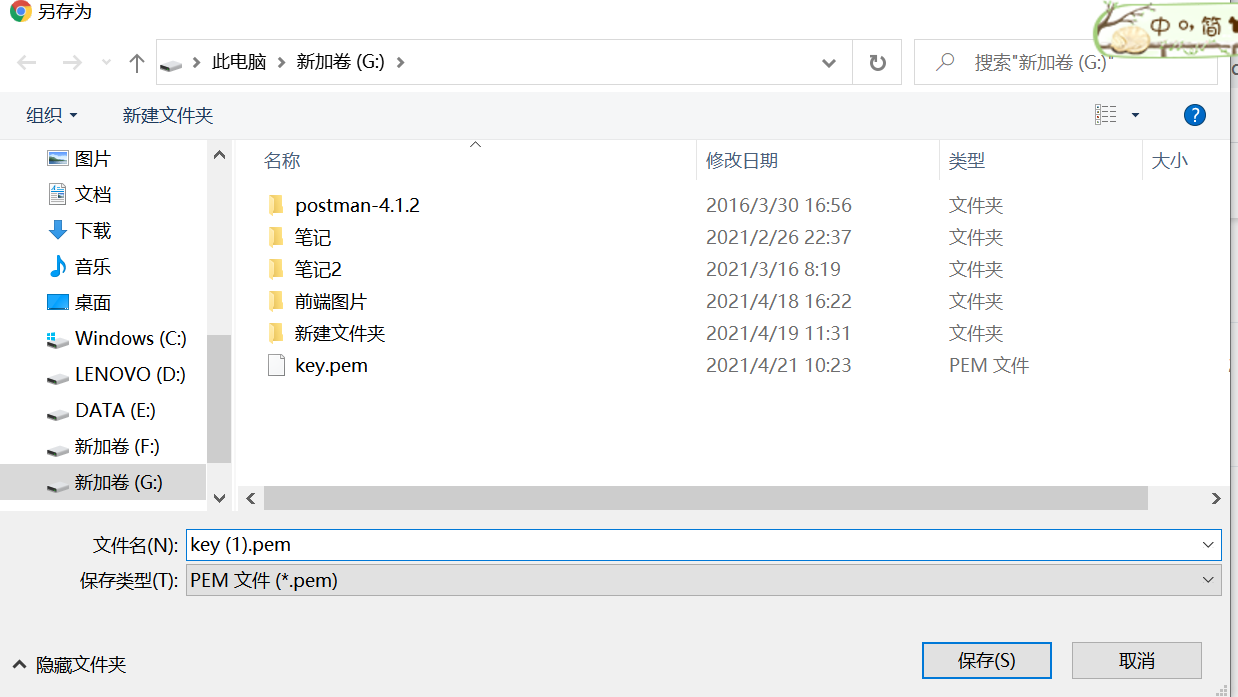
t-3



t-4



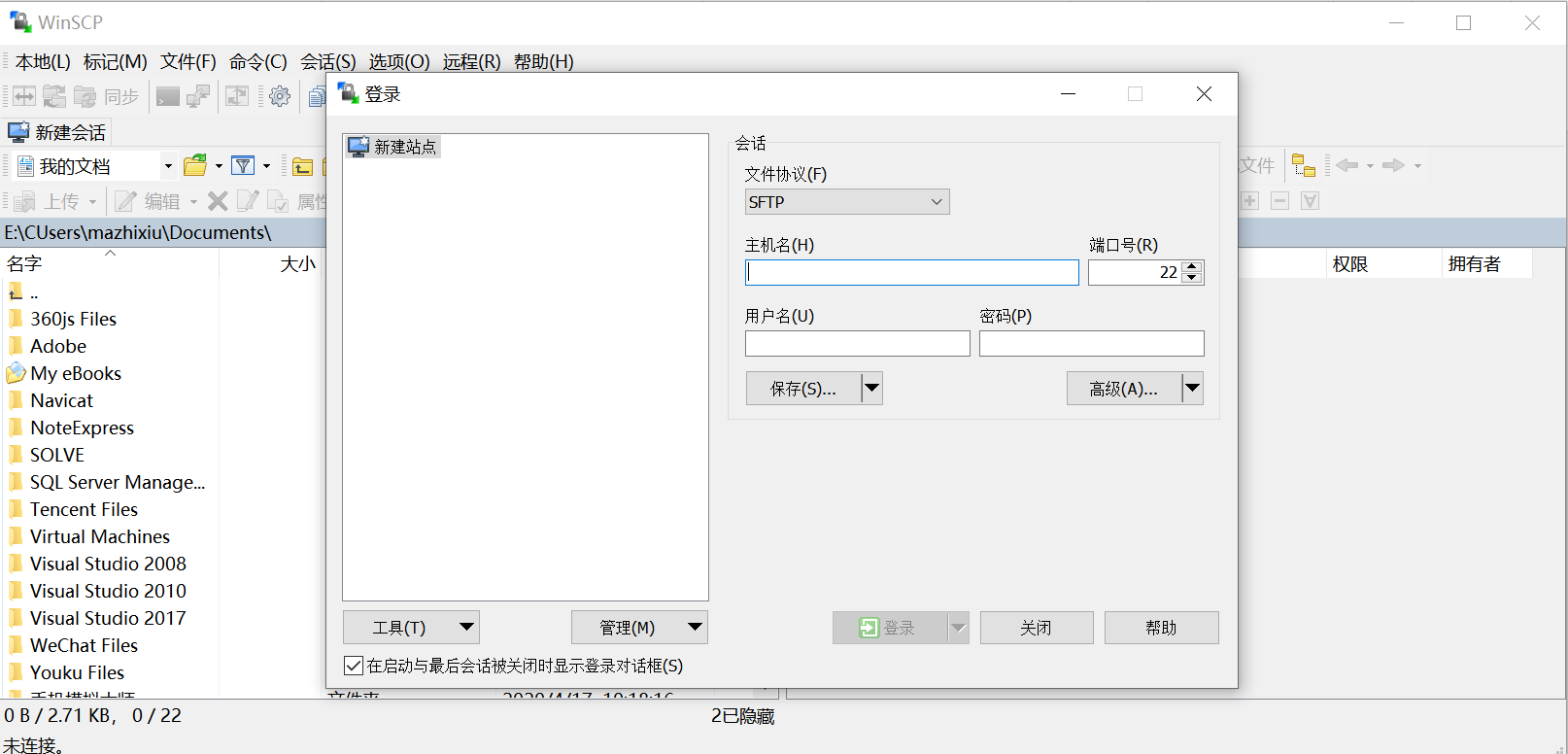
t-5

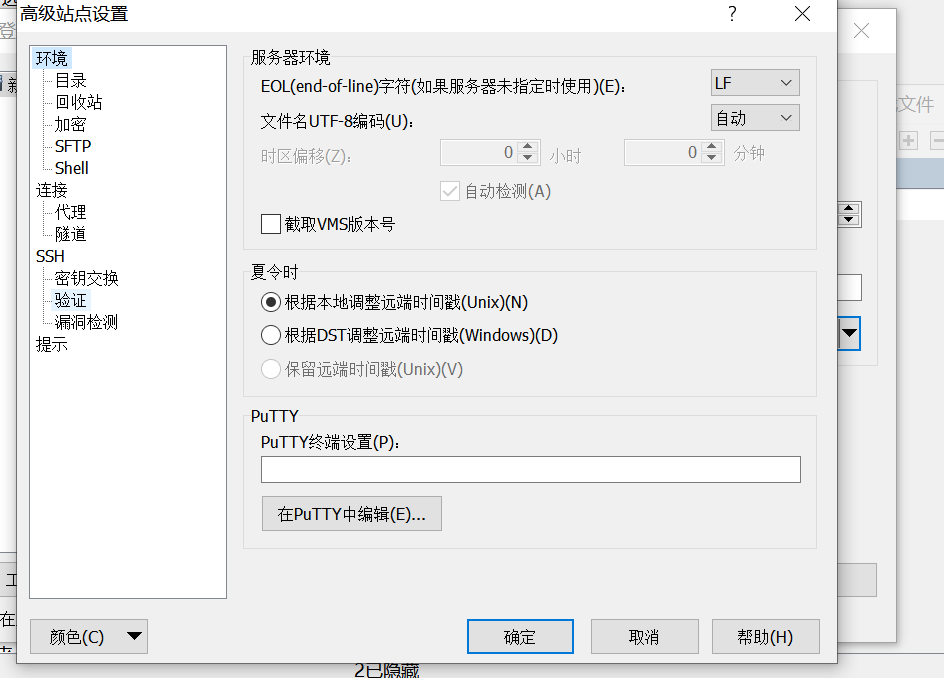


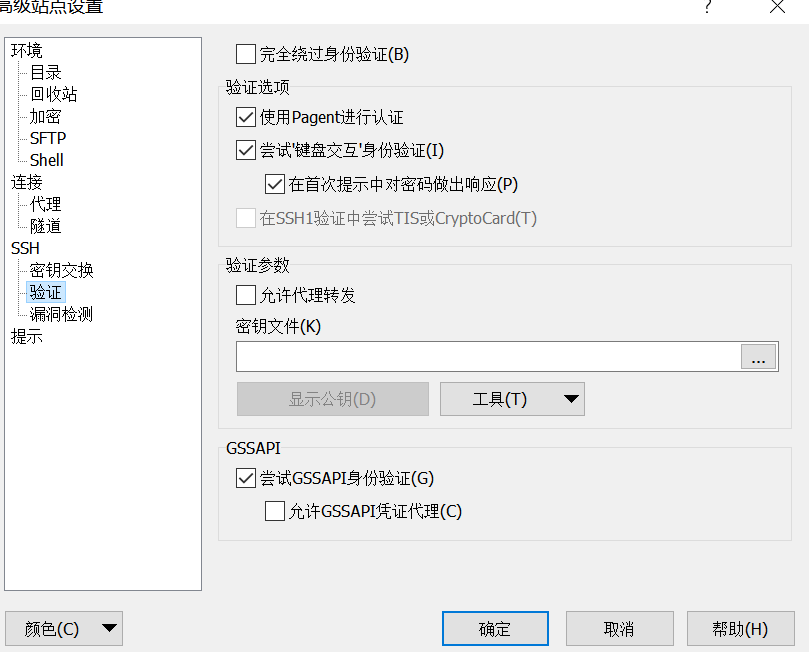


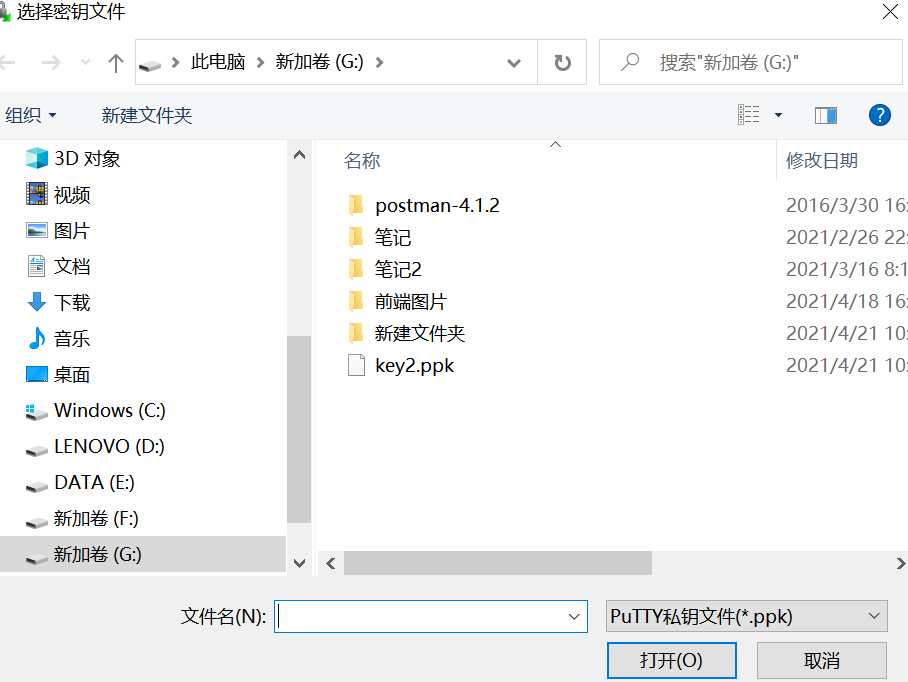
### 2.打开WINSCP软件，按着红框往下走

#### ①选择刚才的保存的.pem形式的密钥转换.ppk形式的密钥

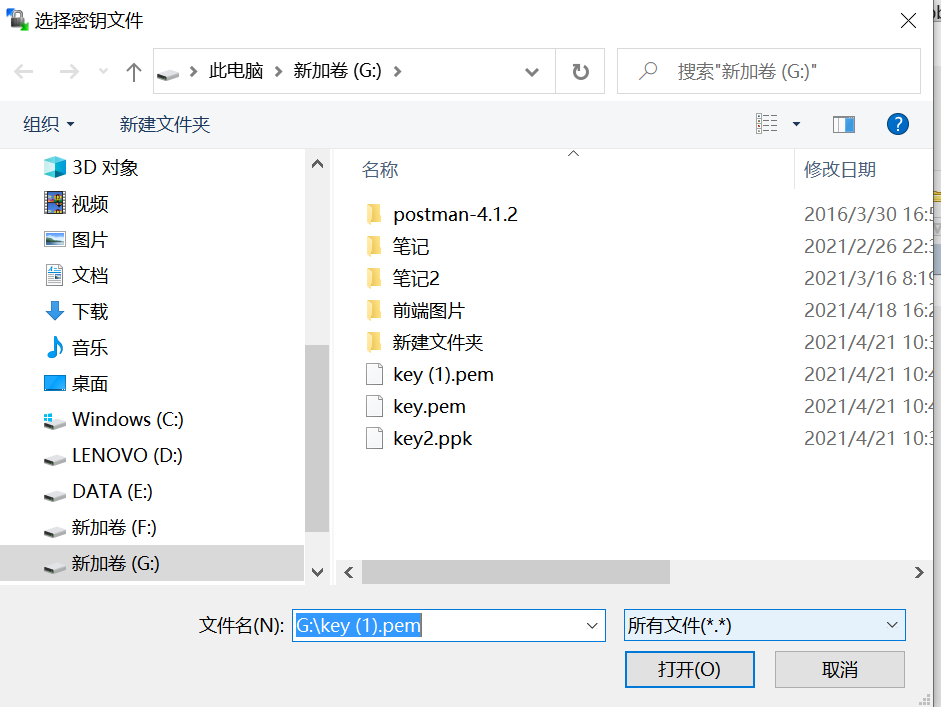


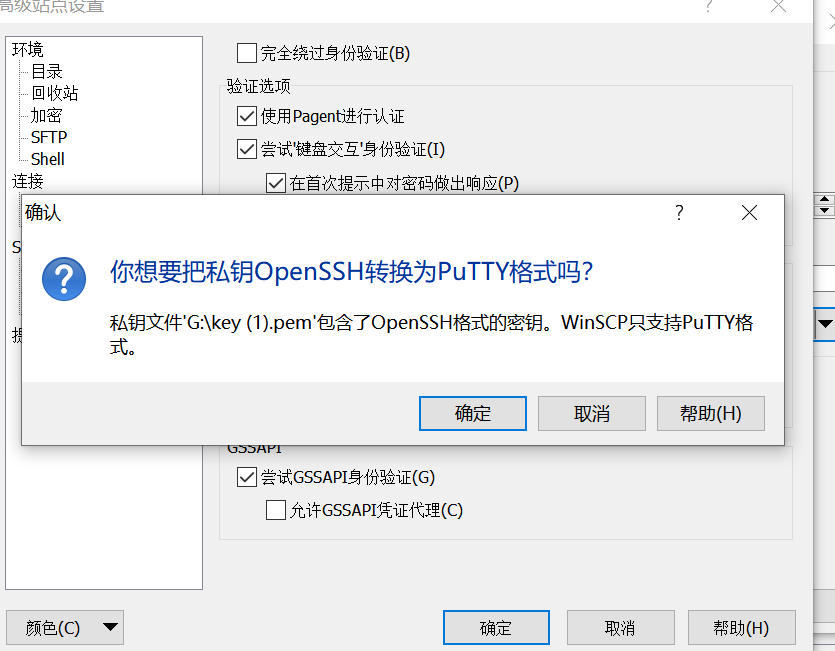




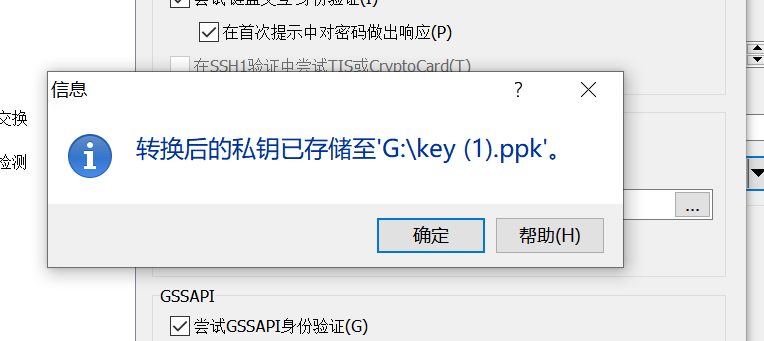


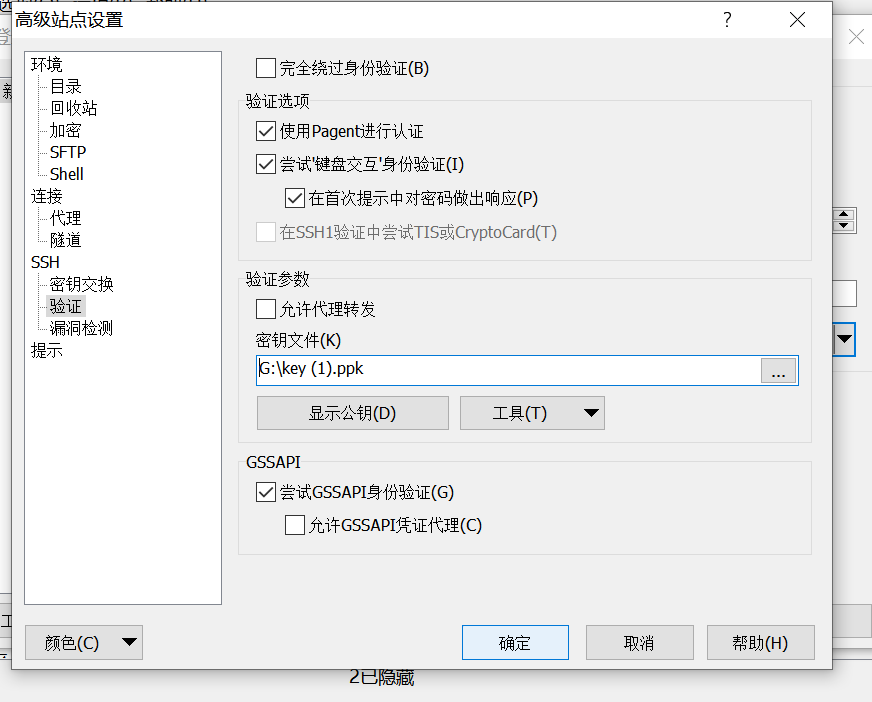




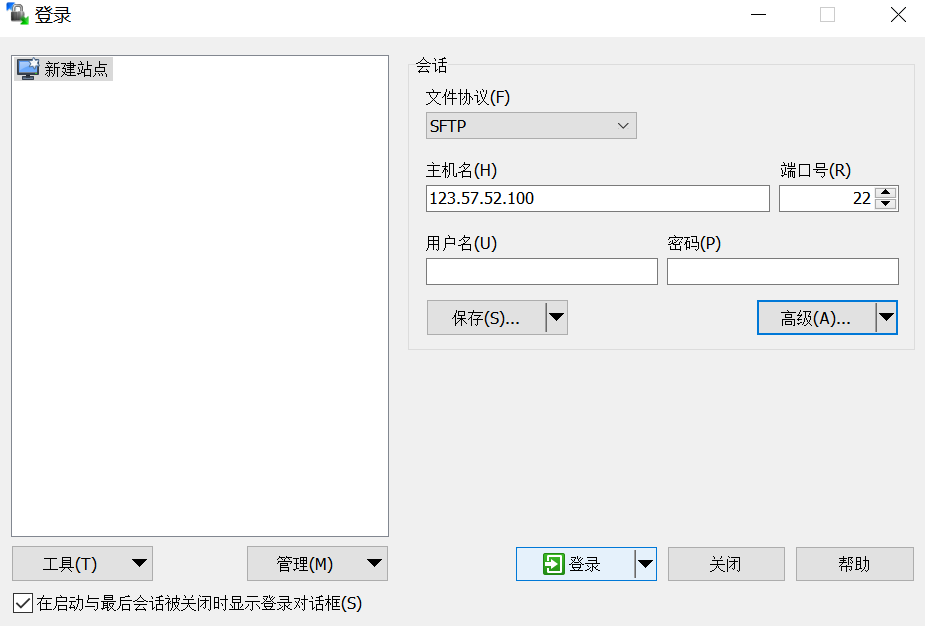


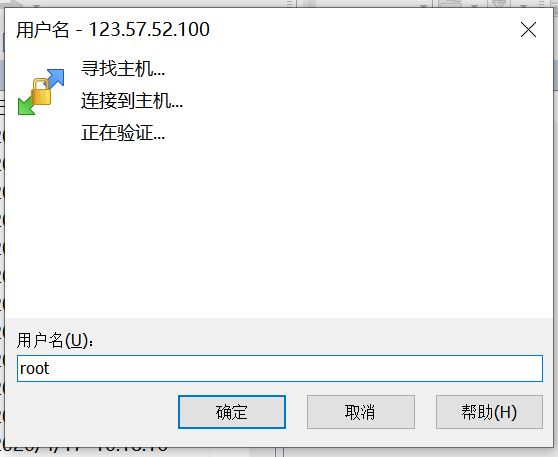




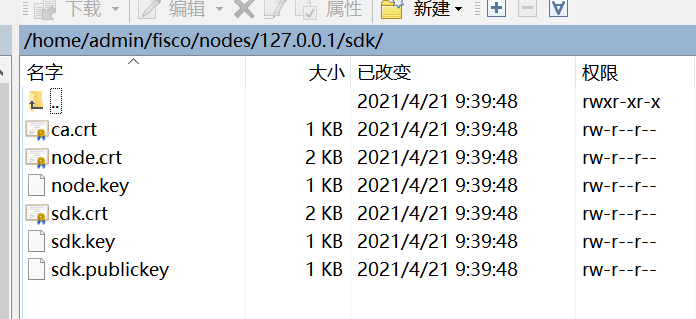


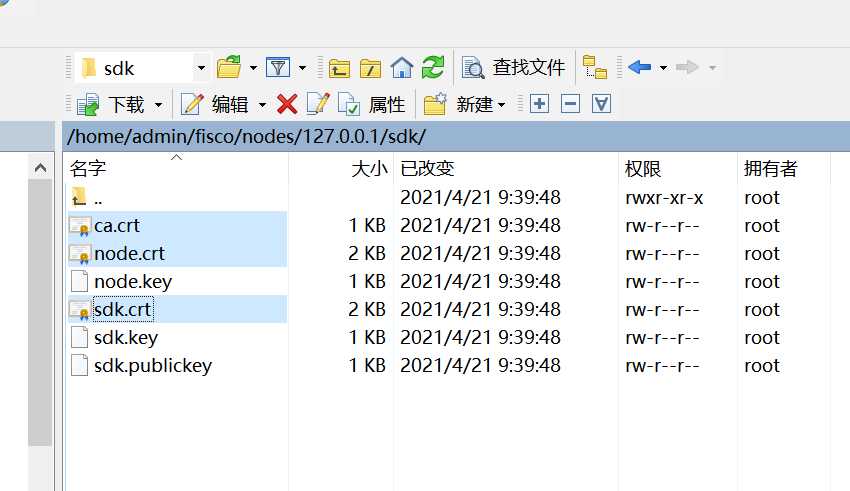
#### ②输入云服务器的IP地址，点击登录，连接上云服务器。

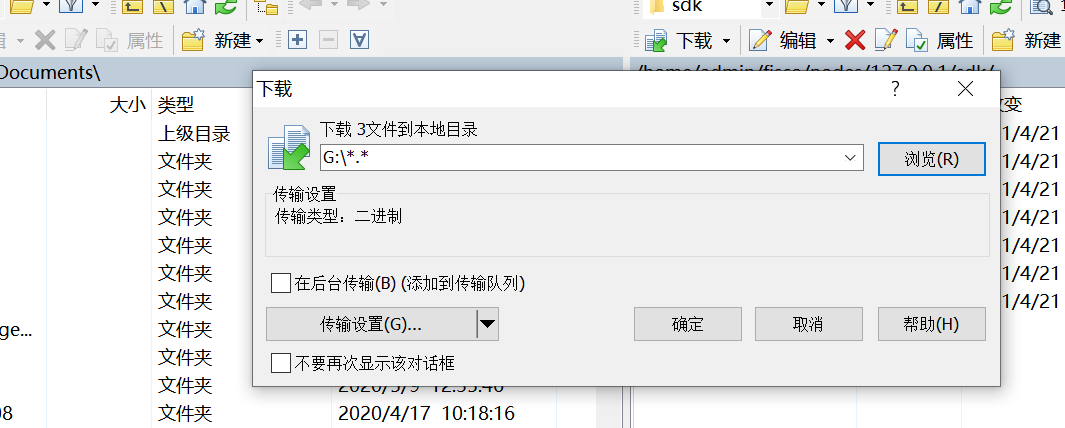




#### ③准备下载相关的证书文件







将G盘下的相关的证书文件拷到项目的src/main/resources/下。

