springboot-2.3.x最新版源码阅读环境搭建-基于gradle构建（全网首发）

码炫课堂技术交流q群：963060292

一、前言

跟很多小伙伴聊天，发现一个严重的问题，很多小伙伴横向发展的貌似很不错，很多技术都能说出一二，但是如果在某个技术上深挖一下就不行了，问啥啥不会。就拿springboot来说，很多同学止步于springboot的应用，再往深处就一问三不知了，那么如何破局呢？smart哥认为最好的办法就是直捣黄龙，要把一个技术理解透了，**听别人讲一万遍原理，不如自己撕一遍源码。**

要阅读源码那就首先得先搭建源码阅读环境，那么本篇文章就来介绍下Spring Boot的源码环境搭建。鉴于spring团队已经全面抛弃maven构建工具而选用gradle来构建，而且网上目前看来还没有文章介绍springboot最新版2.3.x的gradle构建，那么本篇文章就是基于gradle来构建最新版springboot2.3.2的源码阅读环境。

二、环境准备

1、git

拉取源码使用

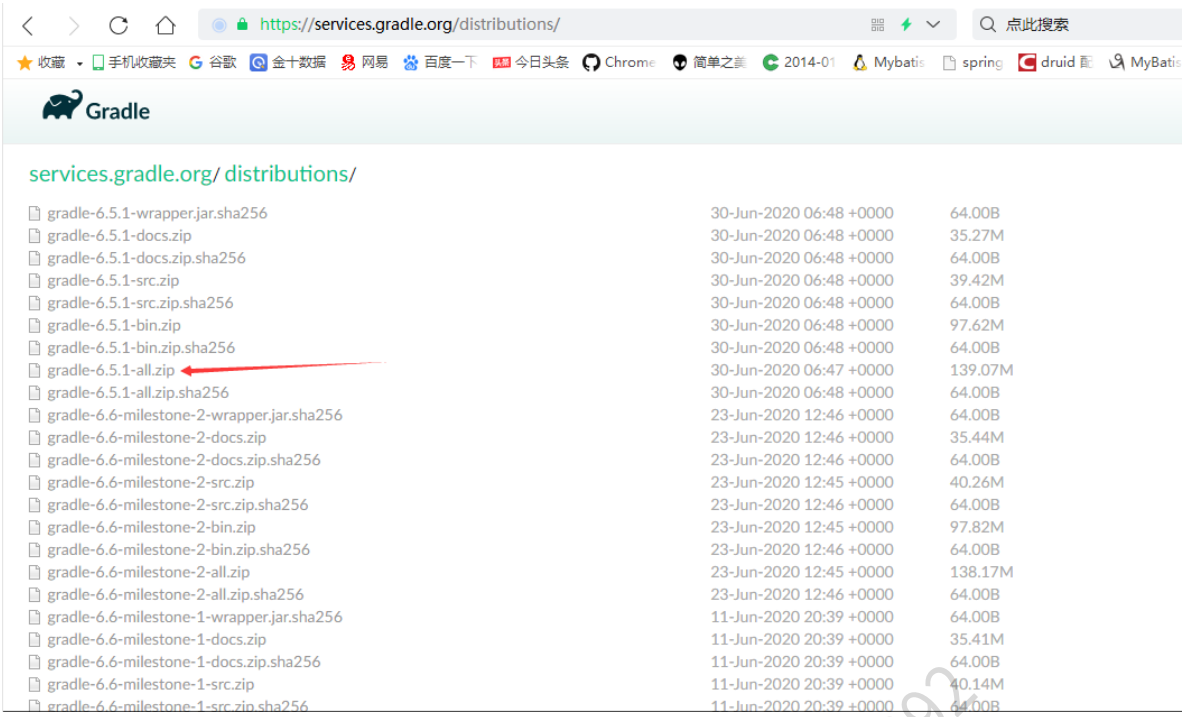
2、jdk8及以上

一般小伙伴机器上都已经装好了

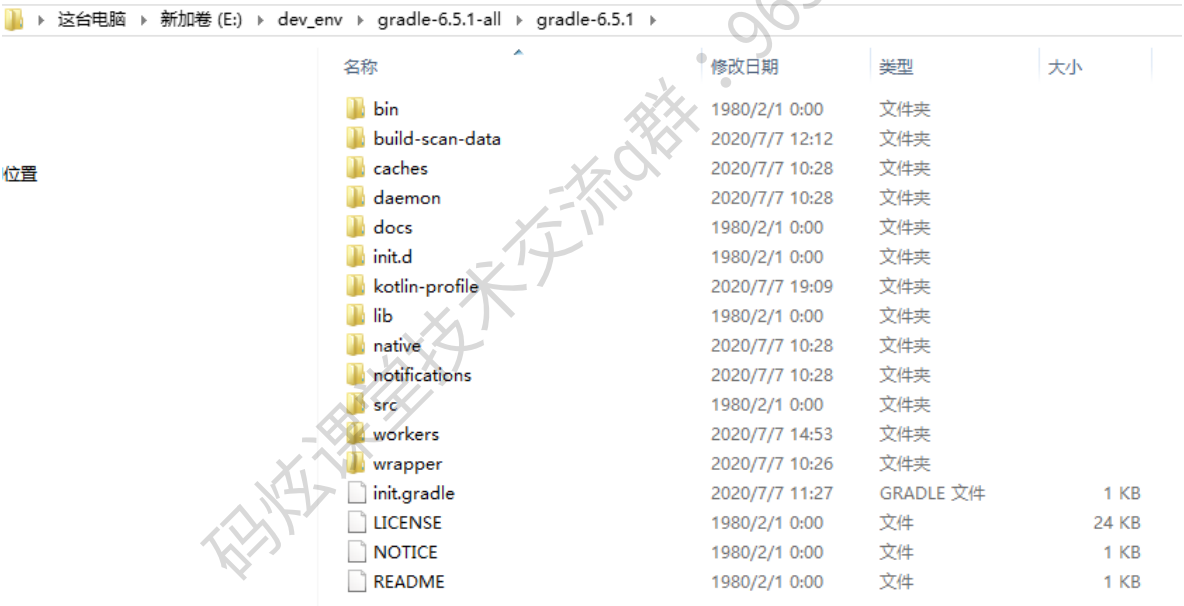
```
E:\mypro\IdeaProjects\spring-boot>java -version
java version "1.8.0_40"
Java(TM) SE Runtime Environment (build 1.8.0_40-b25)
Java HotSpot(TM) 64-Bit Server VM (build 25.40-b25, mixed mode)
```

3、gradle6.5.1

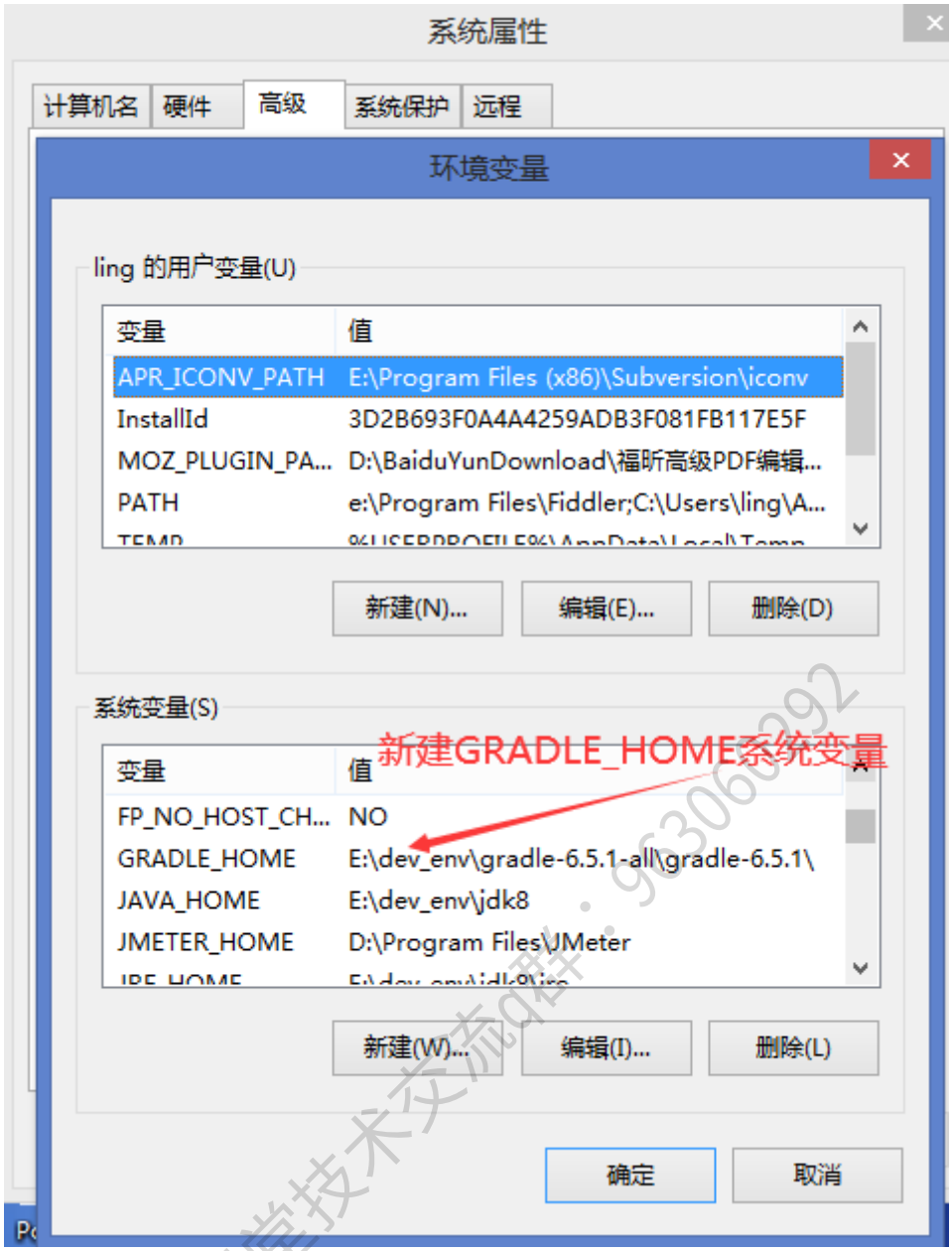
打开 https://services.gradle.org/distributions/ 选择最新版本：gradle-6.5.1-all.zip(all版本是带源码的)



下载解压后目录结构如下：

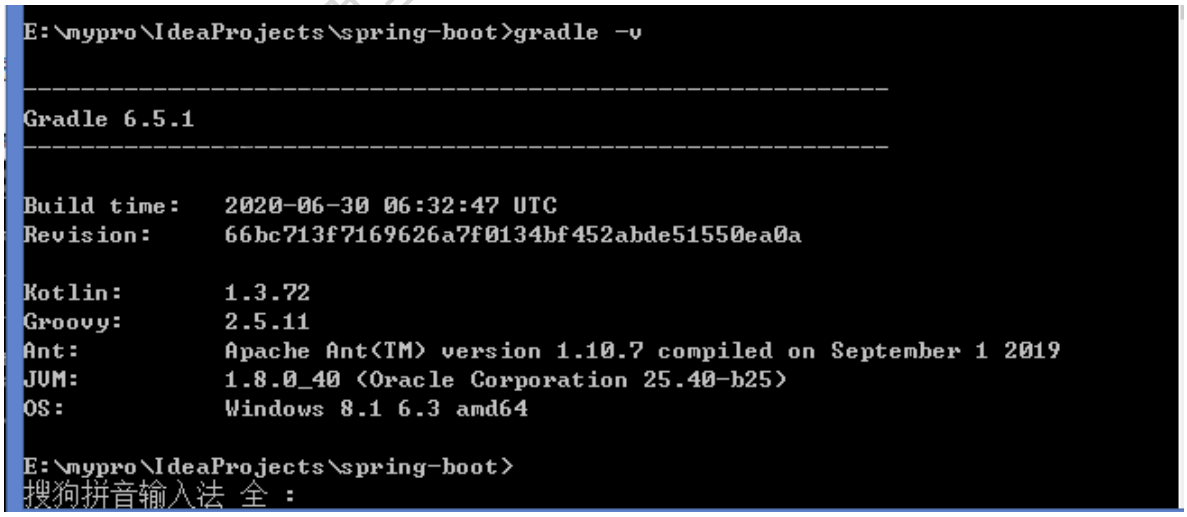


设置环境变量：



完成后打开cmd，执行

```
gradle -v
```



表示已经安装成功，版本为6.5.1

4、idea2020.1.2

（网上很多朋友表示idea2020之前的版本导入时始终有问题，建议升级到2020.1版本，smart哥当前使用的就是2020.1.2版本）

springboot源码构建、编译及冒烟测试会非常的耗内存，建议内存不足16g的小伙伴升级下机器内存，最少16g。

三、下载源码

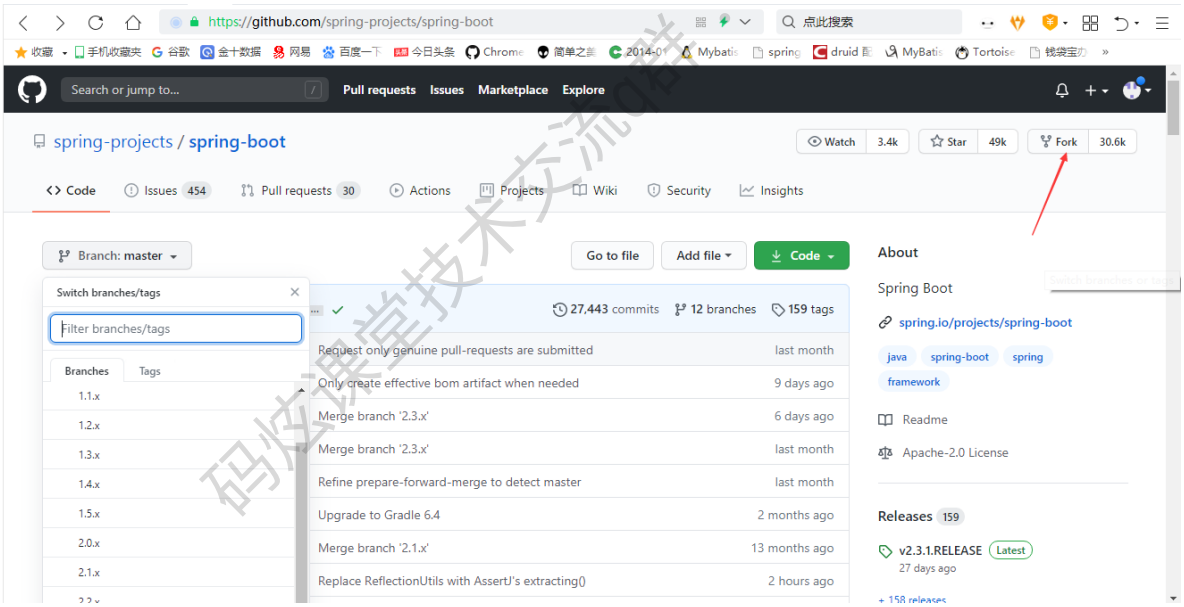
从官方仓库 https://github.com/spring-projects/spring-boot `Fork` 出属于自己的仓库。

- 为什么要 `Fork` ？既然开始阅读、调试源码，我们可能会写一些注释，有了自己的仓库，可以进行自由的提交。
- 本文使用的 Springboot 版本为最新的 `2.3.x` 的分支代码（2.3.2.BUILD-SNAPSHOT）。
- 使用 git 从 `Fork` 出来的仓库拉取代码，注意这里为什么不拉取master分支呢？

  因为smart哥刚开始就是拉取的master分支，但是master分支依赖的spring版本为spring-5.3.0-M1版本，该版本非稳定版本，而且编译到最后会出现问题，报一些spring模块的5.3.0-M1.jar包不存在或无法下载等一些莫名其妙的错误， 所以我这边拉取的是2.3.x分支，这个分支依赖的spring版本为5.2.7.RELEASE版本。所以我就git clone 2.3.x分支到本地，然后再导入idea中。

具体过程如下：

1、打开 https://github.com/spring-projects/spring-boot，点击右上角Fork即可，这样就把spring仓库fork到自己的仓库中了。

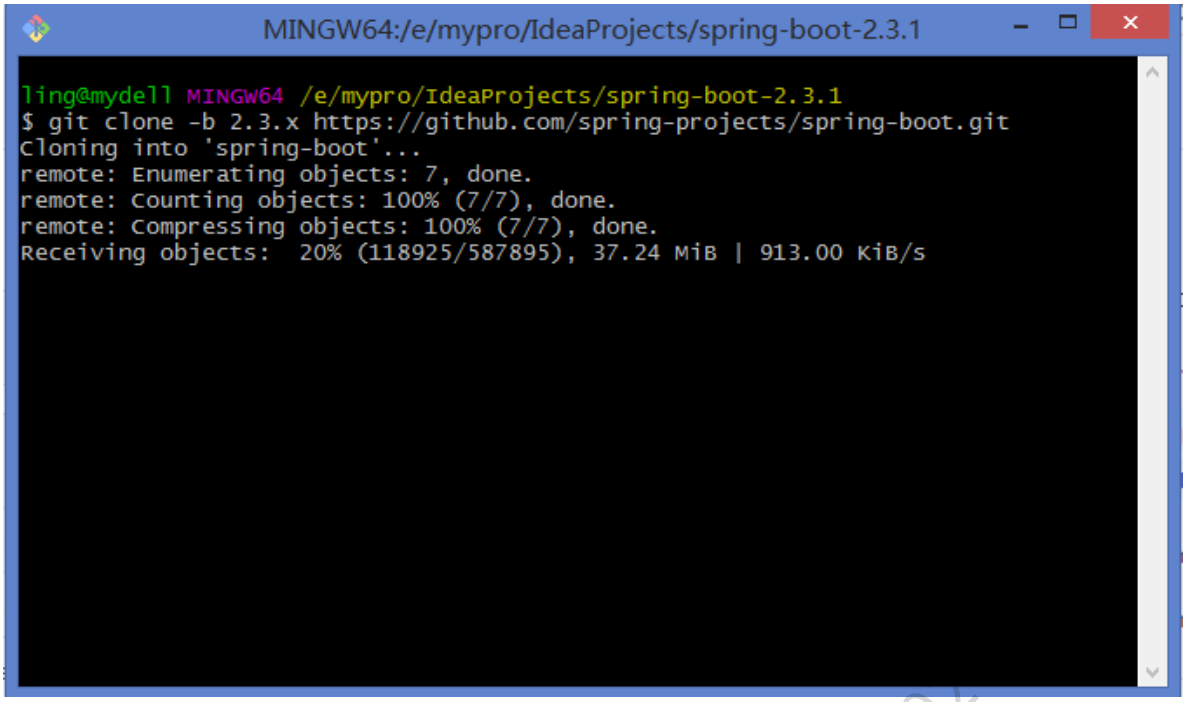


2、选择一个目录，我的是E:\mypro\IdeaProjects\spring-boot-2.3.1，空白处右击Git Bash Here

执行：

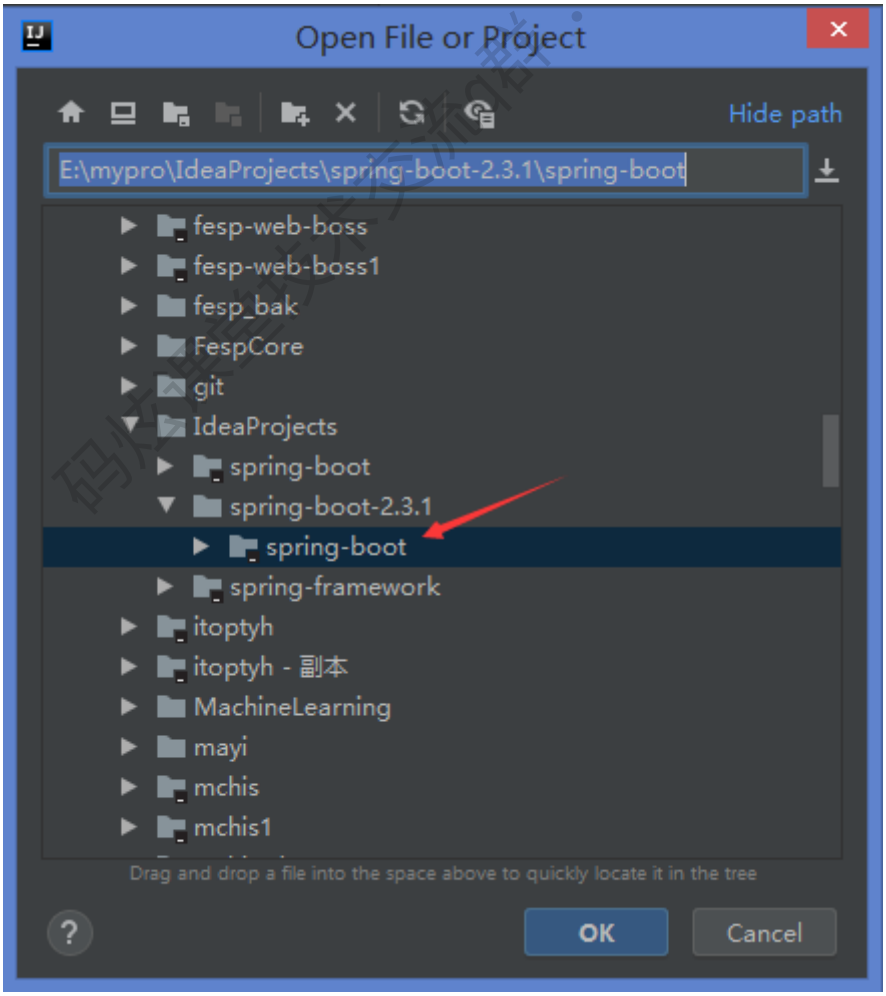
```
git clone -b 2.3.x https://github.com/spring-projects/spring-boot.git
```

下载到本地

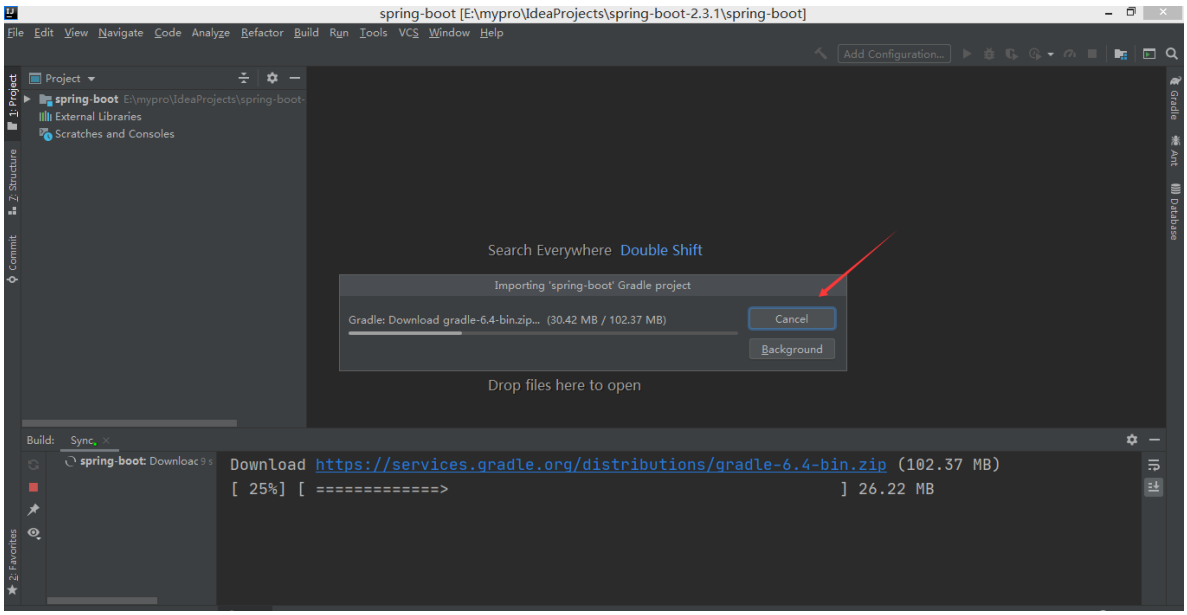


四、开始构建

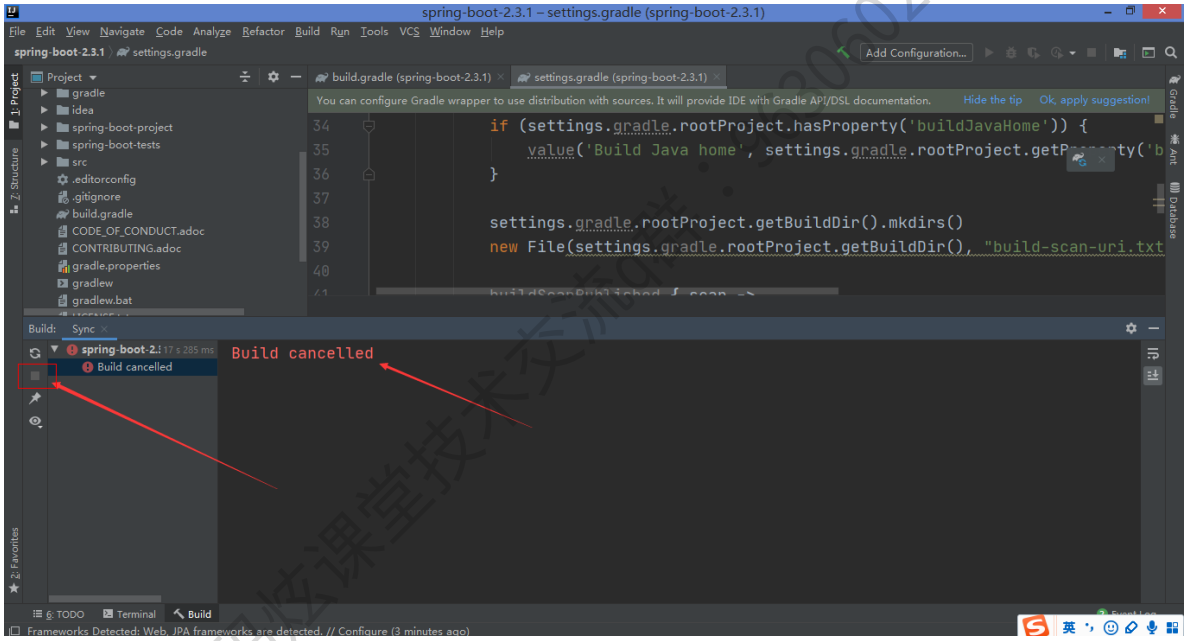
1、打开idea后，【File】->【Open...】,打开刚拉取的spring-boot源码，点击ok即可打开，如下图：



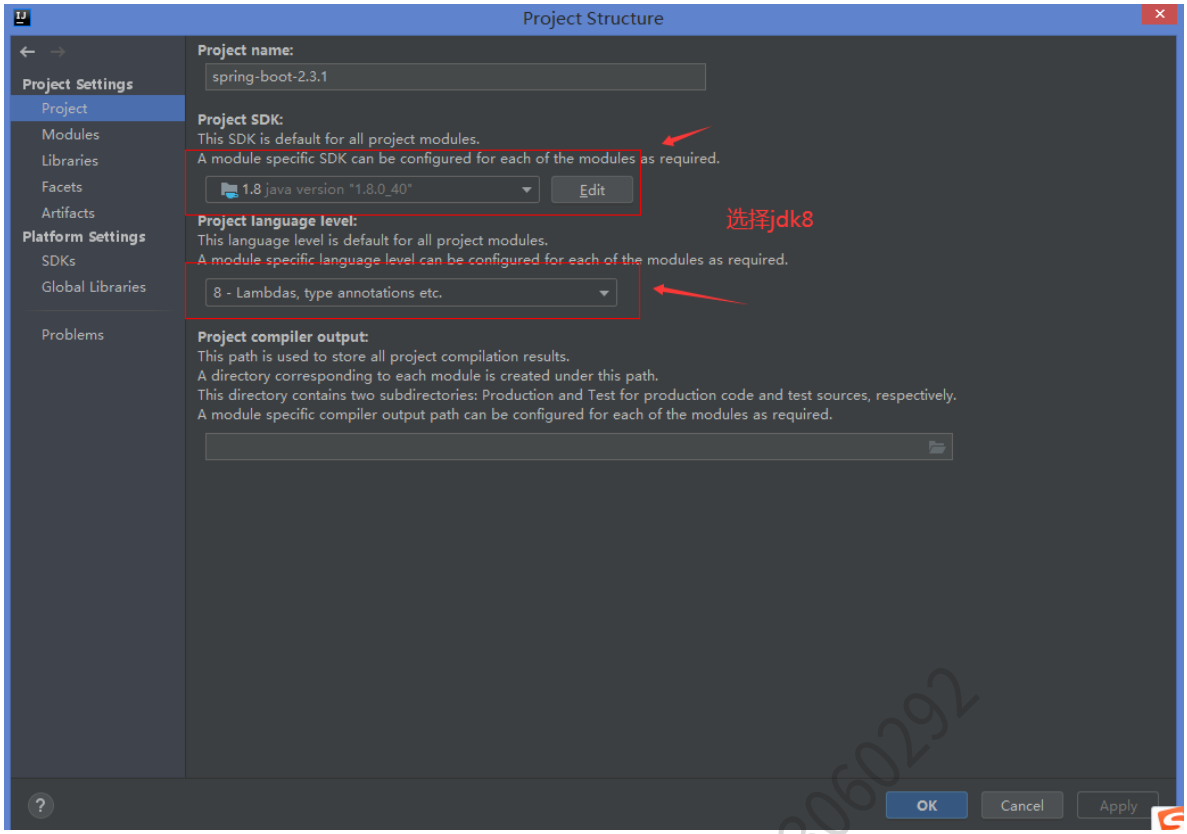
打开之后，gradle会自动构建，开始下载gradle-6.4-bin.zip工具包，idea中还有一些地方需要设置，所以先不构建，点击取消，如下图：



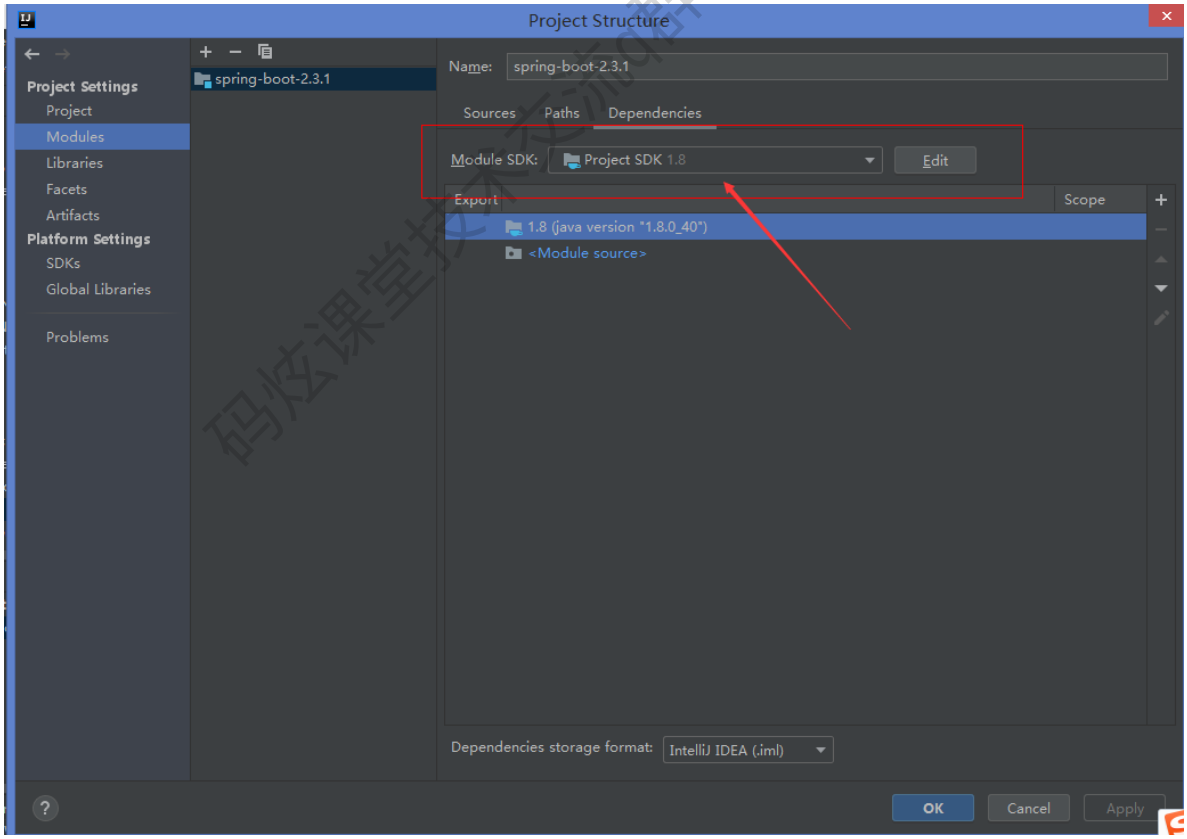
取消后



2、选择【File】->【project Structure...】,打开后点击左侧Project，然后Project SDK选择java version 1.8，Project language level选择8，如下图：



接下来，Modules选择Project SDK 1.8，点击ok即可



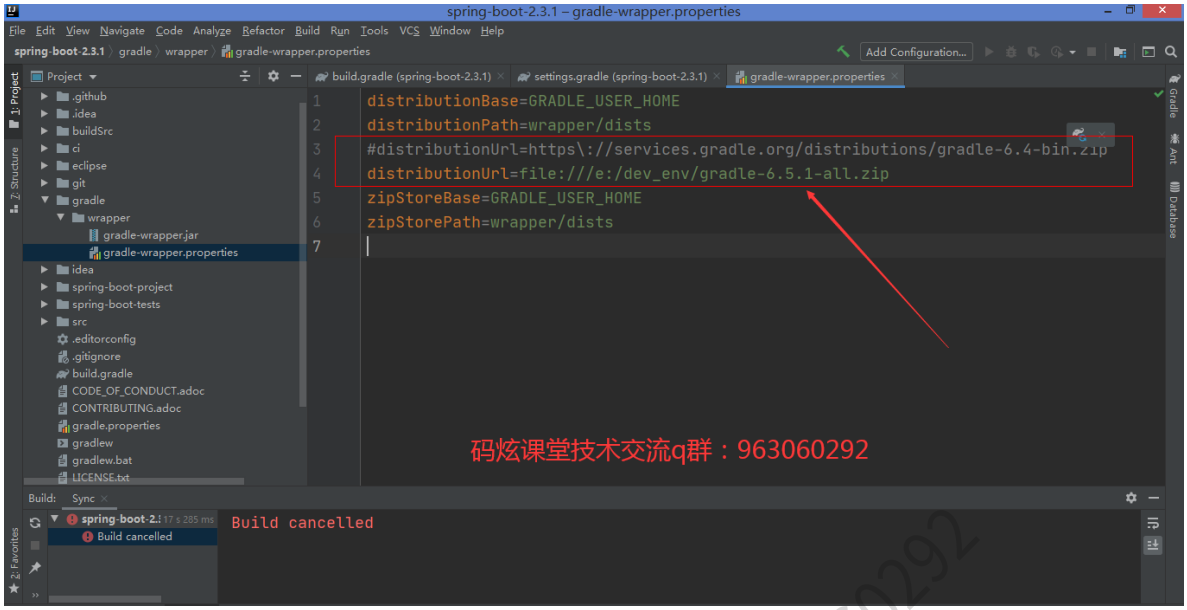
3、设置完毕之后，打开工程下的gradle->wrapper下的gradle-wrapper.properties文件，注释掉：

```
#distributionUrl=https\://services.gradle.org/distributions/gradle-6.4-bin.zip
```

换成本地的gradle-6.5.1-all.zip，这个版本是当前最新版，而且是带源码的。

```
distributionUrl=file:///e:/dev_env/gradle-6.5.1-all.zip
```

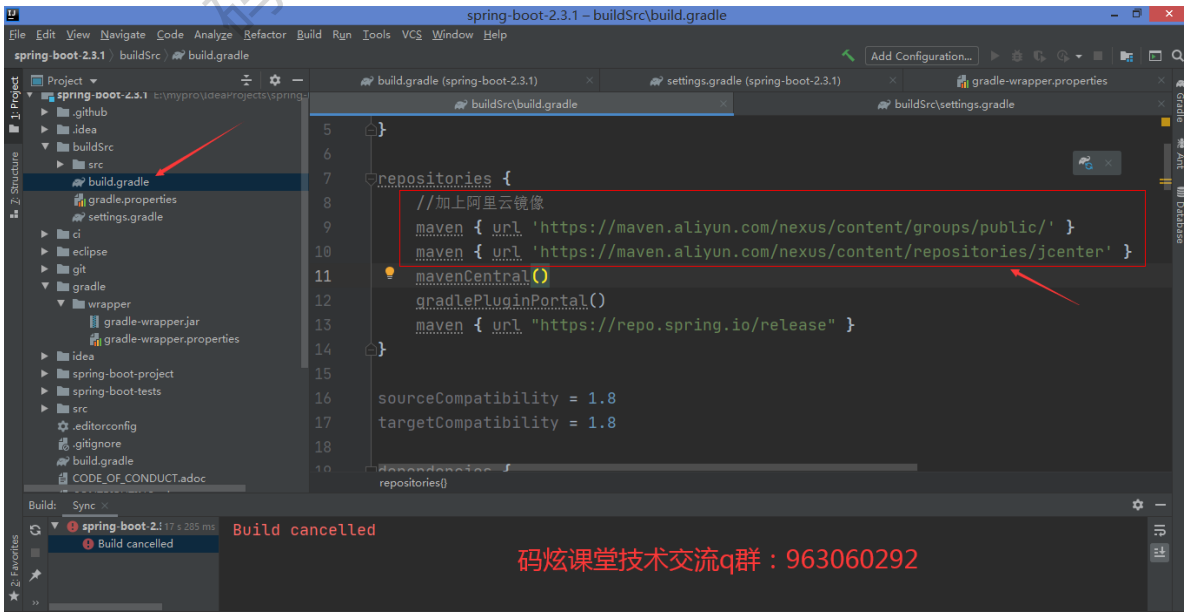
如图所示：



4、修改工程下的buildSrc下的build.gradle文件，找到如下代码段，添加阿里云镜像（不添加的话几个小时也构建不完）

```
repositories {
    //加上阿里云镜像
    maven { url 'https://maven.aliyun.com/nexus/content/groups/public/' }
    maven { url 'https://maven.aliyun.com/nexus/content/repositories/jcenter' }
    maven { url "https://repo.spring.io/plugins-release" }
    mavenCentral()
    gradlePluginPortal()
    maven { url "https://repo.spring.io/release" }
}
```

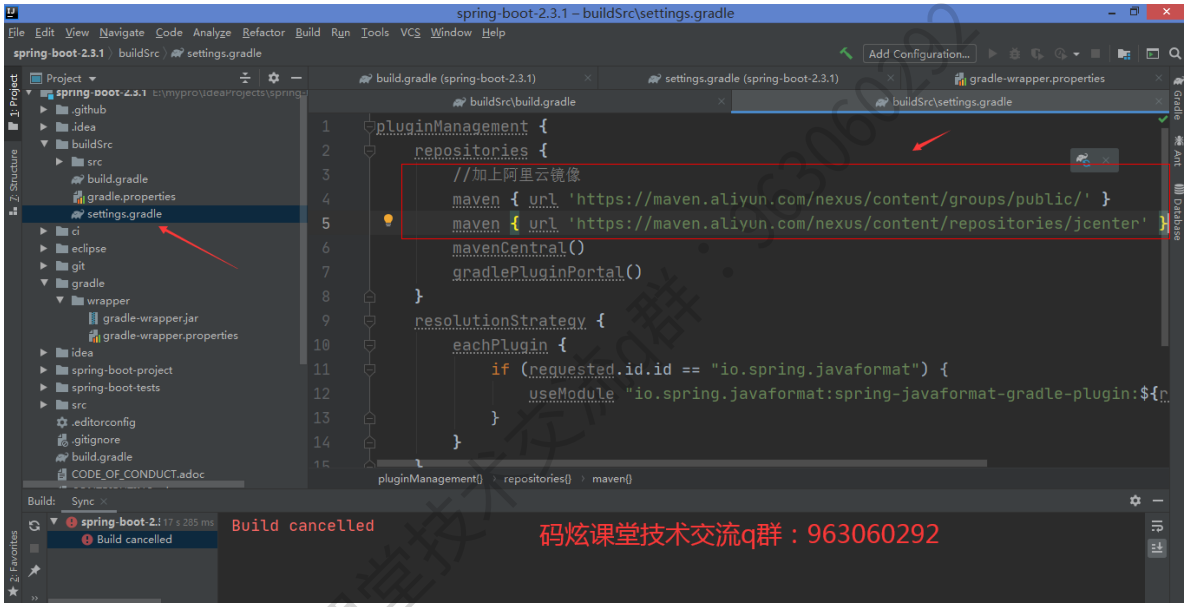
如图：



5、继续修改同目录下的settings.gradle文件，这是全局配置文件，也要加上阿里云镜像，找到如下代码块，修改如下：

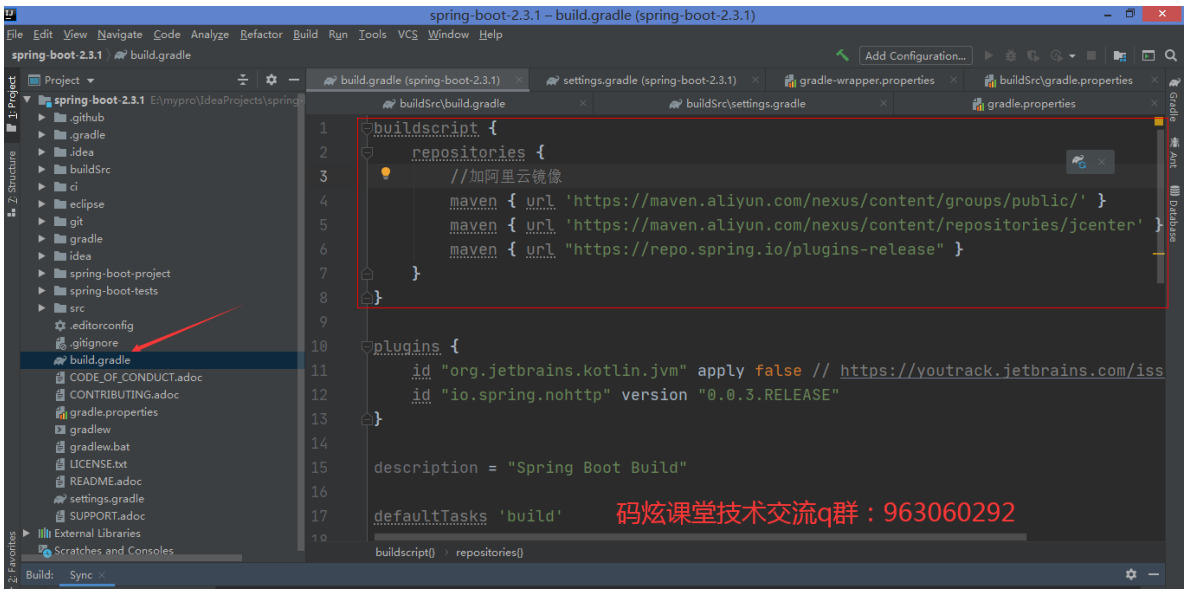
```
pluginManagement {
    repositories {
        //加上阿里云镜像
        maven { url 'https://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'https://maven.aliyun.com/nexus/content/repositories/jcenter'
}
        maven { url "https://repo.spring.io/plugins-release" }
        mavenCentral()
        gradlePluginPortal()
    }
    ......
}
```

如下图所示：



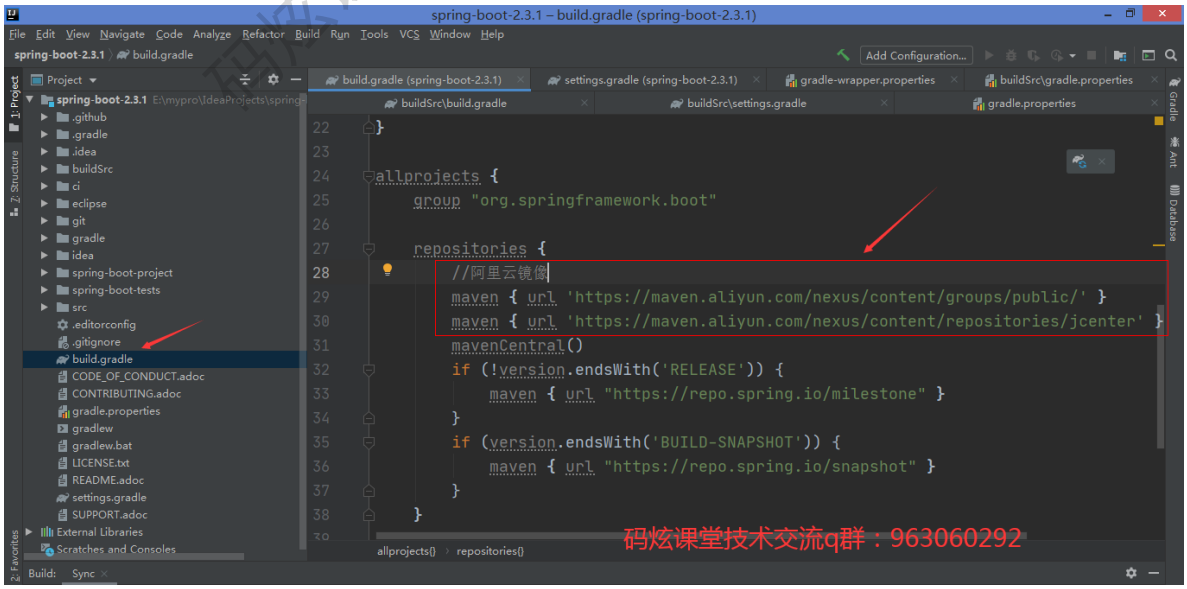
6、修改工程根目录下的build.gradle文件（前面修改的是buildSrc下的，注意区别），同样是加上阿里云镜像，红框中的代码需要全部加上，且只能加在该文件头部。

```
buildscript {
    repositories {
        maven { url 'https://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'https://maven.aliyun.com/nexus/content/repositories/jcenter'
}
        maven { url "https://repo.spring.io/plugins-release" }
    }
}
```



还是这个文件，继续修改，往下找到如下图的代码块，加上阿里云镜像

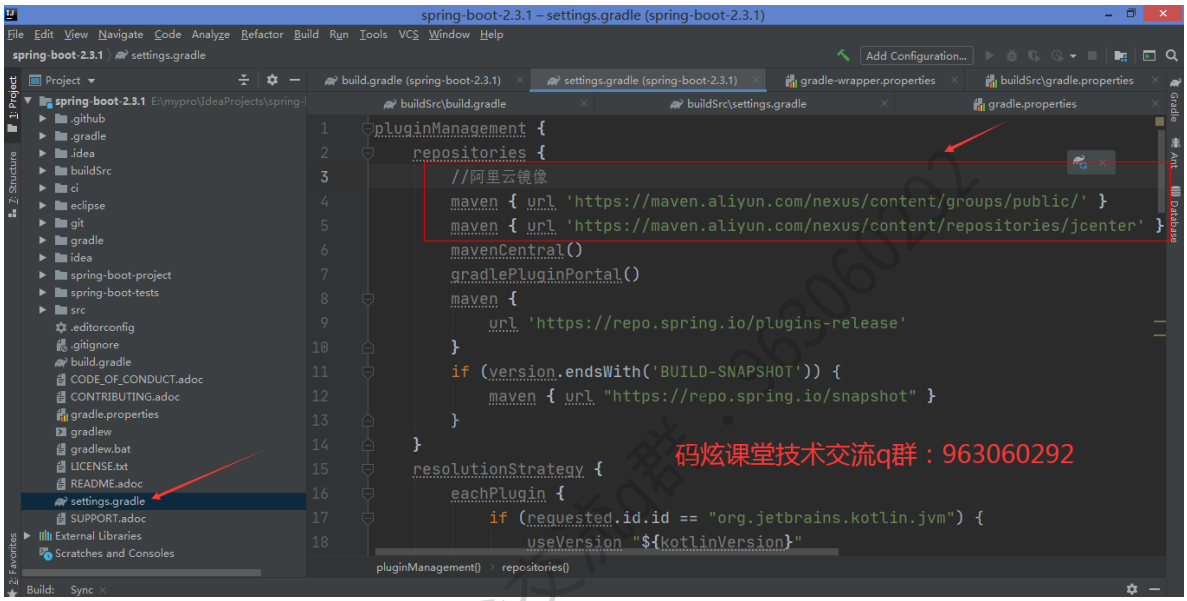
```
allprojects {
    group "org.springframework.boot"

    repositories {
        //阿里云镜像
        maven { url 'https://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'https://maven.aliyun.com/nexus/content/repositories/jcenter'
}
        mavenCentral()
        ......
    }
    ......
}
```

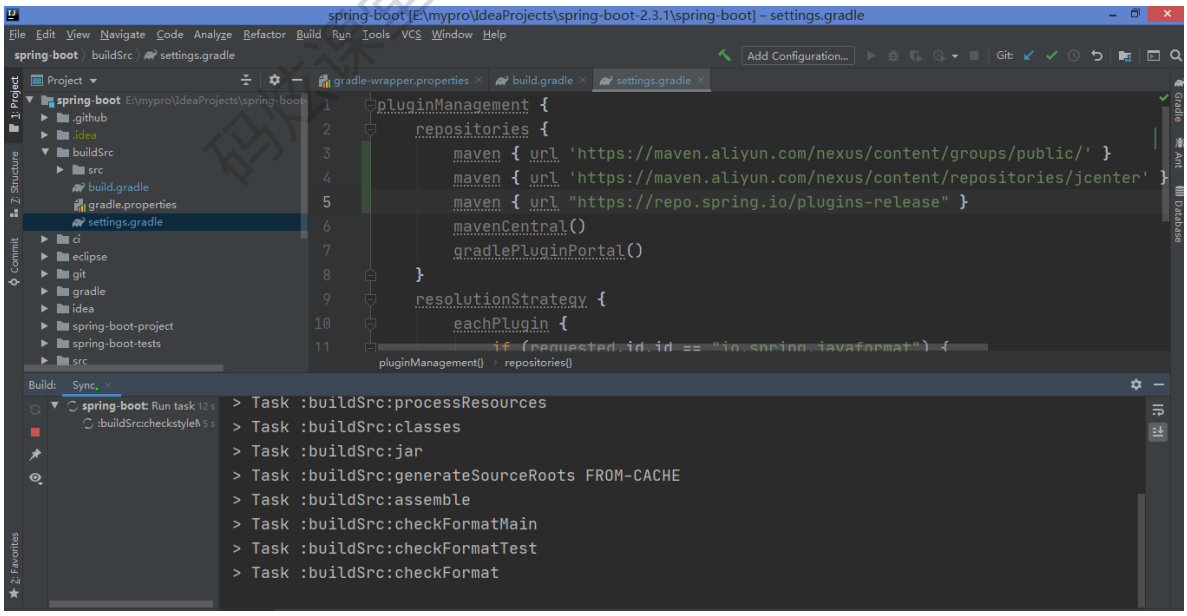
7、继续修改根目录下的全局配置文件settings.gradle，同样是加上阿里云镜像

```
pluginManagement {
    repositories {
        //阿里云镜像
        maven { url 'https://maven.aliyun.com/nexus/content/groups/public/' }
        maven { url 'https://maven.aliyun.com/nexus/content/repositories/jcenter'
}
        mavenCentral()
        ......
    }
    ......
}
```

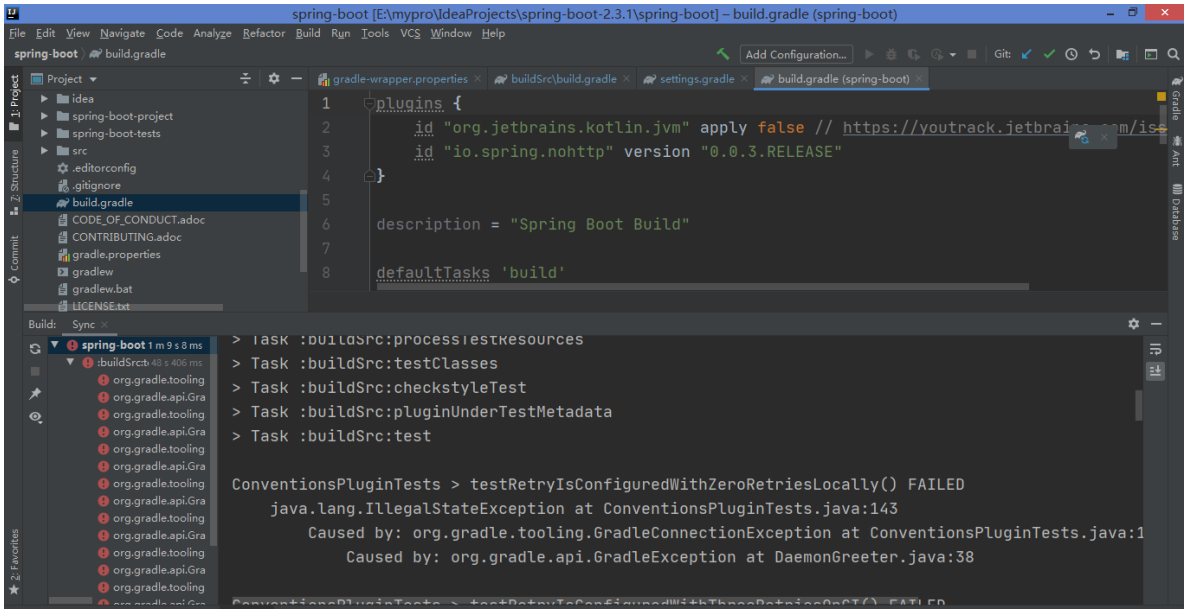
如下图所示：



8、ok，到此才可以开始愉快的构建，如下图，构建中。。。



经过一段时间之后，构建快结束的时候，执行test这一步的时候，出现问题（当然这一步可以省略，但是smart哥先天的强迫症不允许），于是开始破解之法。

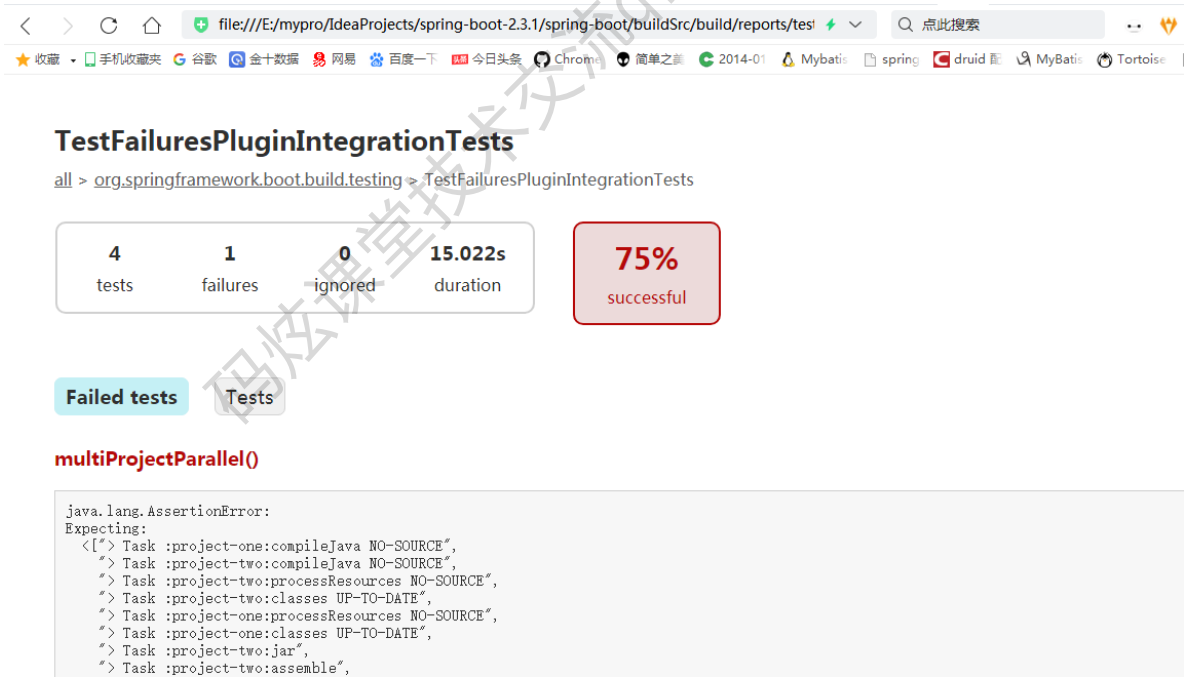


这个问题乍一看是无从入手的，然后往上找到错误提示：

file:///E:/mypro/IdeaProjects/spring-boot-2.3.1/spring-boot/buildSrc/build/reports/tests/test/classes/org.springframework.boot.build.testing.TestFailuresPluginIntegrationTests.html#multiProjectParallel()

于是拷贝这一段地址在浏览器打开，如下图所示：

是一段测试报告



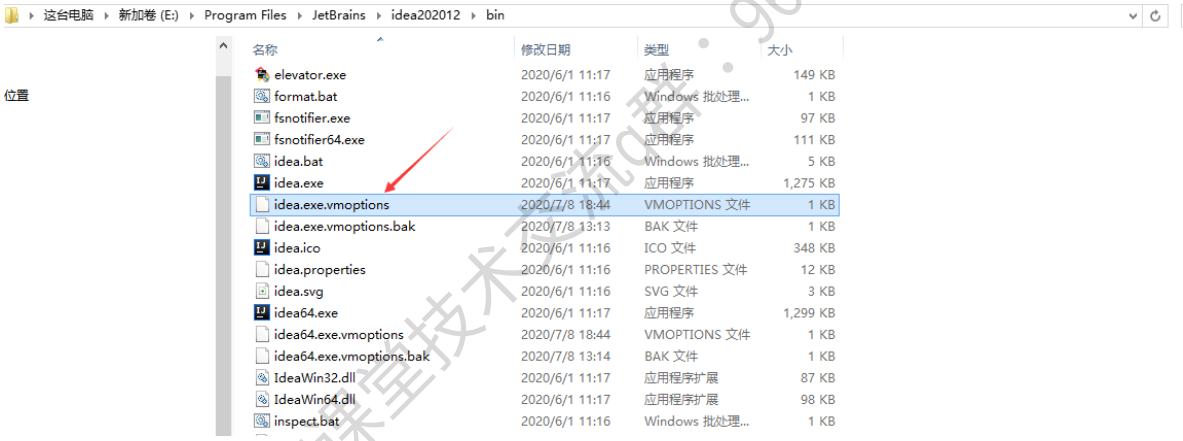
继续往下查找，找到问题关键所在，**heap堆内存没有足够的空间**，错误如下：

```
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)",
    at org.gradle.internal.concurrent.ThreadFactoryImpl$ManagedThreadRunnable.run(ThreadFactoryImpl.java:56)",
    at java.lang.Thread.run(Thread.java:745)",
"Could not write standard input to Gradle Test Executor 3.",
"java.io.IOException: 管道正在被关闭。",
    at java.io.FileOutputStream.writeBytes(Native Method)",
    at java.io.FileOutputStream.write(FileOutputStream.java:326)",
    at java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:82)",
    at java.io.BufferedOutputStream.flush(BufferedOutputStream.java:140)",
    at org.gradle.process.internal.streams.ExecOutputHandleRunner.forwardContent(ExecOutputHandleRunner.java:68)",
    at org.gradle.process.internal.streams.ExecOutputHandleRunner.run(ExecOutputHandleRunner.java:53)",
    at org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:64)",
    at org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:48)",
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)",
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)",
    at org.gradle.internal.concurrent.ThreadFactoryImpl$ManagedThreadRunnable.run(ThreadFactoryImpl.java:56)",
    at java.lang.Thread.run(Thread.java:745)",
"",
"> Task :project-one:test FAILED",
"Error occurred during initialization of VM",
"Could not reserve enough space for object heap",
"Error occurred during initialization of VM",
"Could not reserve enough space for object heap",
"Process 'Gradle Test Executor 3' finished with non-zero exit value 1",
"org.gradle.process.internal.ExecException: Process 'Gradle Test Executor 3' finished with non-zero exit value 1",
    at org.gradle.process.internal.DefaultExecHandle$ExecResultImpl.assertNormalExitValue(DefaultExecHandle.java:417)",
    at org.gradle.process.internal.worker.DefaultWorkerProcess.onProcessStop(DefaultWorkerProcess.java:141)",
    at org.gradle.process.internal.worker.DefaultWorkerProcess.access$000(DefaultWorkerProcess.java:42)",
    at org.gradle.process.internal.worker.DefaultWorkerProcess$1.executionFinished(DefaultWorkerProcess.java:94)",
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)",
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)",
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)",
    at java.lang.reflect.Method.invoke(Method.java:497)",
    at org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:36)",
```

找到了问题关键，heap堆内存没有足够的空间

9、我本机是16g内存，于是修改idea的bin目录下的idea.exe.vmoptions文件，修改idea的初始化内存和最大内存参数：

-Xms1024m，-Xmx4096m

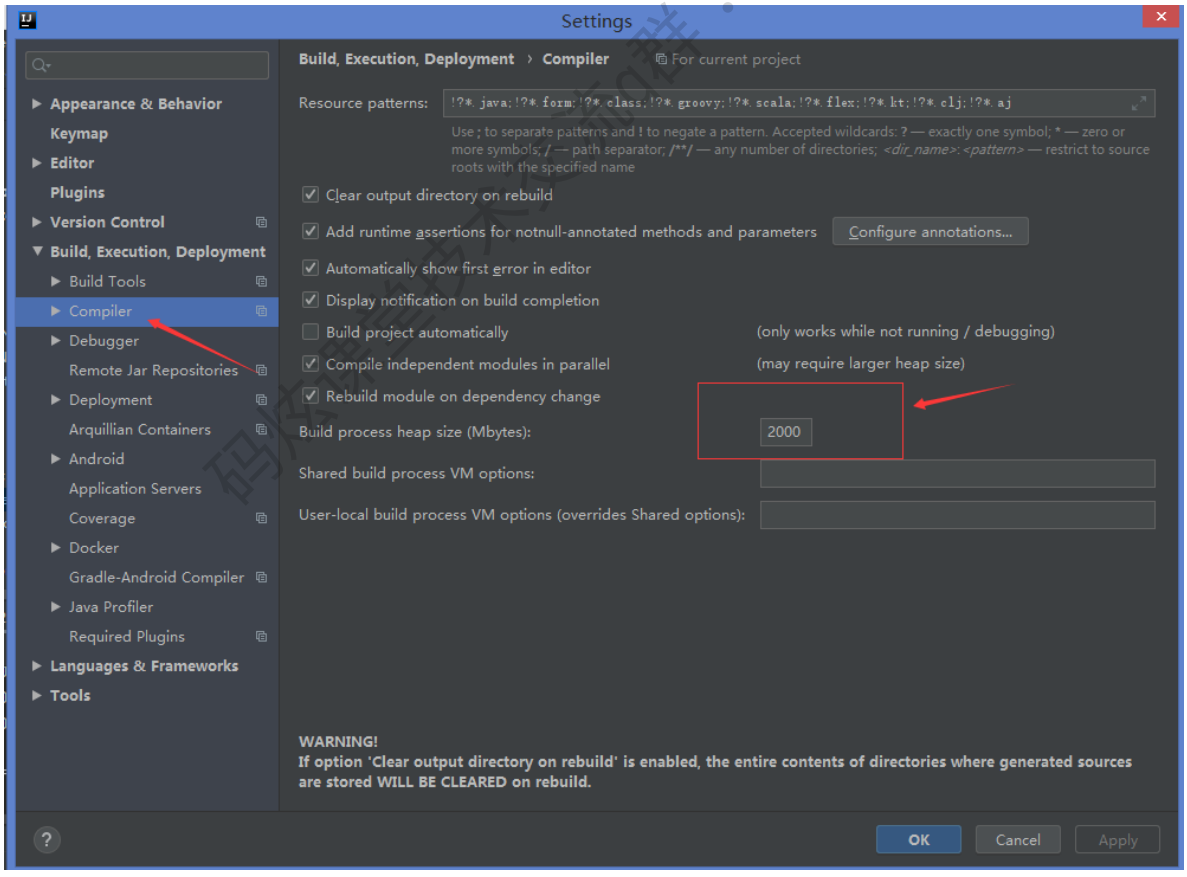



```
1  -server
2  -Xms1024m
3  -Xmx4096m
4  -XX:ReservedCodeCacheSize=500m
5  -XX:+UseConcMarkSweepGC
6  -XX:SoftRefLRUPolicyMSPerMB=50
7  -ea
8  -XX:CICompilerCount=2
9  -Dsun.io.useCanonPrefixCache=false
10 -Djdk.http.auth.tunneling.disabledSchemes=""
11 -XX:+HeapDumpOnOutOfMemoryError
12 -XX:-OmitStackTraceInFastThrow
13 -Djdk.attach.allowAttachSelf=true
14 -Dkotlinx.coroutines.debug=off
15 -Djdk.module.illegalAccess.silent=true
16 -Dfile.encoding=UTF-8
17
```

修改后的文件内容如下：

```
-server
-Xms1024m
-Xmx4096m
-XX:ReservedCodeCacheSize=500m
-XX:+UseConcMarkSweepGC
-XX:SoftRefLRUPolicyMSPerMB=50
-ea
-XX:CICompilerCount=2
-Dsun.io.useCanonPrefixCache=false
-Djdk.http.auth.tunneling.disabledSchemes=""
-XX:+HeapDumpOnOutOfMemoryError
-XX:-OmitStackTraceInFastThrow
-Djdk.attach.allowAttachSelf=true
-Dkotlinx.coroutines.debug=off
-Djdk.module.illegalAccess.silent=true
-Dfile.encoding=UTF-8
```

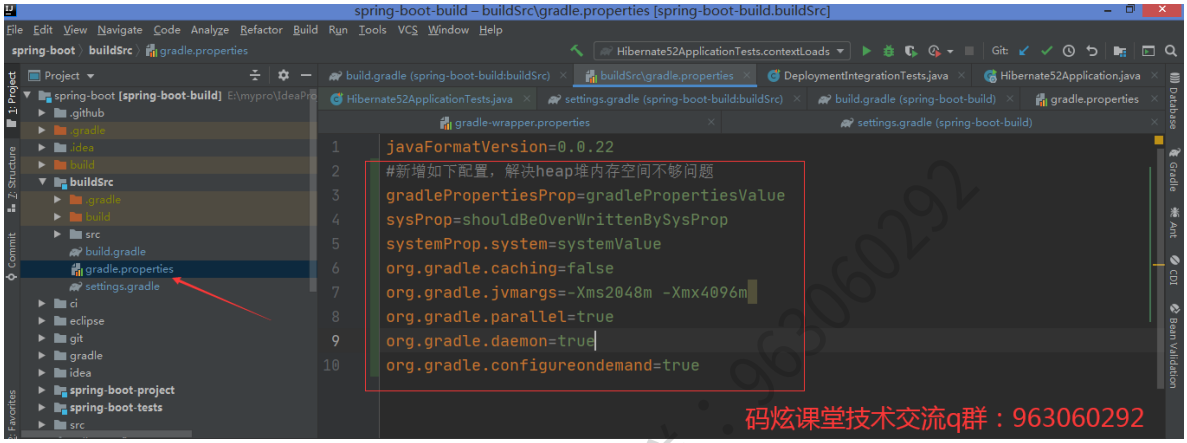
10、重新编译还是报错，于是打开【File】->【Settings】，【Build，Execution，Deployment】->【Compiler】，Build process heap size（Mbytes）改成2000



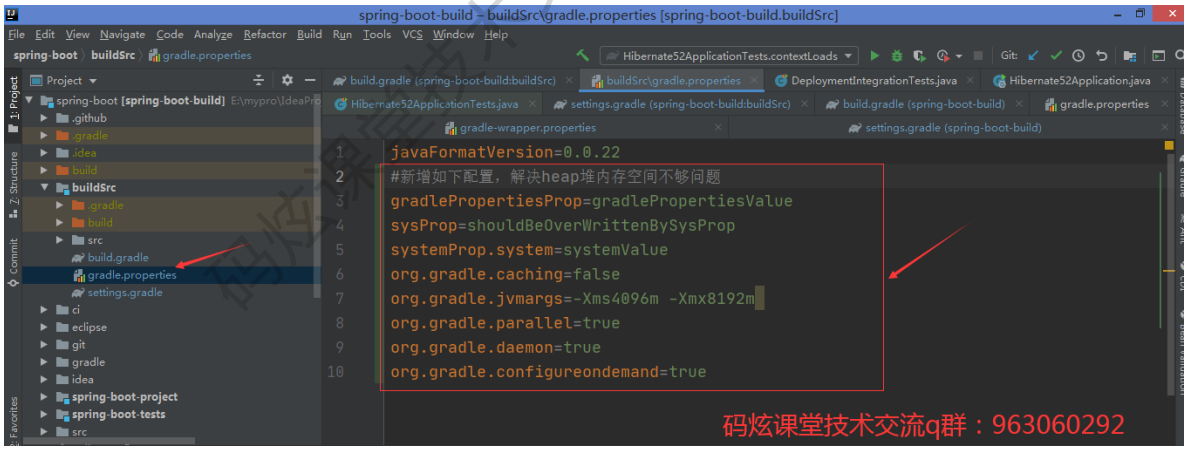
11、重新编译还是报错，继续修改buildSrc目录下的gradle.properties文件，新增如下配置：

```
#新增如下配置，解决heap堆内存空间不够问题
gradlePropertiesProp=gradlePropertiesValue
sysProp=shouldBeOverWrittenBySysProp
systemProp.system=systemValue
org.gradle.caching=false
org.gradle.jvmargs=-Xms2048m -Xmx4096m
org.gradle.parallel=true
org.gradle.daemon=true
org.gradle.configureondemand=true
```

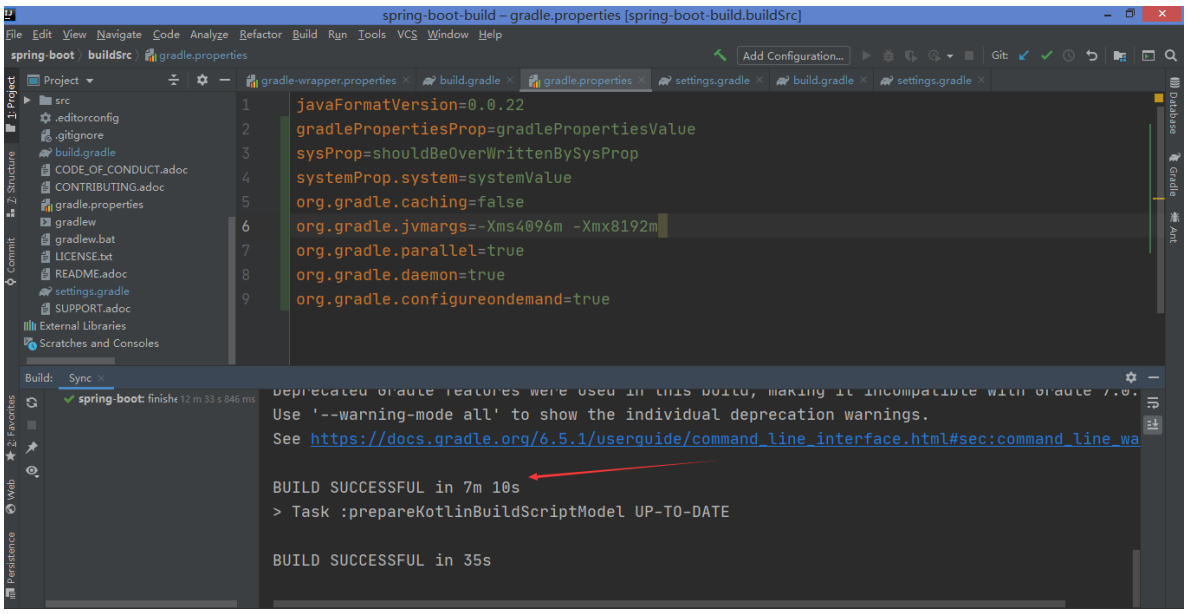
重点是：**org.gradle.jvmargs=-Xms2048m -Xmx4096m**



配置完重新构建，编译，最后还是报空间不足，于是将org.gradle.jvmargs内存扩大一倍，如下：**初始4g，最大8g**



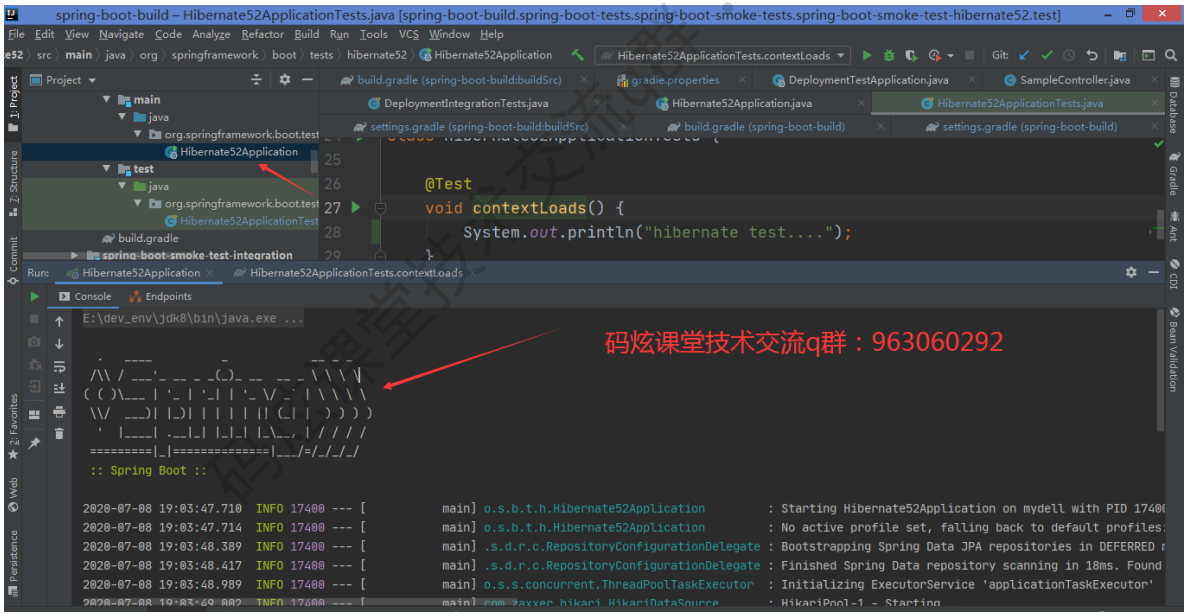
修改完再次重新构建，编译，终于成功！！撒花！！7分10秒，时间还是很快的。



五、源码测试

1、在springboot-boot-tests模块下随便找一个spring-boot-smoke-test-hibernate52工程来进行冒烟测试，打开Hibernate52Application.java文件，直接执行main方法启动springboot，成功!

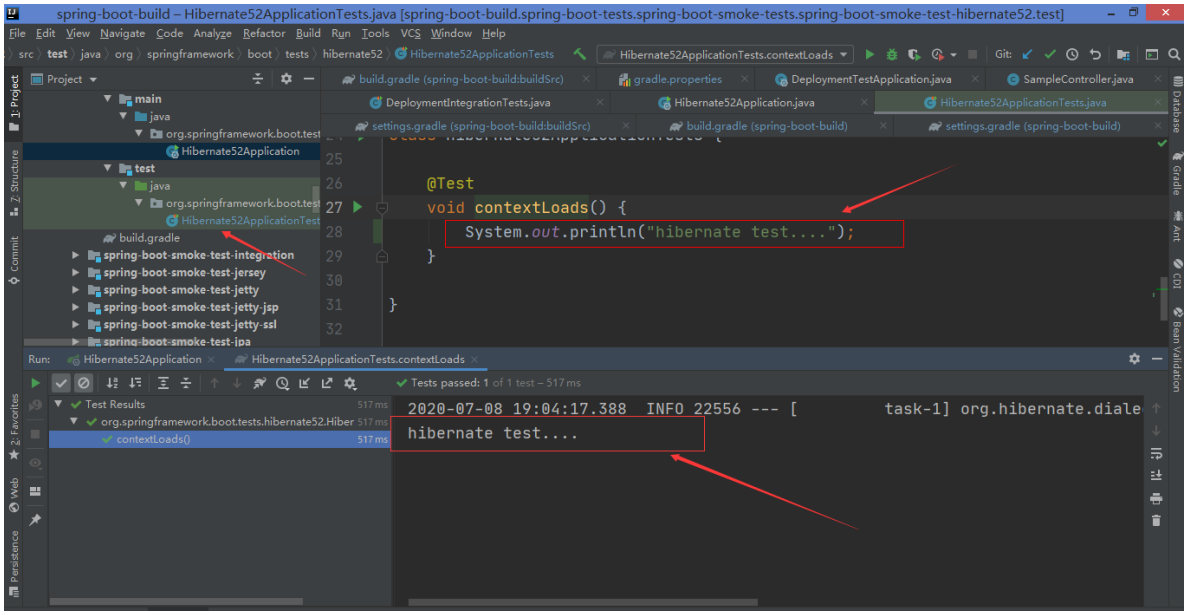
console中出现我们熟悉的图标。



2、下面进行单元测试，还是在spring-boot-smoke-test-hibernate52下的test目录中，打开Hibernate52ApplicationTests.java文件,在contextLoads()方法中加一段打印信息：

```
@Test
void contextLoads() {
    System.out.println("hibernate test....");
}
```

执行Run Test,OK,测试也是没有问题的,测试结果如下图:



至此，spring-boot的源码阅读环境全部搭建并测试完毕，各位小伙伴如果在搭建过程中遇到问题可以加

码炫课堂技术交流q群：963060292

找smart哥进行探讨！

六、问题及解决方案

1、TestFailuresPluginIntegrationTests > multiProjectParallel() FAILED
java.lang.AssertionError at TestFailuresPluginIntegrationTests.java:88

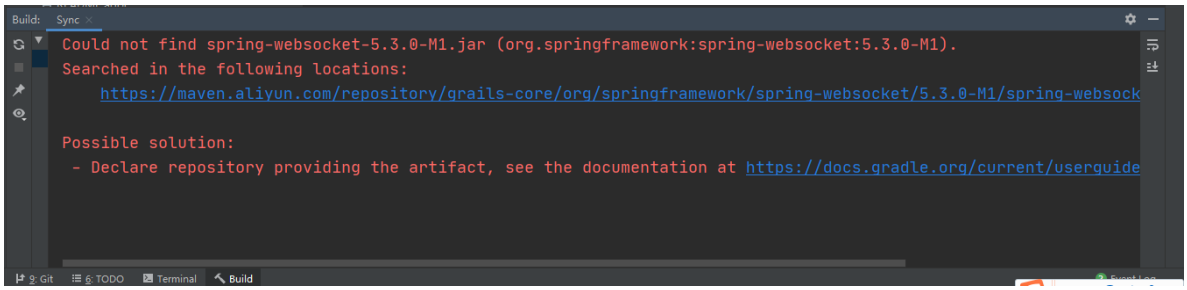
该问题是junit包下载超时，可能是我机器网络抽风了，刷新重新构建即可！

2、spring-websocket-5.3.0-M1.jar包找不到

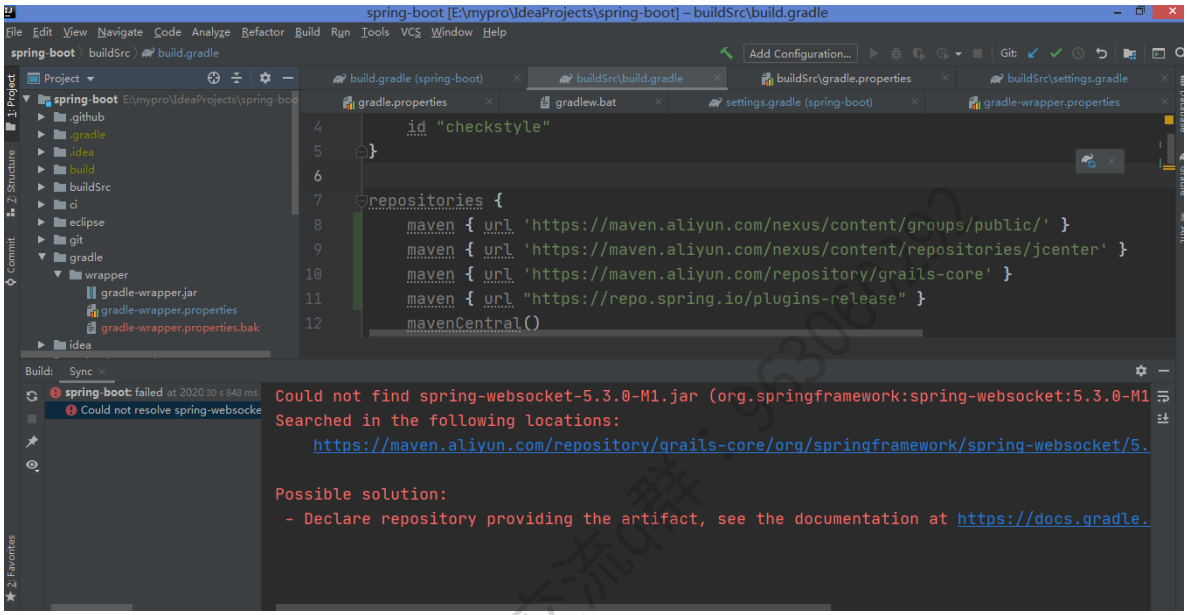
如果小伙伴们拉取的是master分支，就像我在本文开头讲的一样，此时报spring-websocket-5.3.0-M1.jar包找不到，提示在下面的地址中查找。

但是 https://maven.aliyun.com/repository/grails-core/org/springframework/spring-websocket/5.3.0-M1/spring-websocket-5.3.0-M1.jar

这个链接明明是可以下载的，所以smart哥一头雾水，懵圈了很久。



经过smart哥多次刷新重新下还是无法下载，于是抛弃master分支，转而拉取2.3.x分支。



结语

smart哥首创**3位1体**学习法之--**源码篇**-最新【**spring-boot-2.3.x源码解析**】课程即将开启，全盘解析spring-boot-2.3.x的底层源码。

具体开课时间以群公告为准！

额~~群在哪？？

码炫课堂java技术交流Q群：963060292

速来学习，机不可失，时不再来！