查看docker是否安装成功：docker version

修改拉取镜像的国内地址

增加或者修改文件 /etc/docker/daemon.json

加入以下内容：

{

"registry-mirrors": ["http://hub-mirror.c.163.com"]

}

然后重启docker

删除镜像：

docker image rm ubuntu:18.04

docker rmi ubuntu:18.04

启动容器：docker run -p 8080:8080 tomcat

进入容器：docker exec -it 容器id

Dockerfile定制镜像

Dockerfile

FROM tomcat

WORKDIR /usr/local/tomcat/webapps/ROOT/

RUN rm -rf \*

RUN echo “hello world” > /usr/local/tomcat/webapps/ROOT/index.html

根据Dockerfile构建镜像

docker build -t myshop:v3 .

**注意Dockerfile中的每一行命令（比如WORKDIR,RUN等命令）都会构建一个容器在内存中，所以我们整个Dokerfile写错的时候可能会产生容器了，写对了应该没有，所以要注意这种调试时候的无用容器，要记得删除，而且Dockerfile写错了调试的时候也可能生成镜像（虚悬镜像）记得也删除。**

copy和add的区别：

add多一个解压缩的功能，一般生产环境只用copy命令，语义清晰。

RUN和CMD的区别：

run是构建镜像用的，cmd是生成的镜像当启动成容器的时候第一次执行的容器内命令，比如tomcat镜像里面会有catalina.sh run 说明tomcat镜像启动成为了容器就自动运行了

交互式运行

docker run -it -p 80:8080 --name tomcatContainer tomat

后台运行容器

[root@docker home]# docker run -p 80:8080 --name tomcat1 -d tomcat

db129f12e9c0110bb55d08445f62bb9b0da53fb67ed4f262fc609d7e0ccfe09e

[root@docker home]# docker run -p 81:8080 --name tomcat2 -d tomcat

0bdee33d8039fe234679159d8d7ba818166869fedfd97e0af3e70d3e6e4061fe

**启动了两个后台tomcat分别对应宿主机80和81端口，这样也理解了docker容器是负载均衡的入口，**

**容器前面加一层nginx就可以转发到不同的容器了**

**后台运行的怎么进去呢？**

**docker exec -it 容器id bash**

**进去后怎么退出呢？Ctrl+D（不会让容器也停止）**

容器挂载（我是在宿主机/home/docker/ROOT目录下建立了一个index.html）：

docker run -p 82:8080 --name tomcat82 -d -v /home/docker/ROOT:/usr/local/tomcat/webapps/ROOT tomcat

aa18c113043db1ea67c3fefeda357371cfe1da044d8d6ae4dcdbb49d86b75d66

安装mysql

docker pull mysql

安装的mysql是哪个版本呢？去hub.docker.com发现最新版的mysql都8.X了

mysql8.0最新特性 去掉了MyISAM引擎，InnoDB不仅支持事务而且效率提高，互联网企业转型，支持原生分布式解决方案

我们还是换成5.7吧，支持NoSql语法

docker pull mysql:5.7.22

docker run -p 3306:3306 --name mysql \

-v /usr/local/docker/mysql/conf:/var/log/mysql \

-v /usr/local/docker/mysql/logs:/var/log/mysql \

-v /usr/local/docker/mysql/data:/var/lib/mysql \

-e MYSQL\_ROOT\_PASSOWORD=123456 \

-d mysql:5.7.22

docker三剑客：

compose：还在用

machine（不怎么用）

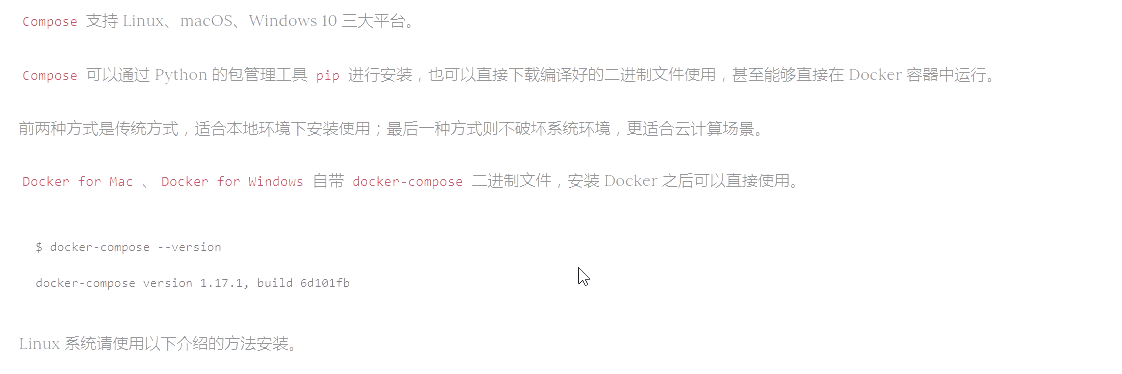
**swarm:(已经过时)**

**docker compose:**





安装



通过pip安装

yum -y install epel-release

yum -y install python3-pip

yum install gcc libffi-devel python-devel openssl-devel -y

pip3 install -U docker-compose

docker-compose文件编写(**注意用两个空格，不能用tab**)

[root@docker tomcat]# cat docker-compose.yml

version: "3.7"

services:

tomcat:(这个名字也可以随便起,有意义的就行)

restart: always

image: tomcat

container\_name: tomcat(启动的容器的名字，随便起)

ports:

- 8080:8080

docker-compose up

然后ctrl+C停止

再查看容器docker ps -a 发现退出了

docker-compose down 删除容器

docker-compose up -d 后台启动一组容器(容器也叫服务)

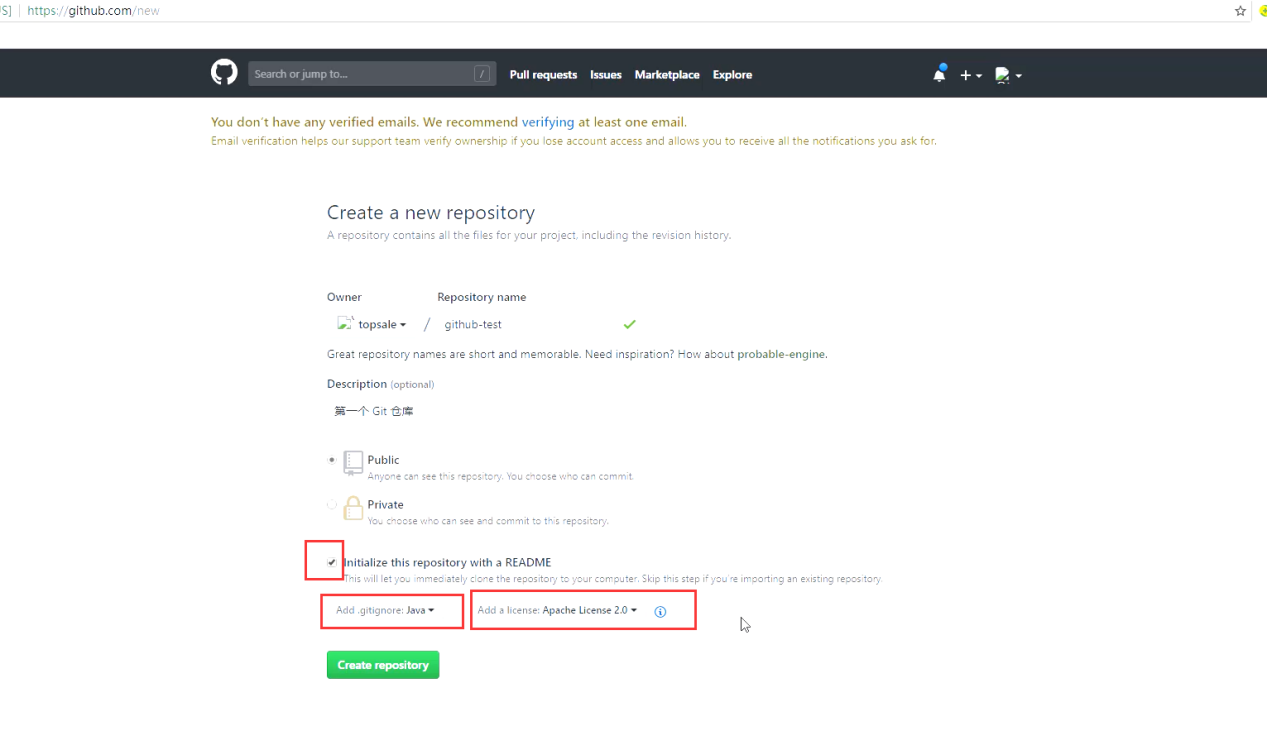
docker-compose logs tomcat 查看日志

git

先安装git，然后安装乌龟git，然后安装乌龟git中文破解包

github

注意几点最好勾选上，以前没注意过



.gitignore选成java有好处

建立好仓库后，选择clone with https的地址

然后本地找一个地方，git clone 地址就能跟仓库建立好联系了

然后.gitignore文件增加两行

.project

.idea

这两个开发工具自动生成不用提交

然后

git add .

git commit -m”增加eclipse和idea的过滤”

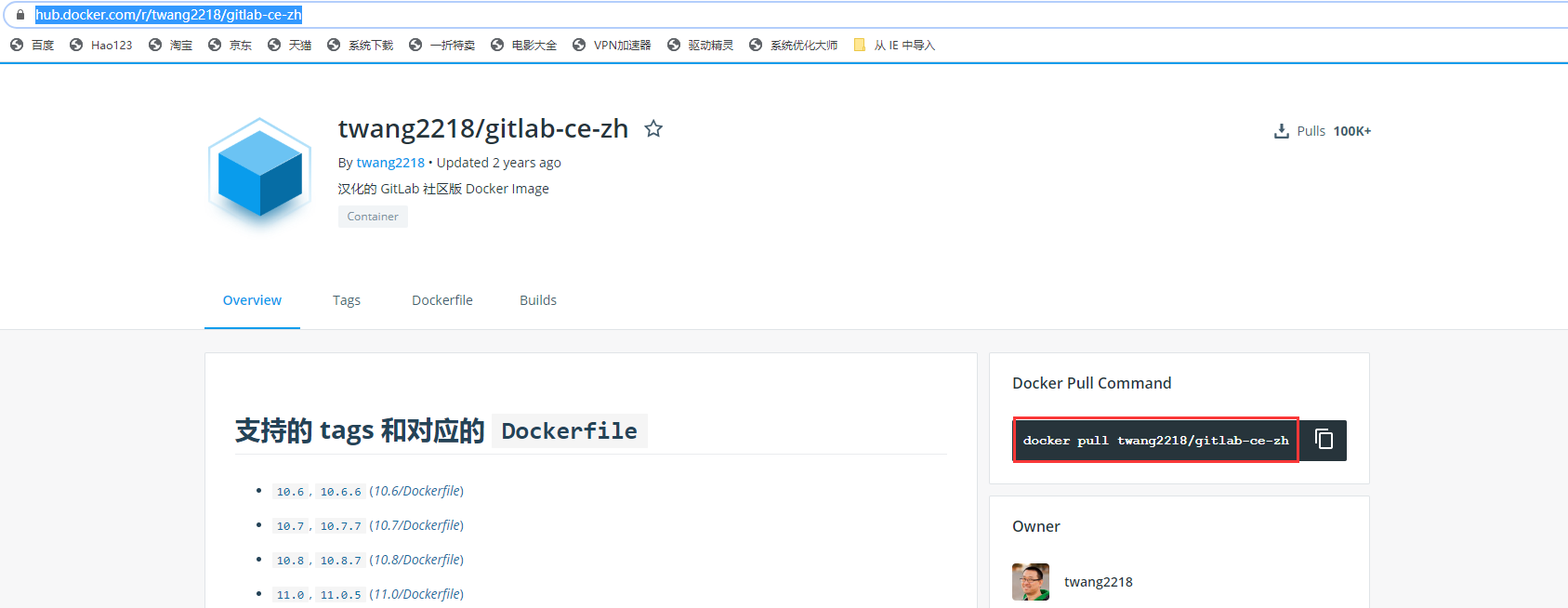
git push

公司内部安装个gitlab

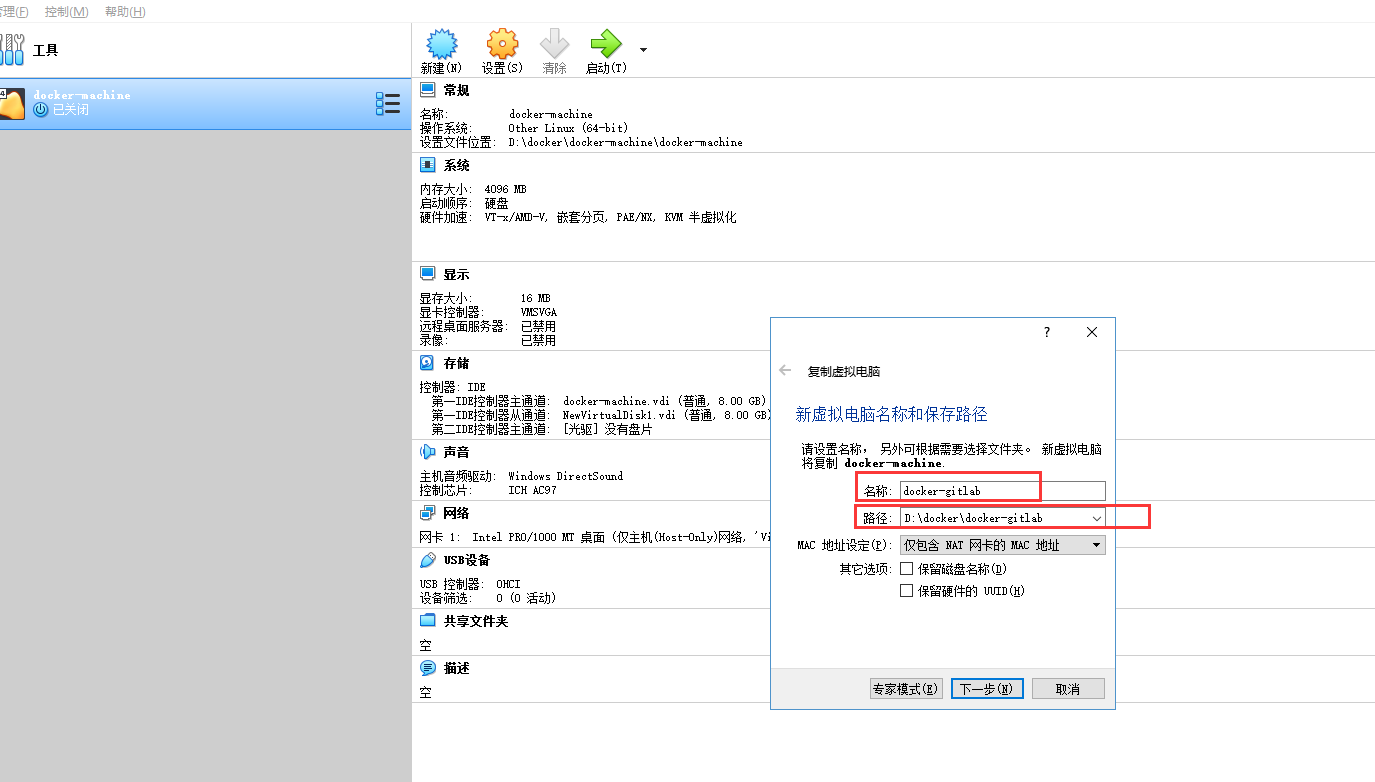
<http://hub.docker.com/>找gitlab的中文版

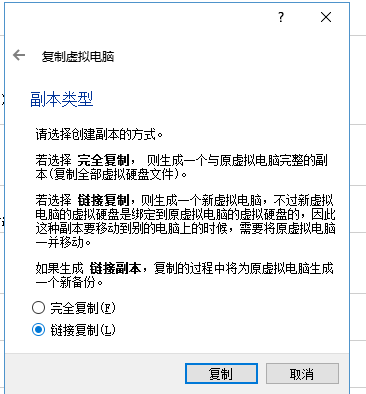
搜索这个gitlab-ce-zh关键字

<https://hub.docker.com/r/twang2218/gitlab-ce-zh>



学习gitlab之前再弄个虚拟机吧





链接复制有个特别有意思的地方，就是新复制的虚拟机跟以前的虚拟机是一样的ip，我用linux客户端工具连接，你都不知道自己连着谁，所以最好同时启动的虚拟机就一个，要不ip都一样有点乱。

题外话

docker pull过慢的解决办法,一直要执行以下三步骤

|  |
| --- |
| **在 /etc/docker/daemon.json 文件中添加以下参数（没有该文件则新建）：****{** **"registry-mirrors": ["https://9cpn8tt6.mirror.aliyuncs.com"]****}****服务重启：** systemctl daemon-reload(**以前都是没执行这个步骤，导致加速器不生效**)  systemctl restart docker |

gitlab compose

|  |
| --- |
| [root@docker gitlab]# cat docker-compose.yml  version: "3"  services:  web:  image: "tweng2218/gitlab-ce-zh"  restart: always  hostname: "192.168.137.100"  environment:  TZ: "Asia/Shanghai"  GITLAB\_OMNIBUS\_CONFIG:  external\_url "http://192.168.137.100"  gitlab\_rails['gitlab\_shell\_ssh\_port'] = 2222  unicorn['port'] = 8888  nginx['listen\_port'] = 80  ports:  - '80:80'  - '8443:443'  - '2222:22'  volumes:  - /usr/local/docker/gitlab/config:/etc/gitlab  - /usr/local/docker/gitlab/data:/var/opt/gitlab  - /usr/local/docker/gitlab/logs:/var/log/gitlab  [root@docker gitlab]# pwd  /usr/local/docker/gitlab |

执行docker-compose up总是执行不起来gitlab，估计是网络问题，先简单学学吧。

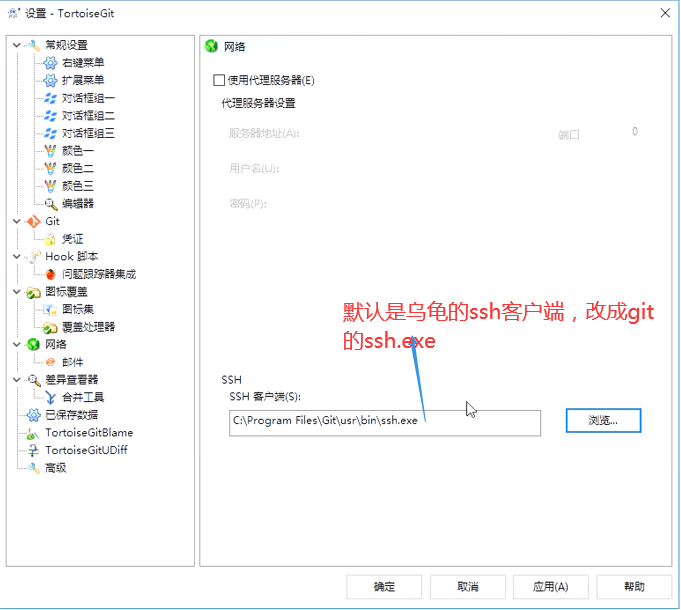
设置ssh访问gitlab

本地建立公钥和私钥，

ssh-kengen -t rsa -C “jinyuchuan@fang.com”

把公钥拷贝到gitlab上，本地留着私钥。

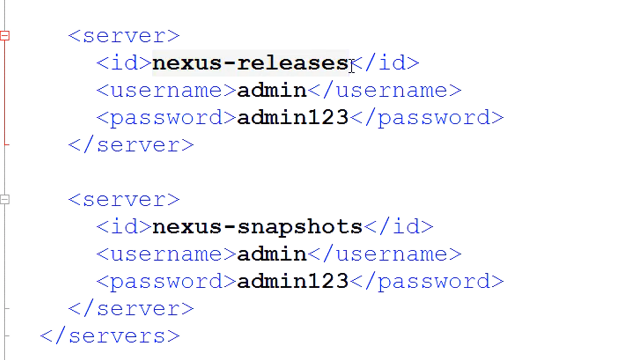
下载项目的时候记得先修改乌龟git客户端的设置



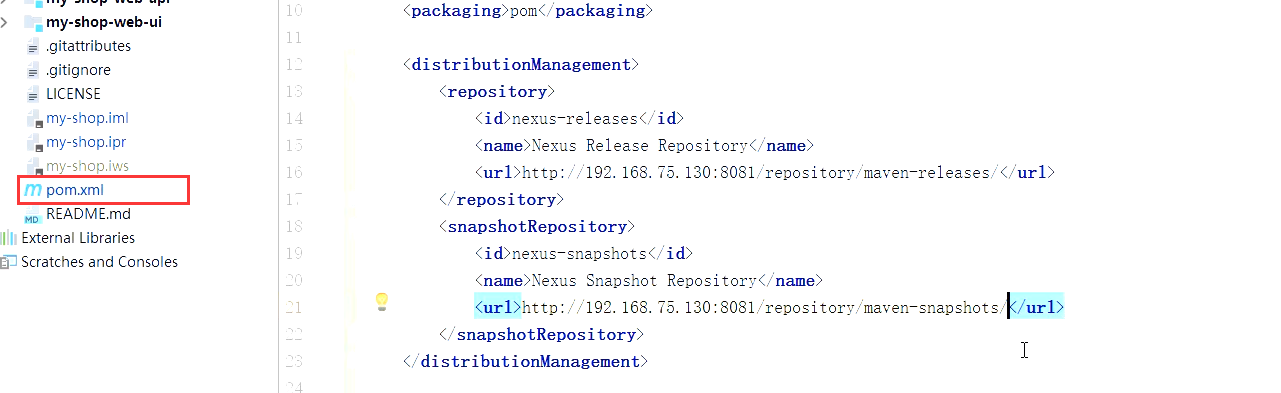
nexus(也用docker-compose安装，装坏了没事docker就是方便)

这里内存要分配两个G，内存溢出了，用htop小工具看内存，比top好用

maven settings里面配置



项目pom里面配置（用来往私服放东西）



注意id名称必须与settings.xml中的的Servers配置的id名称保持一致

项目pom里面配置（用来从私服拉取文件）



下面装docker registry docker镜像私服，略过

私服一般地址是:ip:5000/v2

然后我们docker pull的时候怎么知道走私服的配置呢？

配置机器的/etc/docker/daemon.json文件

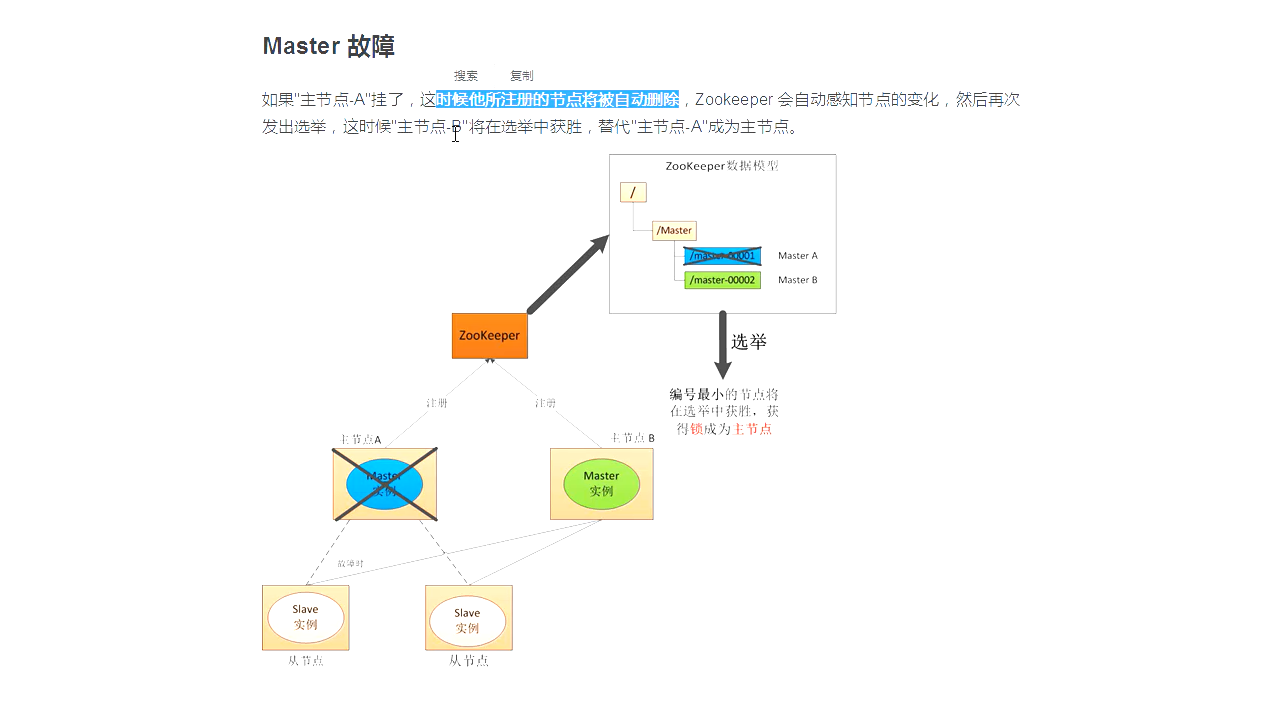


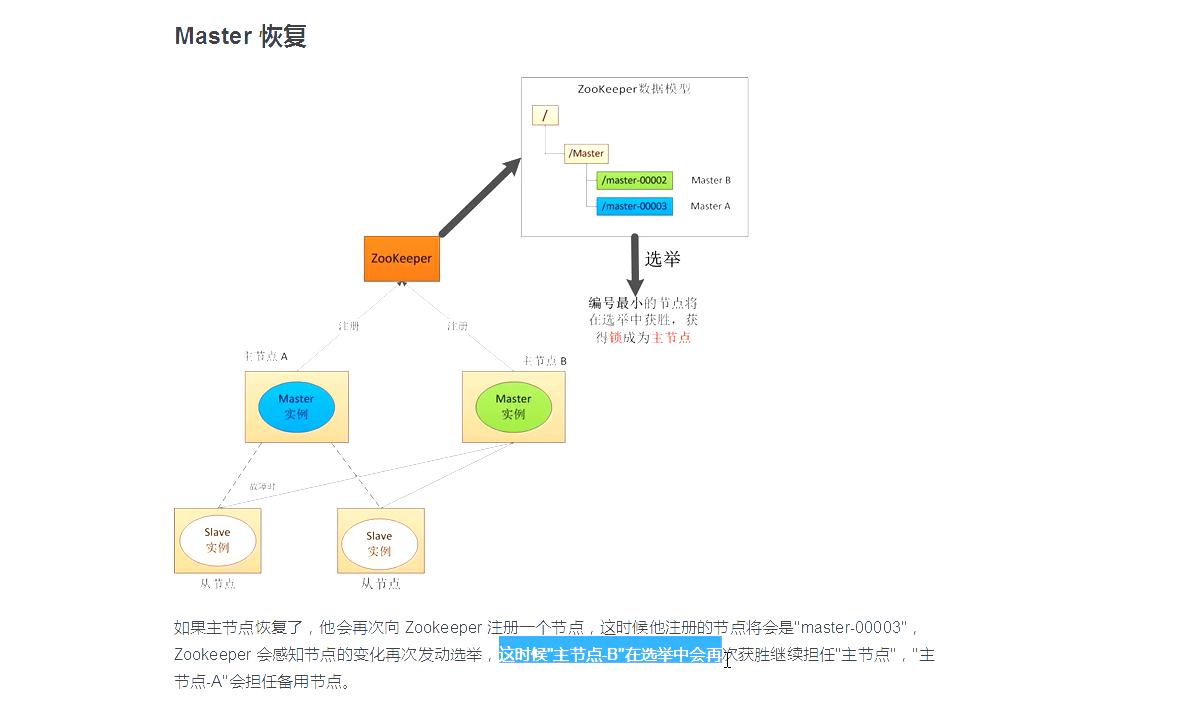
对内(公司内部)rpc（dubbo）,对外rest(springboot)

**单点故障解决办法：使用zookeeper选举主节点（有几个备用主节点，怎么知道哪个主节点真正不能提供服务，而我这个备用节点却可以当上主节点继续提供服务了呢，方法就是所有主节点向zookeeper注册节点，有zk在所有节点上选举出主节点，当主节点挂掉以后，zk会把这个主节点拿掉，从剩余的节点重新选举出来一个当主节点）**

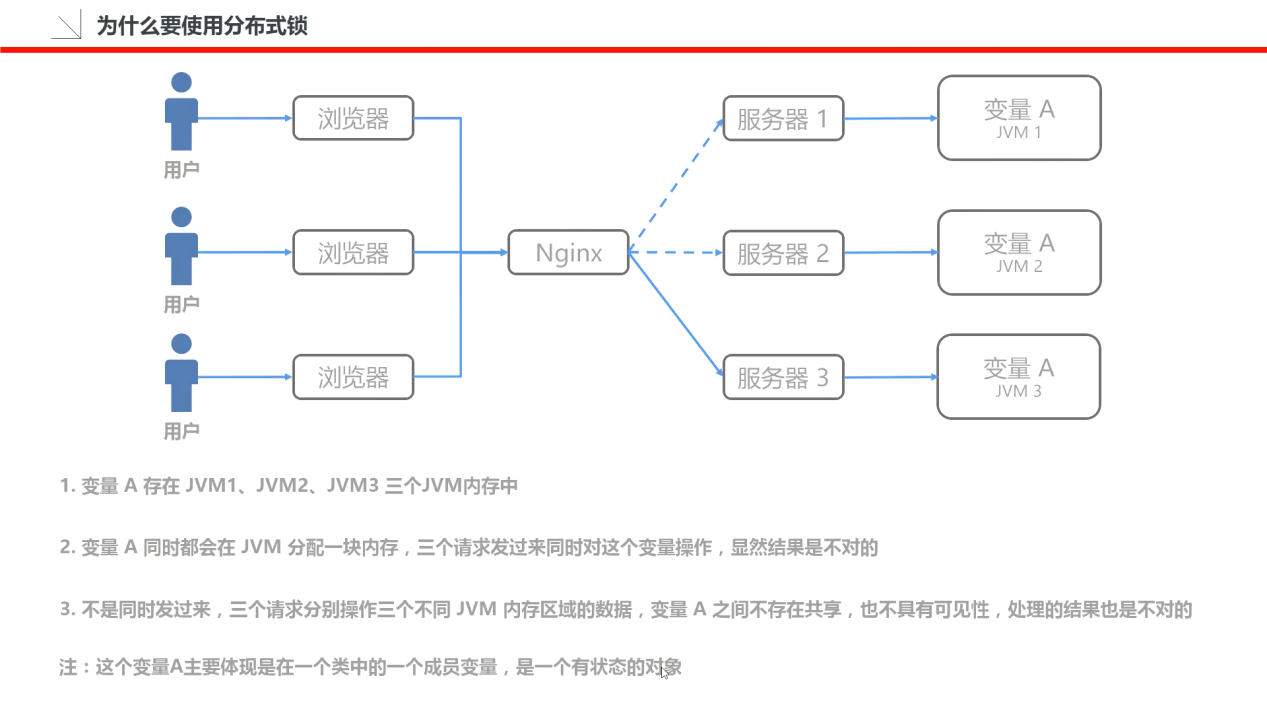
**问题来了，为什么不能备用节点ping主节点，从而知道主节点是否挂了呢？因为由于网络震荡，主节点可能没挂掉，这时候备用节点以为主节点挂了，那就麻烦了，就会启动自己，从而出现了两个主节点，这样系统就乱了**，解释图如下







分布式锁的应用场景？（**注意这时候你用多线程已经解决不了下面这个截图的问题了，因为分布式部署项目，你已经有好多个jvm了，而我们要操作的是同一个变量！update jyc**）

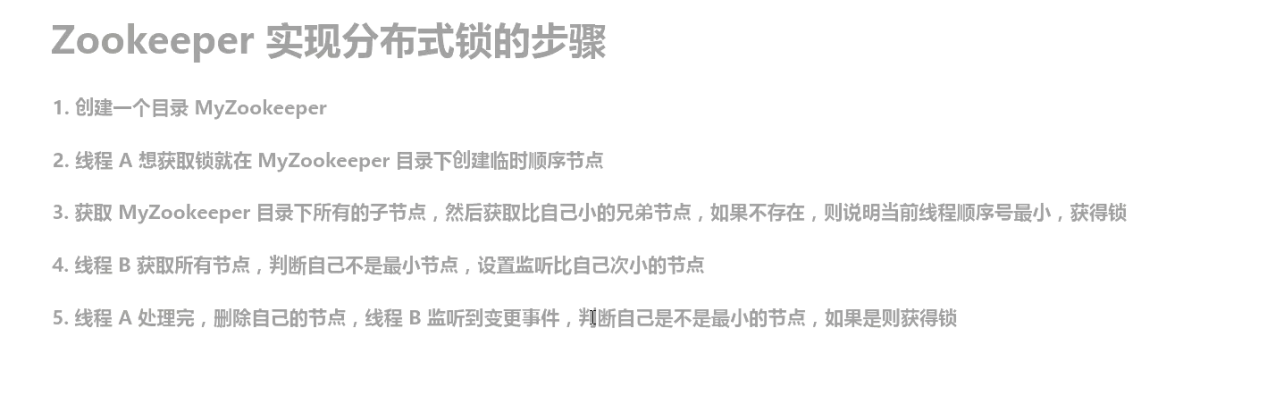




分布式锁特别难实现，要实现以下六个特点

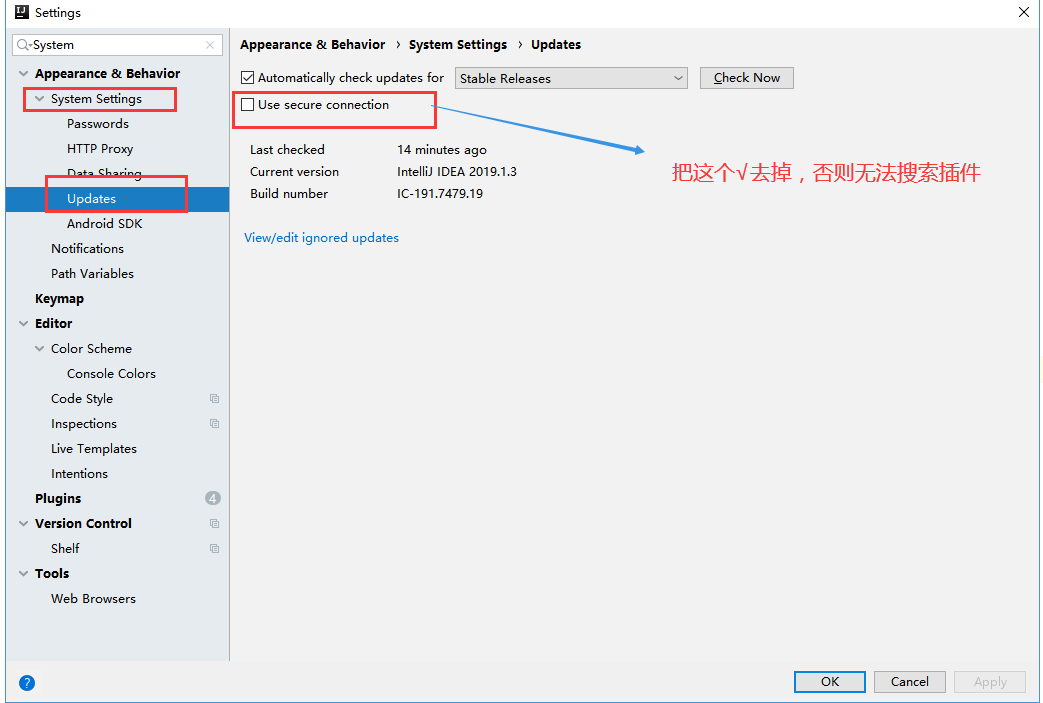


实际项目中实现分布式锁的具体实现方式，指导思想入下图，**这个指导思想很重要**

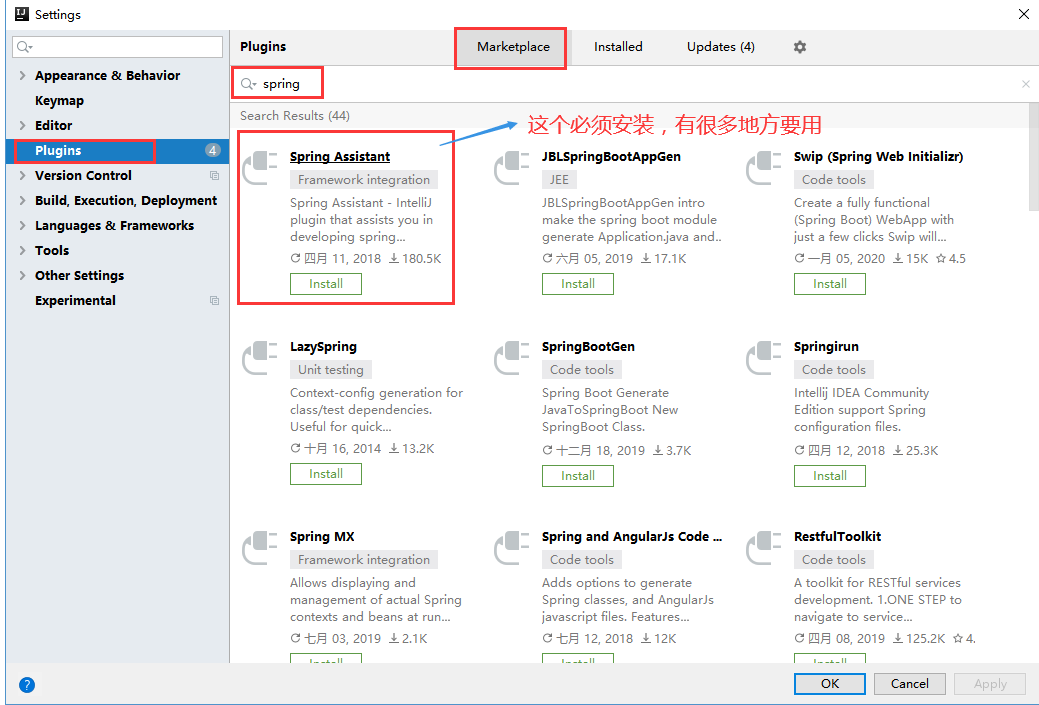


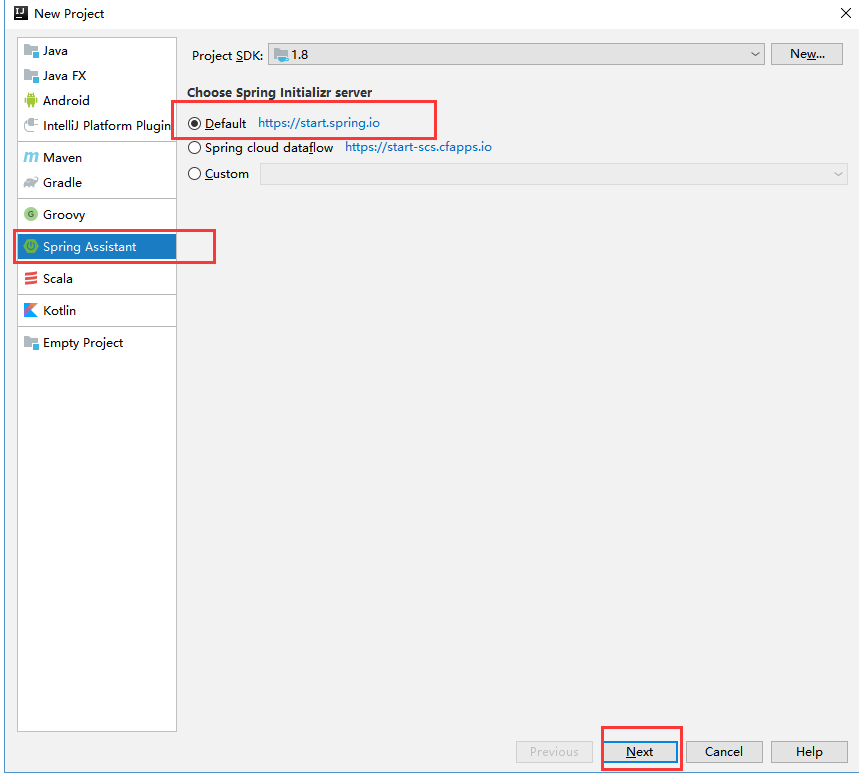
下面开始玩springboot了，先做一个非常重要的设置

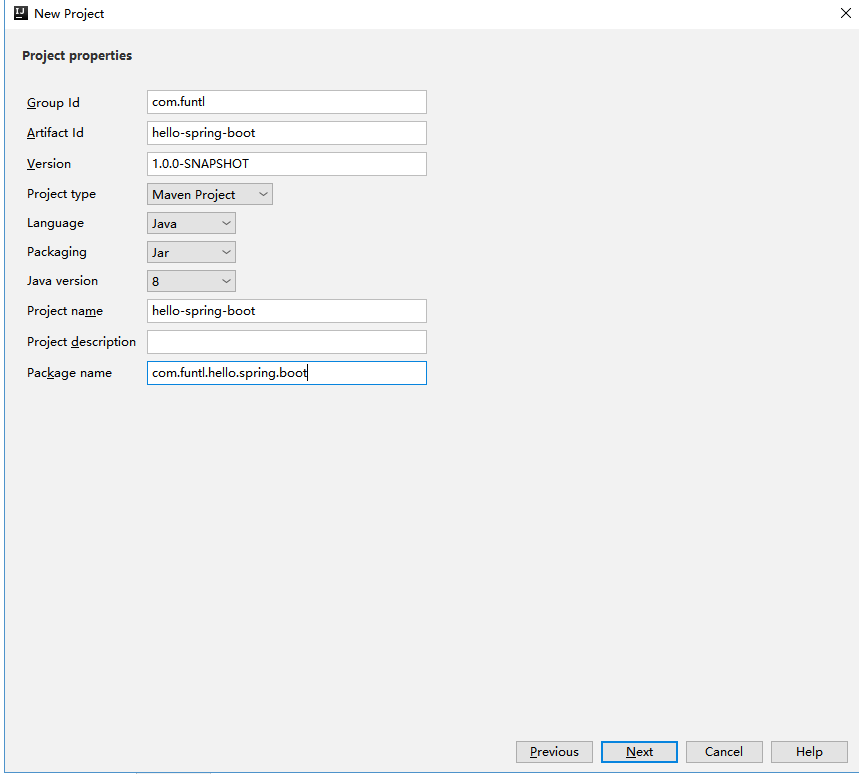
要不然在idea中搜索不到插件

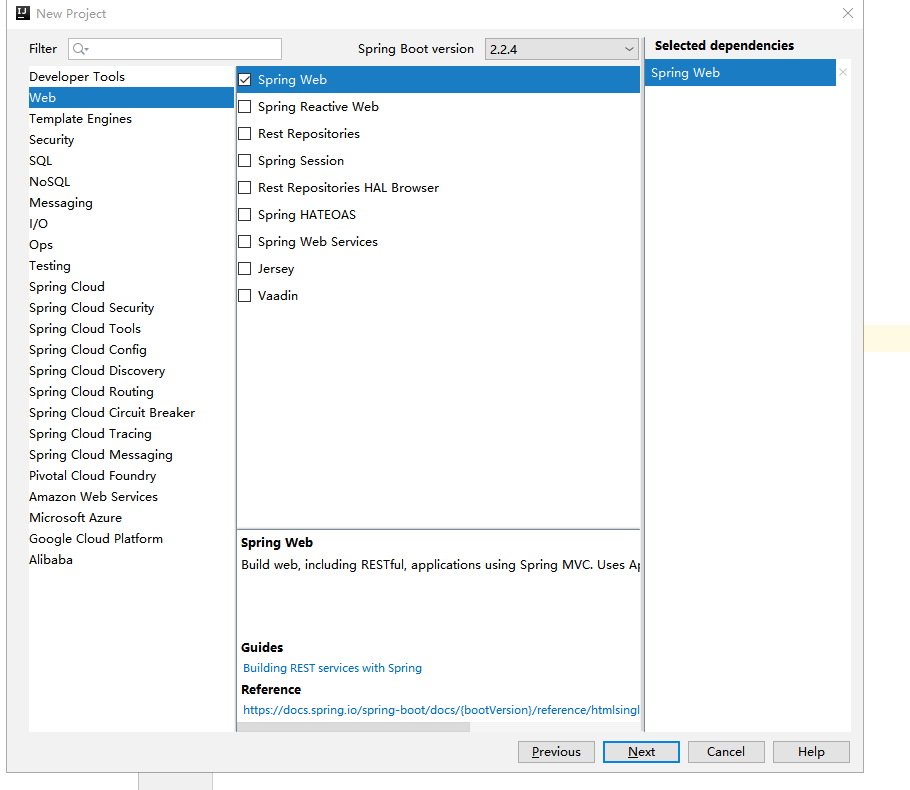


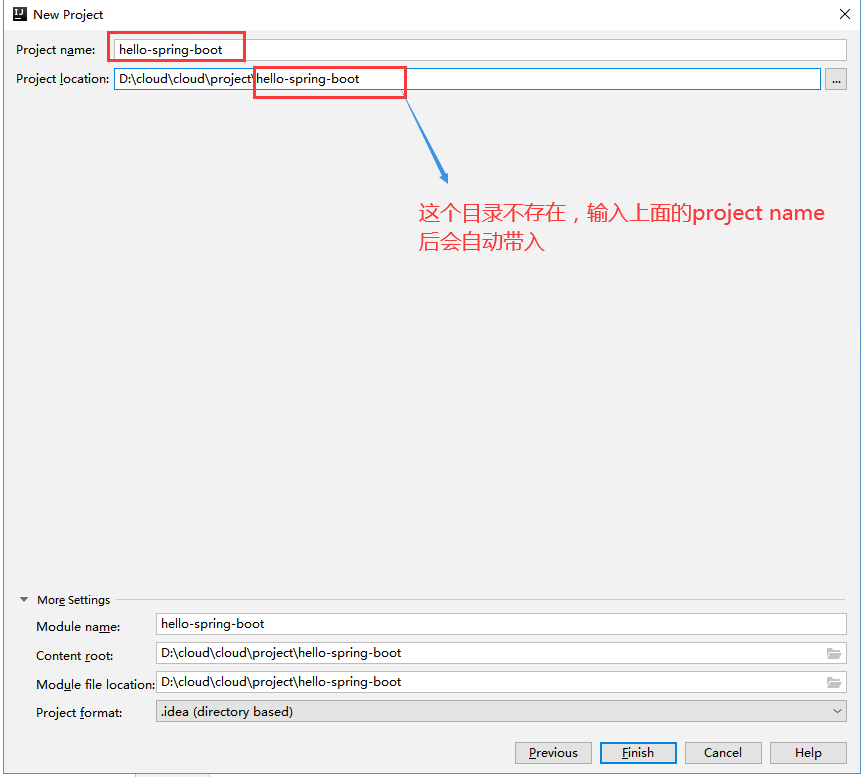
然后重启idea,并安装spring assistant











这才建立好了一个springboot项目！

以后所有的springboot和springcloud的代码放到这个目录下

<https://github.com/15801001455/cloud.git>