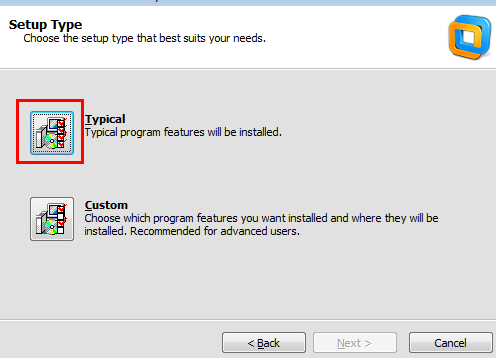
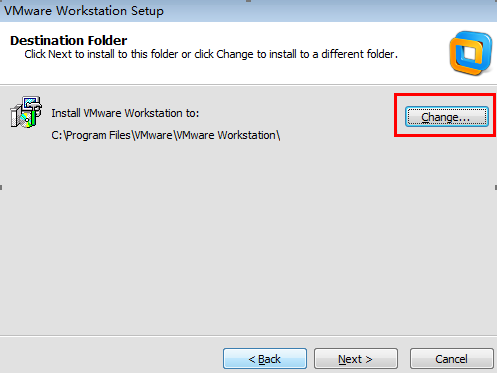
1. **安装VMware虚拟机软件**

下载VMware，双击运行安装程序

选择典型安装和安装位置





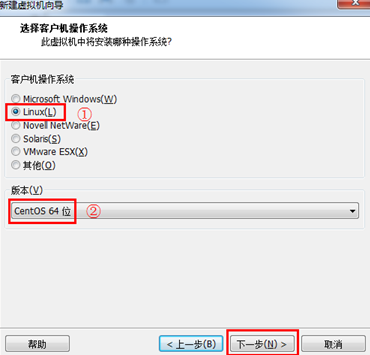
完成软件安装



1. **在虚拟机中安装Linux操作系统**

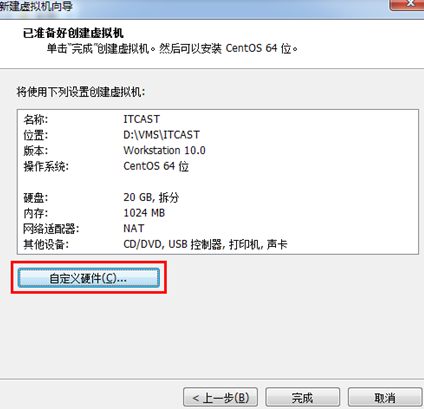
新建虚拟机，选择典型，点击稍后安装操作系统





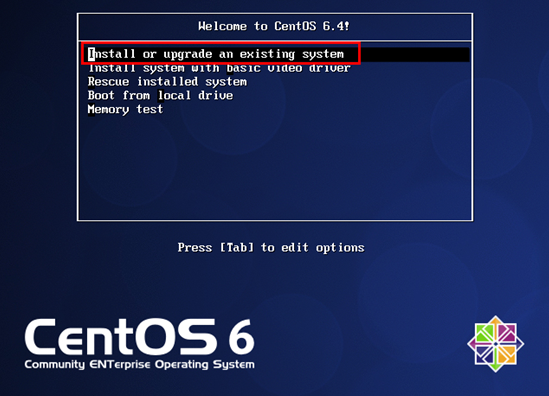
选择Linux和版本为CentOS，点击下一步

输入虚拟机名称和安装路径，设置磁盘大小

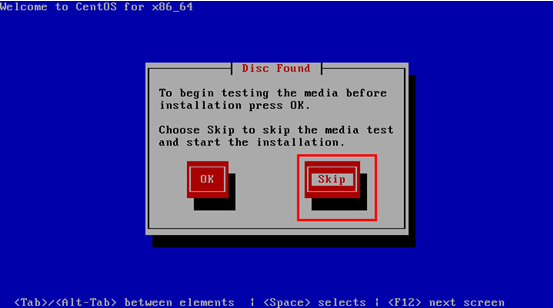


点击自定义硬件并选择CentOS镜像文件，点击完成，随后可以启动虚拟机

选择第一项安装全新操作系统或升级现有操作系统



按Tab键进行选择，选择skip，推出检测



选择语言、键盘样式、存储设备、输入主机名、时区

设置root用户密码

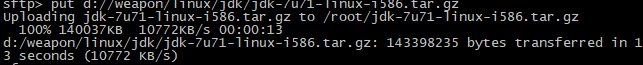


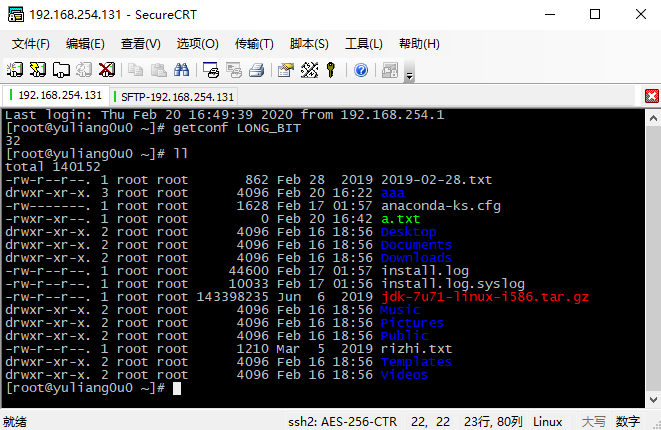
安装完成后点击重新引导，创建用户，完成安装

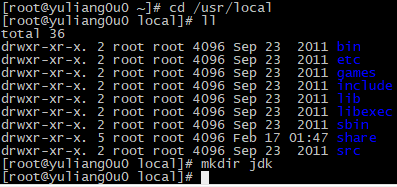
1. **在Linux中安装JDK并完成环境变量配置**

在命令行中输入命令getconf LONG\_BIT查看Linux是多少位的

选择对应版本的jdk使用sftp上传



****

****

卸载Linux自带的open-JDK:

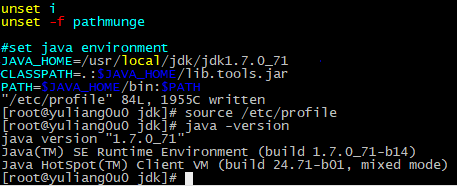
rpm -e --nodeps java-1.6.0-openjdk-1.6.0.35-1.13.7.1.el6\_6.i686

rpm -e --nodeps java-1.7.0-openjdk-1.7.0.79-2.5.5.4.el6.i686

将压缩包解压：

Tar –zxvf jdk-7u71-linux-i586.tar.gz

配置环境变量：vim /etc/profile



Source /etc/profile重新加载更新配置

java –version查看是否安装成功

1. **安装Nginx服务器并启动**

安装gcc环境：yum install gcc-c++

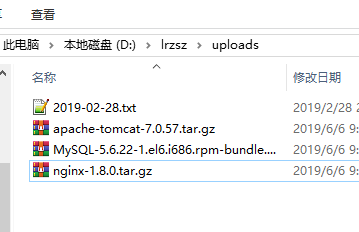
安装第三方的开发包：

yum install –y pcre pcre-devel

yum install –y zlib zlib-devel

yum install -y openssl openssl-devel

下载Nginx并且把源码包上传到linux



解压缩：tar –zxvf nginx-1.8.0.tar.gz

进入nginx-1.8.0目录 使用 configure 命令创建makeFile 文件

./configure \

--prefix=/usr/local/nginx \

--pid-path=/var/run/nginx/nginx.pid \

--lock-path=/var/lock/nginx.lock \

--error-log-path=/var/log/nginx/error.log \

--http-log-path=/var/log/nginx/access.log \

--with-http\_gzip\_static\_module \

--http-client-body-temp-path=/var/temp/nginx/client \

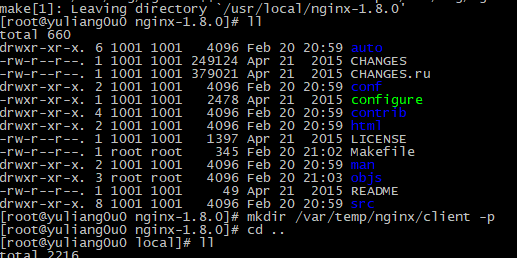
--http-proxy-temp-path=/var/temp/nginx/proxy \

--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \

--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \

--http-scgi-temp-path=/var/temp/nginx/scgi

执行后可以看到Makefile文件



编译：make

安装：make install

启动nginx 之前，上边将临时文件目录指定为/var/temp/nginx/client， 需要在/var下创建此目录：mkdir /var/temp/nginx/client –p

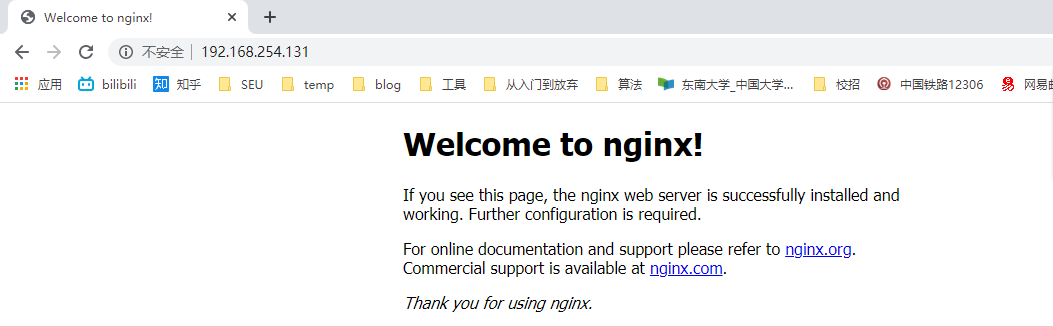
进入到nginx目录下的sbin目录

cd /usr/local/nginx/sbin

输入命令启动Nginx

./nginx

本地浏览器访问



1. **将京东首页部署至SpringBoot项目，并启动三个不同Web服务器**

由于阶段一作业本地备份丢失，Github下载缓慢，直接使用本地其他静态页面替换

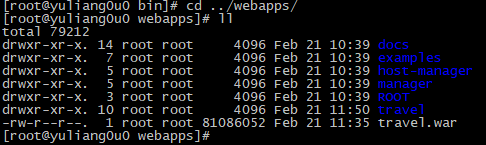
在springboot项目资源路径下的static中放置静态首页

在本地数据库中导出sql脚本，远程连接Linux上安装的mysql,执行该脚本

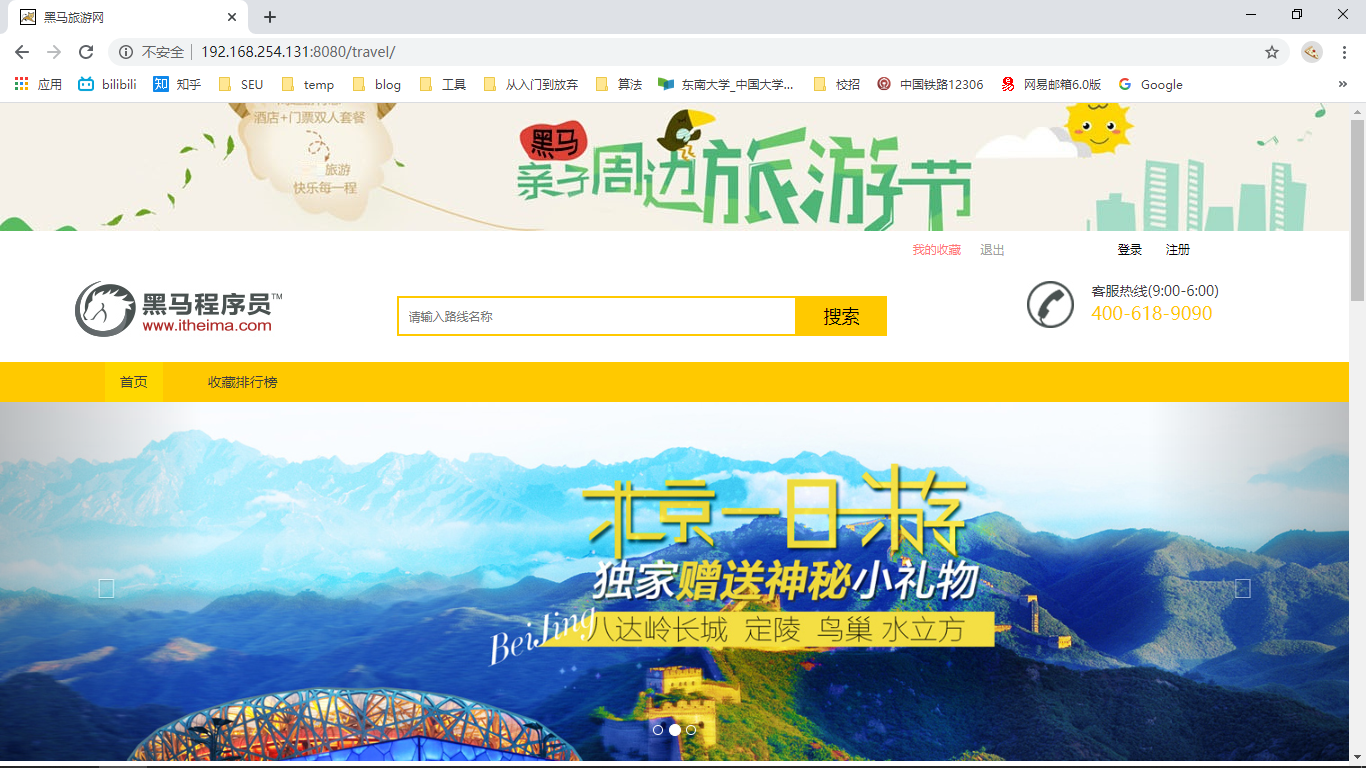
使用maven工具将项目打包

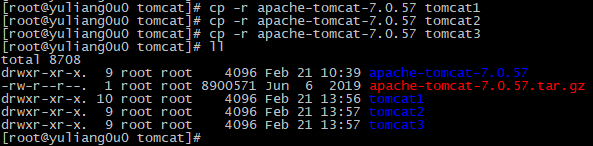
找到对应war包并上传到Linux

启动Linux上的tomcat，war包就会自动解压缩

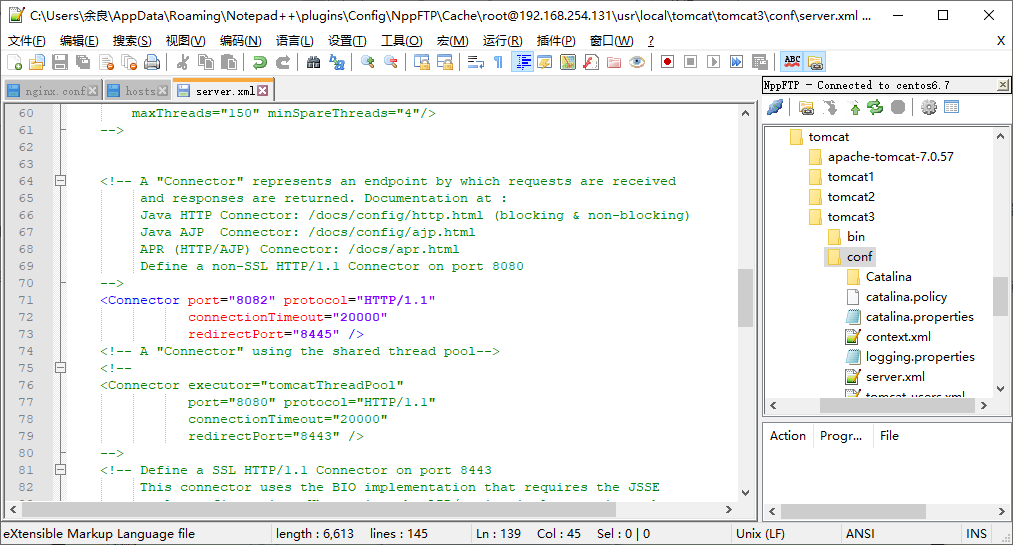


本地浏览器输入ip端口项目路径即可访问

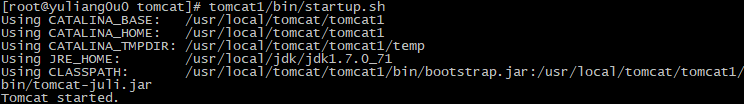




将tomcat复制三分，并且修改端口

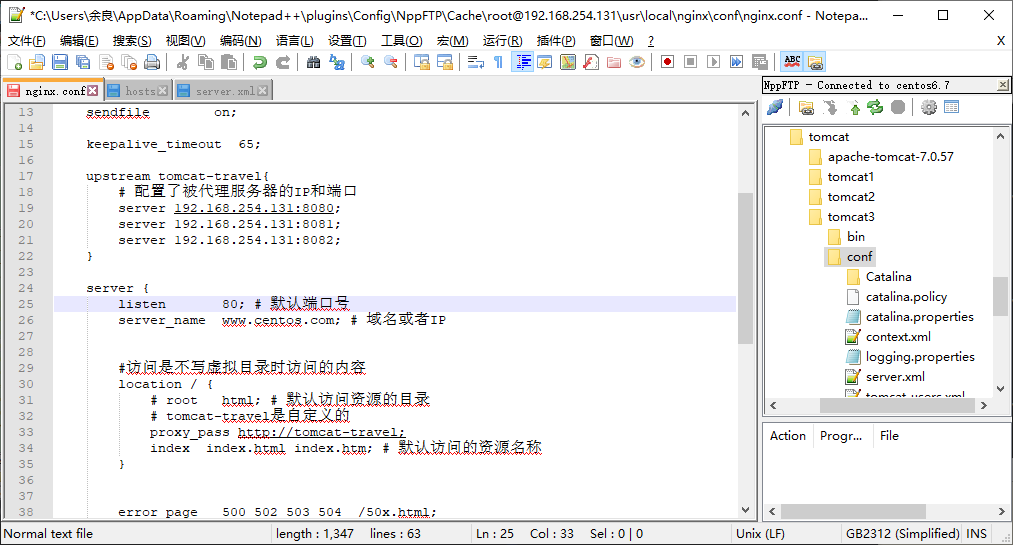


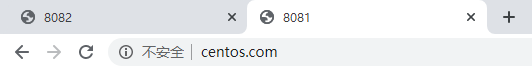
分别启动三个tomcat



修改代理服务器Nginx配置，输入域名nginx会随机分发到三个端口中的一个

为了观察方便在index.html中将title标签体分别改为8080、8081、8082

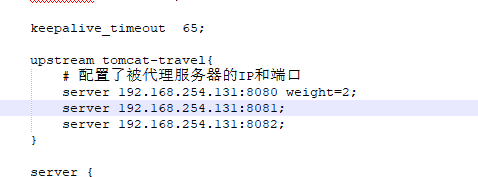




多次刷新发现是随机访问不同端口的

1. **使用Nginx实现负载均衡配置，并测试负载是否生效**

配置权重，增加8080端口的权重



结果是访问到8080端口机率增加