

# 关卡 1

## 鲲鹏平台部署轻商城系统



华为技术有限公司

# 1 华为鲲鹏代码迁移工具移植部署 MySQL

作业提交任务如下：

要求：截图中请包含个人的 ECS、IP 等信息，包括但不限于在浏览器中出现的公网 IP、ECS 个人命名等，若无相关信息，则忽略本条要求。

1. MySQL 源码迁移报告截图。（10 分）

|   |                                      |         |            |         |            |  |  |
|---|--------------------------------------|---------|------------|---------|------------|--|--|
| Scanned time  | 2020/7/6 19:58                       |         |            |         |            |  |  |
| Make/cmake parsing failed. Check all the files under the path   |                                      |         |            |         |            |  |  |
| Scan code path  | /opt/portadv/portadmin/mysql-5.7.29/ |         |            |         |            |  |  |
| GCC   | GCC 7.3                              |         |            |         |            |  |  |
| SW make command   | cmake                                |         |            |         |            |  |  |
| Target OS   | openeuler20.03                       |         |            |         |            |  |  |
| kernel  | 4.19.90                              |         |            |         |            |  |  |
| Scanned 1 SO libraries.   |                                      |         |            |         |            |  |  |
| Here, 0 SO libraries have been verified on the Kunpeng platform, the Kunpeng community has an arm64 version, the url is the download address (binary package).                    |                                      |         |            |         |            |  |  |
| Here, 0 SO libraries have been verified on the Kunpeng platform, the Kunpeng community has an arm64 version, the url is the source address, user need to compile on the platform. |                                      |         |            |         |            |  |  |
| Here, 0 SO libraries cannot be supported on the Kunpeng platform, and the Kunpeng community has no alternative.   |                                      |         |            |         |            |  |  |
| Here, 1 SO libraries are not recognized on the Kunpeng platform.  |                                      |         |            |         |            |  |  |
| Source Need Migrated: false   |                                      |         |            |         |            |  |  |
| Scanned 13867 C/C++ files,10 Makefile/CMakelists/Automake related files,total 0 files need to be migrated.  |                                      |         |            |         |            |  |  |
| Total 0 lines C/C++/Makefile/CMakelists/Automake code and 0 lines embedding ASM code need to be migrated.   |                                      |         |            |         |            |  |  |
| Scanned 0 pure assembly files,no pure assembly files to be migrated.  |                                      |         |            |         |            |  |  |
| SO files scan details are as follows:   |                                      |         |            |         |            |  |  |
| Here, 1 SO libraries are not recognized on the Kunpeng platform.libname   | libversion                           | os_name | os_version | level   | url        | suggestion   |  |
| libdos32a.so  |                                      |         |            | 3       |            | so libraries are not recognized on the Kunpeng platform. |  |
| Source files scan details are as follows:   |                                      |         |            |         |            |  |  |
| filename  | lineno                               | rows    | category   | keyword | suggestion | description  |  |

2020/07/06 19:58:43

|         |                                      |        |                |          |         |
|---------|--------------------------------------|--------|----------------|----------|---------|
| 源代码存放路径 | /opt/portadv/portadmin/mysql-5.7.29/ | 编译器版本  | GCC 7.3        | 构建工具     | cmake   |
| 编译命令    | cmake                                | 目标操作系统 | openeuler20.03 | 目标系统内核版本 | 4.19.90 |

### 分析结果

#### 依赖库SO文件

总数: 1, 需要迁移: 1

| 序号 | 名称           | 处理建议                  | 操作             |
|----|--------------|-----------------------|----------------|
| 1  | libdos32a.so | 无法确认鲲鹏平台是否支持该文件，请检查确认 | 无法找到下载链接，请检查确认 |

|   |                                 |
|---|---------------------------------|
|  需要迁移的源文件  | 0                               |
|  需要迁移的代码行数 | C/C++和Makefile源代码: 0行; 汇编代码: 0行 |

## 2. 启动 MySQL 服务成功截图。（10 分）

```
[root@ecs-litemall-kunpeng-tsz run]# cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql
[root@ecs-litemall-kunpeng-tsz run]# chmod +x /etc/init.d/mysql
[root@ecs-litemall-kunpeng-tsz run]# chkconfig mysql on
[root@ecs-litemall-kunpeng-tsz run]# service mysql start
Starting MySQL. SUCCESS!
[root@ecs-litemall-kunpeng-tsz run]#
```

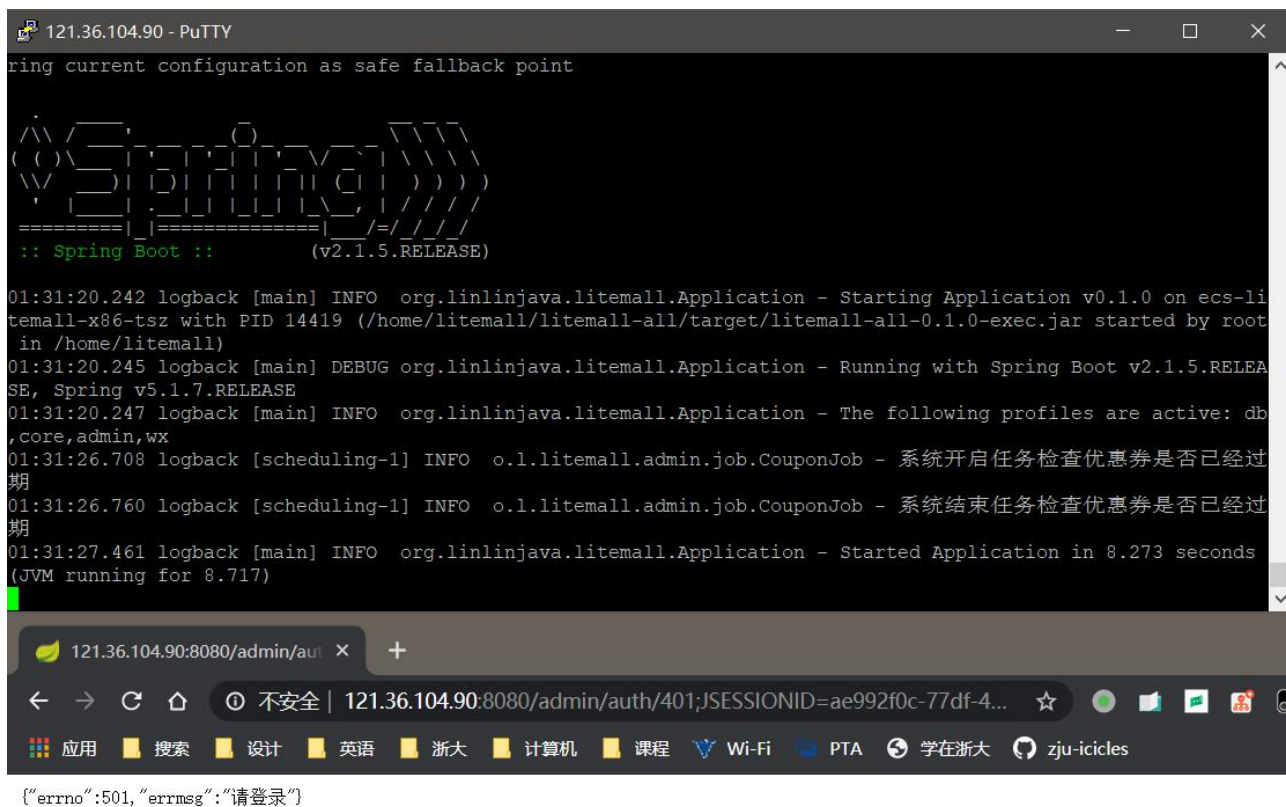
# 2

## x86 平台部署轻商城后端服务

作业提交任务如下：

要求：截图中请包含个人的 ECS、IP 等信息，包括但不限于在浏览器中出现的公网 IP、ECS 个人命名等，若无相关信息，则忽略本条要求。

1. 通过 ecs-litemall-x86 的弹性公网 IP 加上 8080 端口的形式访问管理后台截图，包含“请登录”信息。（10 分）



The screenshot displays a terminal window titled "121.36.104.90 - PuTTY" showing the startup logs of the application. The logs indicate that the application is starting on ECS-litemall-x86-tsz with PID 14419, running with Spring Boot v2.1.5.RELEASE. The application is started in 8.273 seconds. Below the terminal window, a web browser window is shown with the address bar displaying "121.36.104.90:8080/admin/auth/401;JSESSIONID=ae992f0c-77df-4...". The browser window shows the login page with the message "请登录" (Please login).

```
ring current configuration as safe fallback point

:: Spring Boot :: (v2.1.5.RELEASE)

01:31:20.242 logback [main] INFO org.linlinjava.litemall.Application - Starting Application v0.1.0 on ecs-li
temall-x86-tsz with PID 14419 (/home/litemall/litemall-all/target/litemall-all-0.1.0-exec.jar started by root
in /home/litemall)
01:31:20.245 logback [main] DEBUG org.linlinjava.litemall.Application - Running with Spring Boot v2.1.5.RELEA
SE, Spring v5.1.7.RELEASE
01:31:20.247 logback [main] INFO org.linlinjava.litemall.Application - The following profiles are active: db
,core,admin,wx
01:31:26.708 logback [scheduling-1] INFO o.l.litemall.admin.job.CouponJob - 系统开启任务检查优惠券是否已经过
期
01:31:26.760 logback [scheduling-1] INFO o.l.litemall.admin.job.CouponJob - 系统结束任务检查优惠券是否已经过
期
01:31:27.461 logback [main] INFO org.linlinjava.litemall.Application - Started Application in 8.273 seconds
(JVM running for 8.717)
```

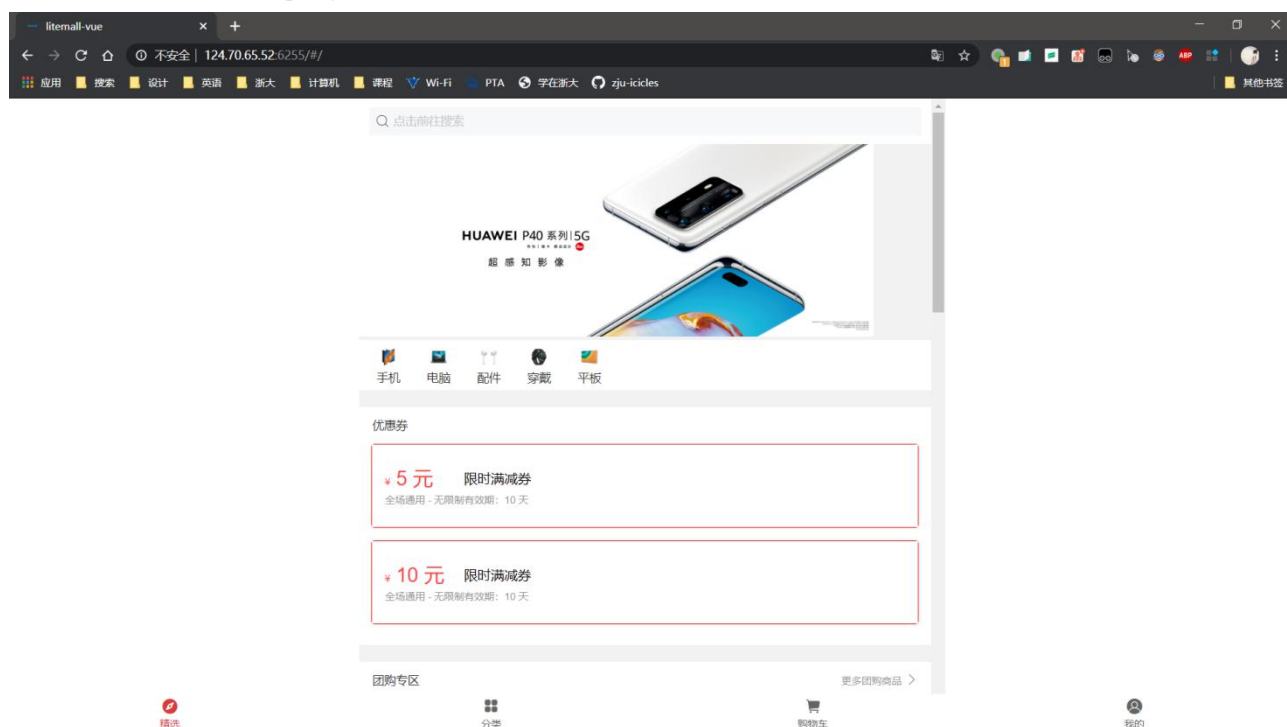
121.36.104.90:8080/admin/auth/401;JSESSIONID=ae992f0c-77df-4... 不安全 | 121.36.104.90:8080/admin/auth/401;JSESSIONID=ae992f0c-77df-4... 请登录

# 3 鲲鹏平台部署轻商城系统

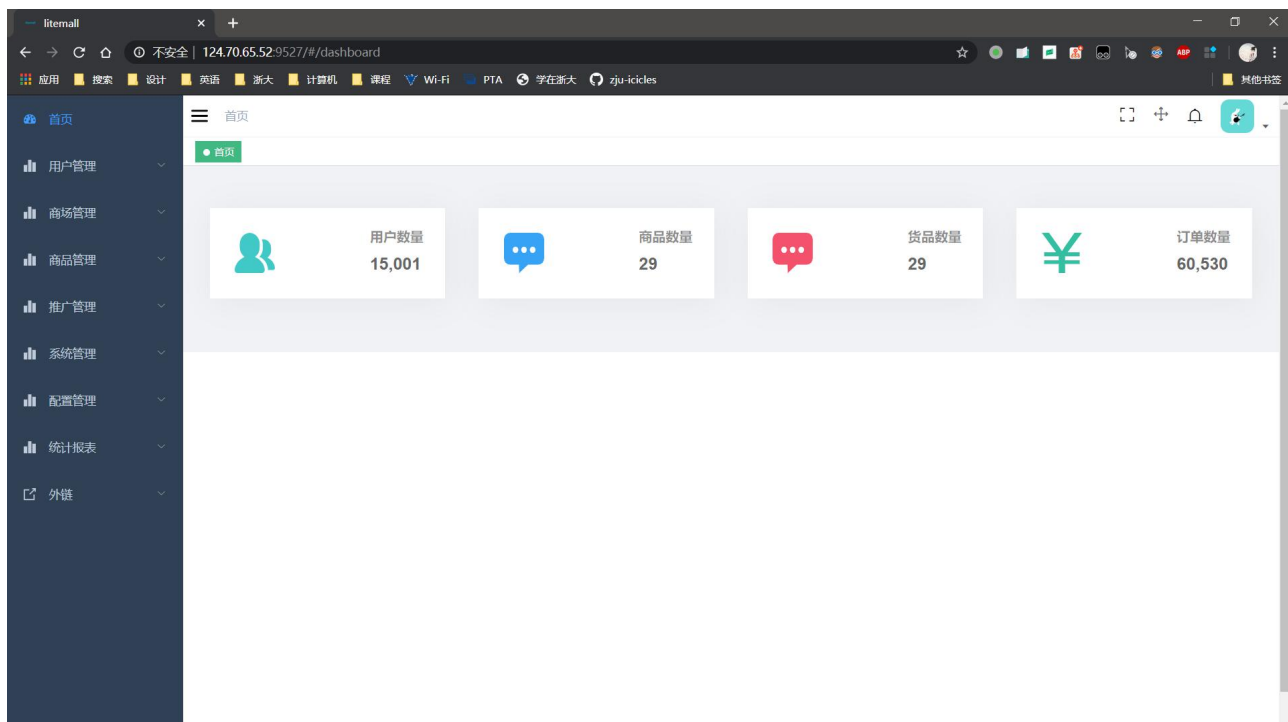
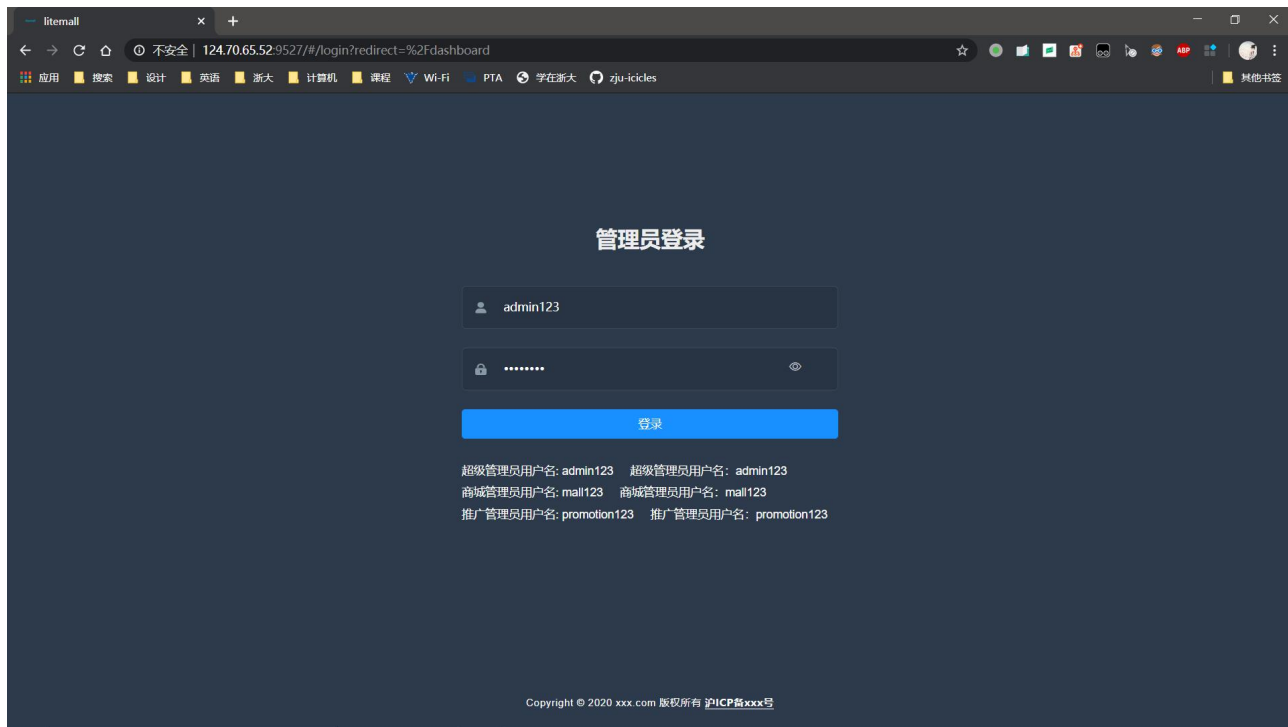
作业提交任务如下：

要求：截图中请包含个人的 ECS、IP 等信息，包括但不限于在浏览器中出现的公网 IP、ECS 个人命名等，若无相关信息，则忽略本条要求。

1. 通过 ecs-litemall-kunpeng 的弹性公网 IP 加上 6255 端口的形式访问轻商场前端截图。（20 分）



2. 通过 ecs-litemall-kunpeng 的弹性公网 IP 加上 9527 端口的形式访问管理后台截图。（20 分）



# 4

## 课堂思考题

挑战一：分析 Java 源码到应用的过程，如何使用 Javac 编译项目？（5 分）

Java 编译过程主要分为以下几步：

- 1、词法分析：把源代码中的字符（各个关键字、变量等）转为标记（Token）集合，单个字符的程序编写的最小单元，而 token 是编译过程的最小单元。
- 2、语法分析：将标记（Token）集合构造为抽象语法树。语法树的每一个节点都代表代码中的一个语法结构（如包、类型、接口、修饰符等等）。
- 3、填充符号表：符号表是有一组符号地址和符号信息构成的表格。填充符号表的过程的出口是一个待处理列表，包含了每一个抽象语法树（和 package-info.java）的顶级节点。
- 4、插入式注解处理器处理注解：注解处理器可以增删改抽象语法树的任意元素。因此每当注解处理器对语法树进行修改时，都将重新执行 1,2,3 步，直到注解处理器不再对语法树进行修改为止。每一次的循环过程都称为一次 Round。
- 5、语义分析：对语法树结构上正确的源程序进行上下文有关的审查。标注检查：包括是否变量声明、变量和赋值类型是否匹配等、常量折叠。数据和控制流分析：对程序上下文逻辑更进一步验证。包括变量使用前是否赋值、方法是否有返回值、异常是否被正确处理等。
- 6、解语法糖：把高级语法（如：泛型、可变参数、拆箱装箱等）转为基础语法结构，虚拟机运行时不支持这些高级语法。
- 7、生成字节码：把语法树、符号表里的信息转为字节码写到磁盘，同时进行少量的代码添加和转换工作。

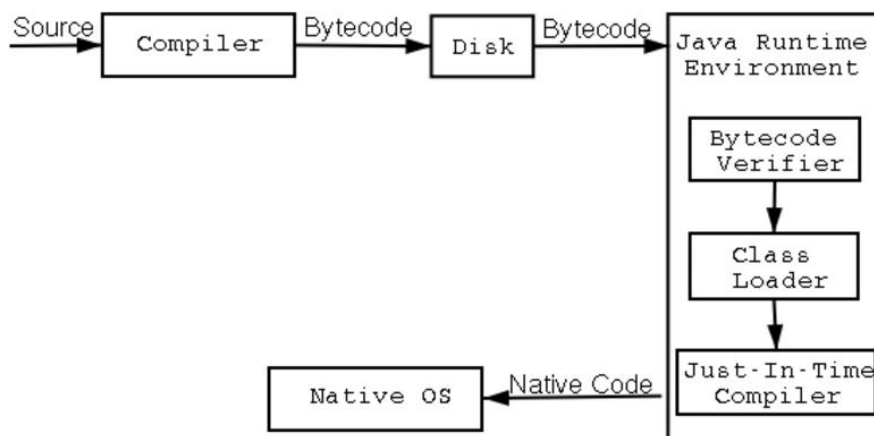


图 Java 代码编译过程

用 Javac 编译项目可以运行如下脚本程序：

```

#!/bin/bash
# 此处应该是项目文件夹所在目录
cur_dir=$(pwd)
  
```

```
echo $cur_dir
function compile(){
    # 记录项目的根目录所在路径
    project_name=javacDemo
    project_dir=$cur_dir/$project_name
    project_src=$project_dir/src # 源代码所在根目录
    project_lib=$project_dir/lib #依赖 jar 所在目录
    project_class=$project_dir/target # 编译后 class 文件存放根目录
    echo "begin compile"
    echo $project_dir
    echo $project_src
    echo $project_lib
    echo $project_class

    # src 目录下的所有 java 文件的名称存入到 项目根目录/src/sources.list 文件中 先检查是否存在, 如存在先删除
    rm -rf $project_src/sources.list
    # $project_src -name '*.java'表示在 $project_src 目录下以及子目录下寻找以.java 目录结尾的文件 并存放到 source.list 临时文件
    find $project_src -name '*.java' > $project_src/sources.list
    echo "java source file >>>"
    cat $project_src/sources.list

    # 构建存放编译好的 class 文件的基目录,先删除目录

    rm -rf $project_class
    mkdir $project_class

    # 组装 cp 参数
    # 将所有的 jar 文件绝对路径记录下来到 lib.list 文件中
    rm -rf $project_lib/lib.list
    find $project_lib -name '*.jar' > $project_lib/lib.list
    # 将当前目录添加进去
    cpvar=.:
    # 一行一行读取 lib.list 文件并去每行文件路径最终的文件名 ${line##*/}
    while read line
    do
        echo $line
        cpvar=${cpvar}${project_lib}/${line##*/}:"
        echo $cpvar
    done
}
```



```
done < $project_lib/lib.list

echo "print cpvar "
echo $cpvar
# 删除这个中间文件
rm -rf $project_lib/lib.list
# 截取 cpvar 最后一个字符:
# 获取 cpvar 字符串长度
length=${#cpvar}-1
# 取 0 - length 长度的字符串
cpvar=${cpvar:0:length}
echo $cpvar
# 批量编译 java 文件
# 编码: -encoding utf-8
# 依赖库以冒号:隔开
# -sourcepath 参数指定源码目录跟目录, @$project_src/sources.list 指定源码文件名
javac -d $project_class -encoding UTF-8 -cp $cpvar -g -sourcepath $project_src @$project_src
/sources.list

# 删除 sources.list 零时文件
rm -rf $project_src/sources.list

#删除存在的 jar 若编译过的话
# rm $qddemo/qddemo.jar
cd $project_class
jar -cvfm $project_class/${project_name}.jar $project_dir/MANIFEST.MF *
chmod a+x $project_class/${project_name}.jar

echo "将依赖包从"${project_lib}"复制到"${project_class}/lib"目录下. "
# 将依赖 jar 包从$project_lib 目录 复制到 $project_target/lib 目录下
cp -r $project_lib $project_class/lib
}
compile
exit 0
```

挑战二：npm run dev 和 npm run build 之间的区别是什么？如何配置？（5 分）

npm run dev 和 npm run build 之间的区别为：

- 1、package.json 里面"dev": "node build/dev-server.js","build": "node build/build.js",即运行 npm run dev 的时候执行的是 build/dev-server.js 文件，运行 npm run build 的时候执行的是 build/build.js 文件。
- 2、运行 npm run dev 执行的文件 build/dev-server.js，执行了：检查 node 和 npm 的版本，引入相关插件和配置，创建 express 服务器和 webpack 编译器，配置开发中间件（webpack-dev-middleware）和热重载中间件（webpack-hot-middleware），挂载代理服务和中间件，配置静态资源，启动服务器监听特定端口（8080），自动打开浏览器并打开特定网址（localhost:8080）。
- 3、运行 npm run build 执行的 build/build.js 文件用于构建环境下的配置，主要完成了：loading 动画，删除创建目标文件夹，webpack 编译，输出信息。

配置时要在 package.json 中配置如下：

```
"scripts": {  
  "dev": "node build/dev-server.js",  
  "build": "node build/build.js",  
  "unit": "karma start test/unit/karma.conf.js --single-run",  
  "e2e": "node test/e2e/runner.js",  
  "test": "npm run unit && npm run e2e",  
  "lint": "eslint --ext .js,.vue src test/unit/specs test/e2e/specs"  
},
```

有些项目中根据需要，还会配置其他命令，例如自动生成文档：

```
"build:doc": "node ./scripts/build-doc.js",
```

挑战三：如何实现应用、服务的开机启动？（5 分）

- 1、可以用 vim 编辑 /etc/rc.local，在文件最后一行添加要执行程序的全路径。
- 2、可以同过注册系统服务实现开机启动，命令如下：

```
chkconfig --add 服务名称  
chkconfig --level 启动级别 服务名 on
```

如果要查看哪些服务被添加为系统服务可以使用命令：ntsysv 或者 chkconfig --list