

ID 22031212122  
NAME xiaoning Shu  
TEAC zhiwei Zhang  
DATE 20230523



*Experimentation and practice of new generation information technology*

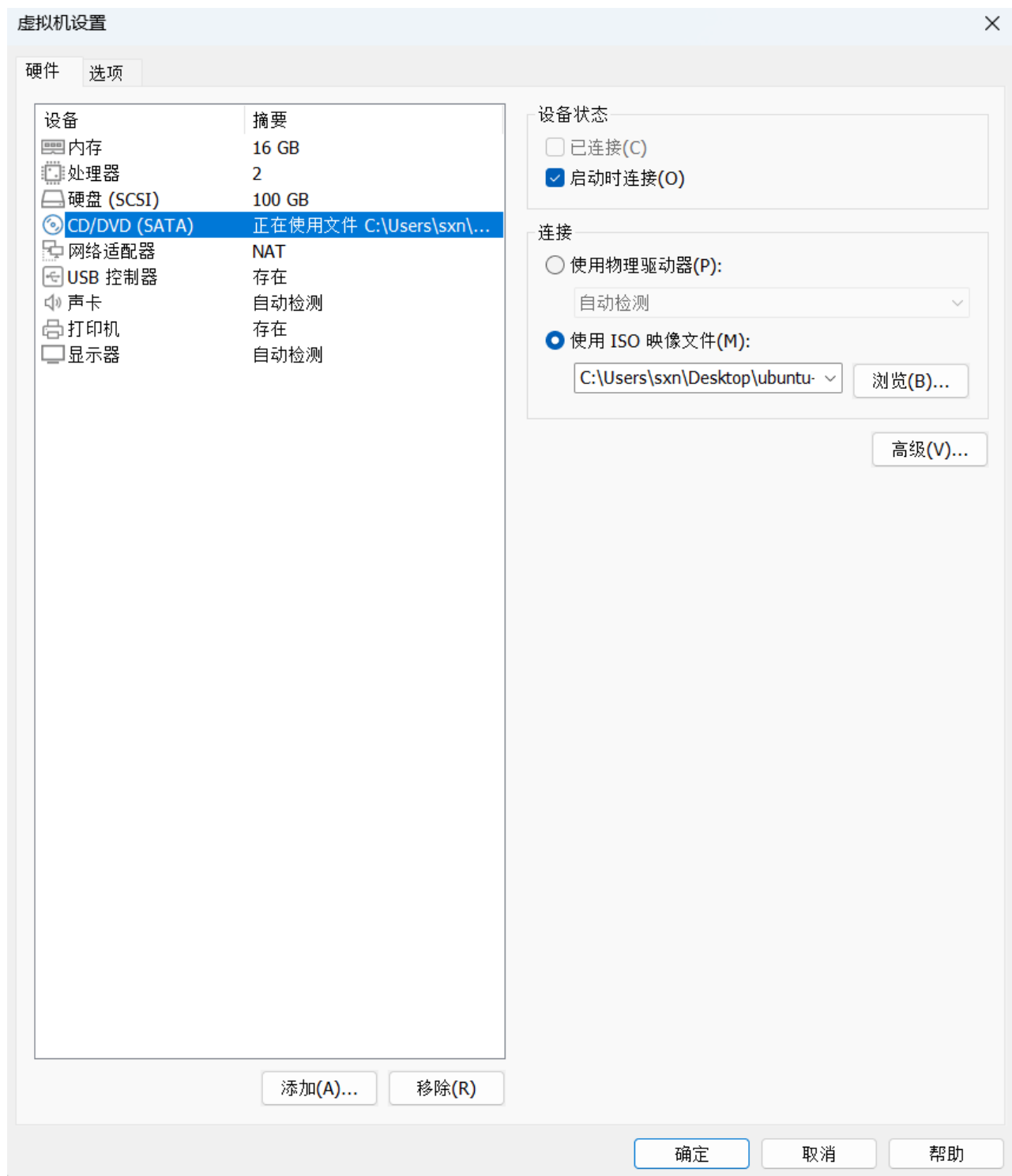
HOMEWORK 1 Virtualization experiments

虚拟机实验

在虚拟机实验中我们采用ubuntu-23.04-desktop-amd64.iso镜像。毕竟这个镜像比较炫酷。又是最新一代的Ubuntu系统值得一试。Web应用就采用以前写过的学生信息管理系统。因为装mysql在不同的虚拟机和镜像上虽然方便，但我还是喜欢租用云数据库，现在即便宜又好使，不需要额外的进行配置。直接点击产生数据库，然后写入不同的表即可。

1.实验配置

虚拟机软件	VMware Workstation Pro
镜像文件	ubuntu-23.04-desktop-amd64
内存	16GB
硬盘	100GB
CD/DVD (SATA)	ubuntu-23.04-desktop-amd64



直接开启虚拟机进行安装。

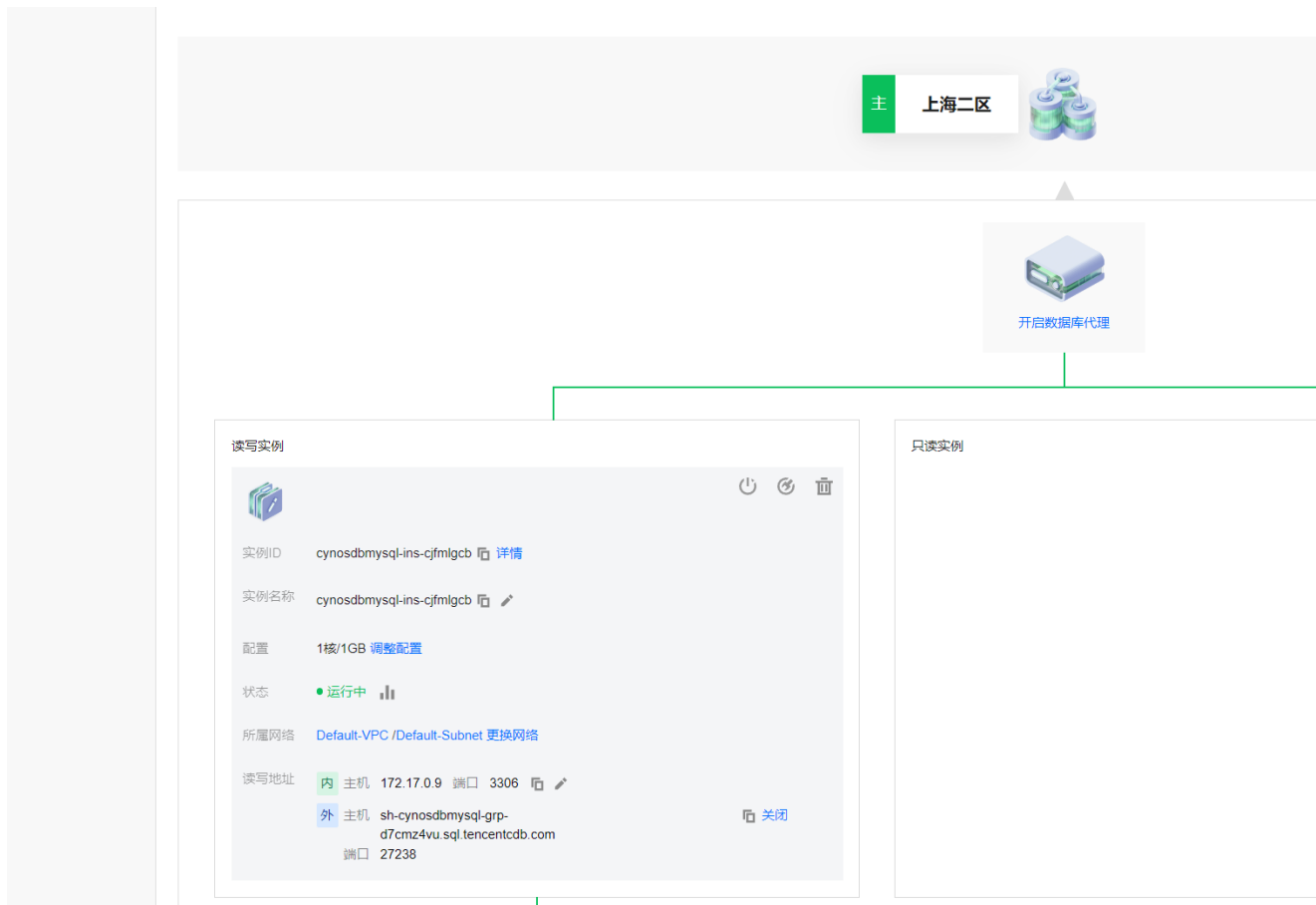


然后我们打开终端使用一些常用的命令进行软件的更新。

```
sudo apt update ## 一开始是一台空白的机子，我们需要更新
sudo apt install unzip ## 进行解压的软件包的安装
sudo apt install mysql-server 安装MySQL8的服务器版本
sudo snap install intellij-idea-community --classic ## 安装社区版idea
```

关于MySQL数据库的管理，我拒绝使用破解版的navicat for mysql，同时拒绝任何破解版的软件，自带的MySQL管理软件较为老土，所以我们选择云MySQL数据库。

管理分组												联系人管理
多个关键字用竖线“ ”分隔，多个过滤状态用回车键分隔												
实例 ID / 名称	集群 ID / 名称	状态	健康得分	异常告警	实例类型	地域	配置	引擎版本	接入点地址	接入来源	分组	操作
<input type="checkbox"/> cynosdbmysql-ins-cfmgcb cynosdbmysql-ins-cfmgcb	cynosdbmysql-ecqxluq	<div><div></div><div>巡检</div><div>✓ 概览</div></div>	100	0	主实例	上海	1核1000MB/10GB	MySQL 8.0	172.17.0.9:3306	腾讯云数据库	Default	<a href="#">诊断优化</a>



数据库内的内容如下所示：我们可以看到内部有很多表，学生，教师，超级管理员，同样可以作为一个开发模板，进行开发，这是我以前开发程序的样本。所以这次直接使用，没有任何问题。

schoolteammanage	首页	yirc_activites_apply   编辑表	yirc_activites   编辑表	yirc_user   编辑表	yirc_operator_log   编辑表	yirc_menu   编辑表
表	快速返回表名	新增	删除	导出	复制	删除
树状图	快速操作 >>	id	create_time	update_time	name	url
存储过程		1	2020-04-01 21:45:41	2020-04-02 22:11:52	系统设置	mdi-octagram
函数		5	2020-04-01 22:03:48	2020-06-30 15:50:08	系统管理	mdi-layers
触发器		6	2020-04-01 22:03:48	2020-06-30 15:56:59	角色管理	mdi-account
视图		15	2020-04-01 23:13:07	2020-06-30 15:56:41	添加	mdi-account-plus
事件		16	2020-04-02 22:22:26	2020-06-30 15:56:02	新增	mdi-plus
触发器		17	2020-04-02 22:27:01	2020-04-19 16:41:22	编辑	mdi-border-color
事件		18	2020-04-02 22:29:29	2020-06-30 15:56:13	删除	mdi-minus
触发器		19	2020-04-04 21:42:30	2020-06-30 15:56:48	编辑	mdi-account-check
事件		20	2020-04-04 21:42:51	2020-06-30 15:57:10	删除	mdi-account-remove
触发器		21	2020-04-04 21:46:00	2020-06-30 15:57:20	用户管理	mdi-account-multiple
事件		22	2020-04-04 21:46:53	2020-06-30 15:57:30	添加	mdi-account-multi...
触发器		23	2020-04-07 23:23:21	2020-06-30 15:57:42	编辑	mdi-account-edit
事件		24	2020-04-07 23:23:47	2020-06-30 15:57:56	删除	mdi-account-remove
触发器		26	2020-04-20 23:08:17	2020-06-30 15:59:00	上传图片	mdi-arrow-up-bold
事件		27	2020-04-22 22:16:02	2020-04-22 22:17:16	日志管理	mdi-calendar
触发器		28	2020-04-22 22:20:13	2020-04-22 23:59:10	删除	mdi-calendar-rem...
事件		29	2020-04-22 22:21:32	2020-04-23 00:00:56	清空日志	mdi-page-layout-h...
触发器		30	2020-04-23 20:35:54	2020-06-30 15:59:29	数据库备份	mdi-database
事件		31	2020-04-23 20:37:55	2020-06-30 15:59:44	备份	mdi-database-plus
触发器		32	2020-04-23 20:39:39	2020-06-30 15:59:57	删除	mdi-database-minus
事件		33	2020-04-25 12:49:35	2020-06-30 16:00:10	还原	mdi-database-plus
触发器		34	2020-06-21 14:13:13	2020-06-30 16:06:37	新闻管理	mdi-book-open-pa...
事件		35	2020-06-21 14:41:43	2020-06-30 16:06:46	添加	mdi-plus
触发器		36	2020-06-21 14:46:11	2020-06-30 16:07:00	编辑	mdi-pencil
事件		37	2020-06-21 14:50:11	2020-06-30 16:07:13	删除	mdi-window-minim...

然后主要看一下开发环境，主要是MySQL数据库方面的内容：内部包含MySQL的驱动，账号密码，以及链接的最长等待时间，和备份的路径。

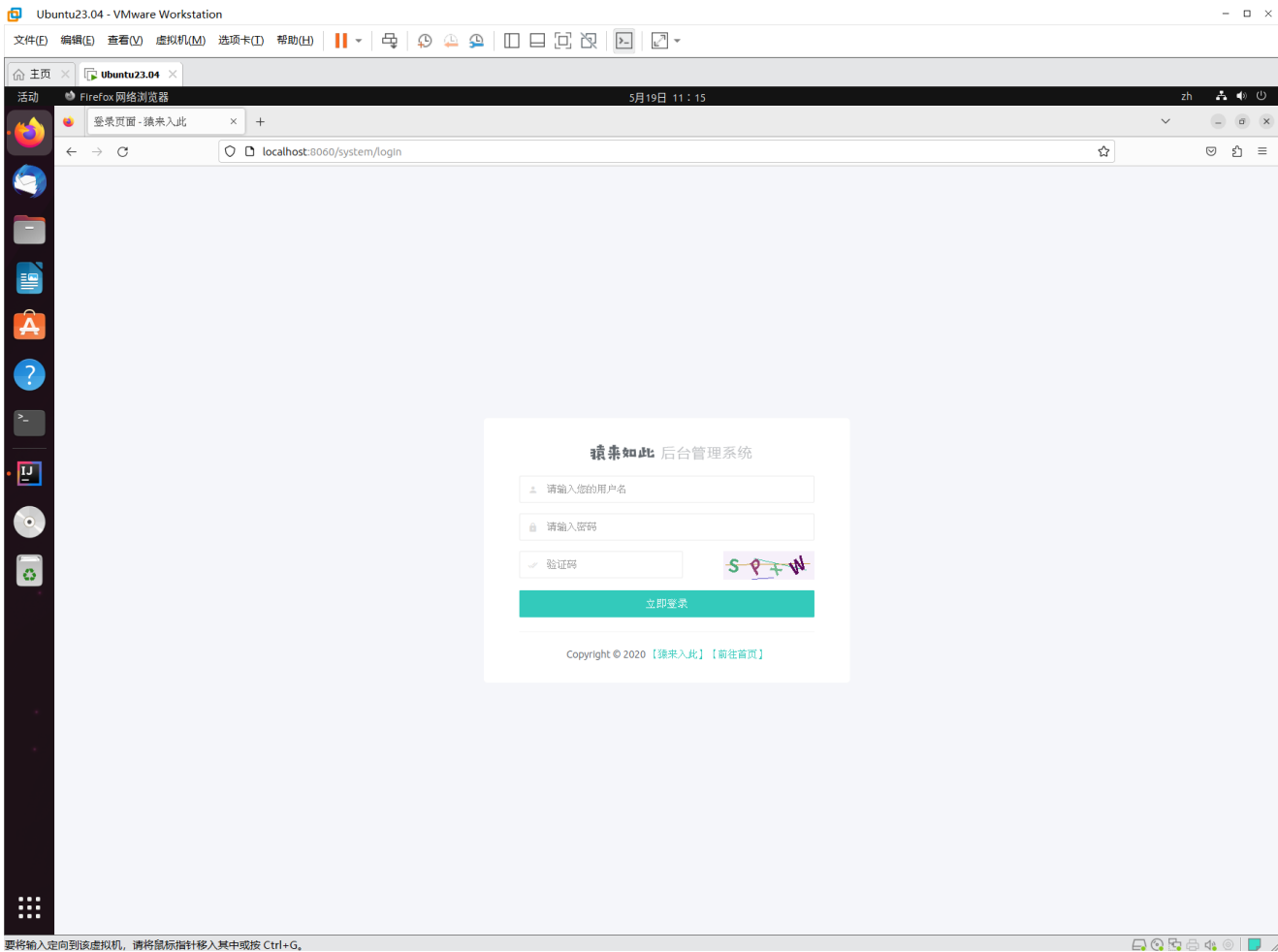
```
spring.datasource.url=jdbc:mysql://sh-cynosdbmysql-grp-d7cmz4vu.sql.tencentcdb.com:27238/schoolteammanage?useUnicode=true&characterEncoding=utf-8&serverTimezone=Asia/Shanghai
spring.datasource.username=root
spring.datasource.password=Shu123456!
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.tomcat.test-while-idle=true
spring.datasource.continue-on-error=false
server.port=8060
spring.freemarker.template-loader-path=classpath:/templates
spring.freemarker.cache=false
spring.freemarker.charset=UTF-8
spring.freemarker.check-template-location=true
spring.freemarker.content-type=text/html
spring.freemarker.expose-request-attributes=false
spring.freemarker.expose-session-attributes=false
spring.freemarker.request-context-attribute=request
spring.freemarker.suffix=.ftl
spring.freemarker.settings.template-exception-handler=ignore
server.servlet.session.timeout=1800
spring.jpa.database=MYSQL
# {u662F\u5426\u6253\u5378sq\u8BED\u53E5\u5230\u63A7\u5236\u53F0
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
# {u81EA\u5248\u68C0\u67E5\u5B9E\u64F3\u548C\u6570\u636E\u5E93\u8868\u662F\u5426\u4E00\u81F4\u4F8C\u5982\u679C\u4E0D\u4E00\u81F4\u5219\u4F1A\u8FDB\u884C\u66F4\u65B0\u6570\u636E\u5E93\u8868
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
# {u56FE\u7247\u4E0A\u4F20\u8BBE\u7F6E
jinku.upload.photo.suffix=.jpg,.png,.gif,.jpeg
# {u8BBE\u7F6E\u6587\u4EF6\u5927\u5C0F
spring.servlet.multipart.max-file-size=1MB
# {u6211\u4EEC\u81EA\u5B9A\u4E49\u8FC7\u68EE4\u6587\u4EF6\u5927\u5C0F
jinku.upload.photo.maxsize=1024
jinku.upload.photo.path=E:/workspace/team/src/main/resources/upload/
spring.http.encoding.charset=utf-8
# {u6570\u636E\u5E93\u5987\u4EFD\u8BBE\u7F6E
jinku.database.backup.dir=D:/dataBaseBak/
jinku.database.backup.username=root
jinku.database.backup.password=root
jinku.database.backup.database.name=schoolteammanage

spring.devtools.restart.enabled=true
spring.devtools.restart.additional-paths=src/main/java
```

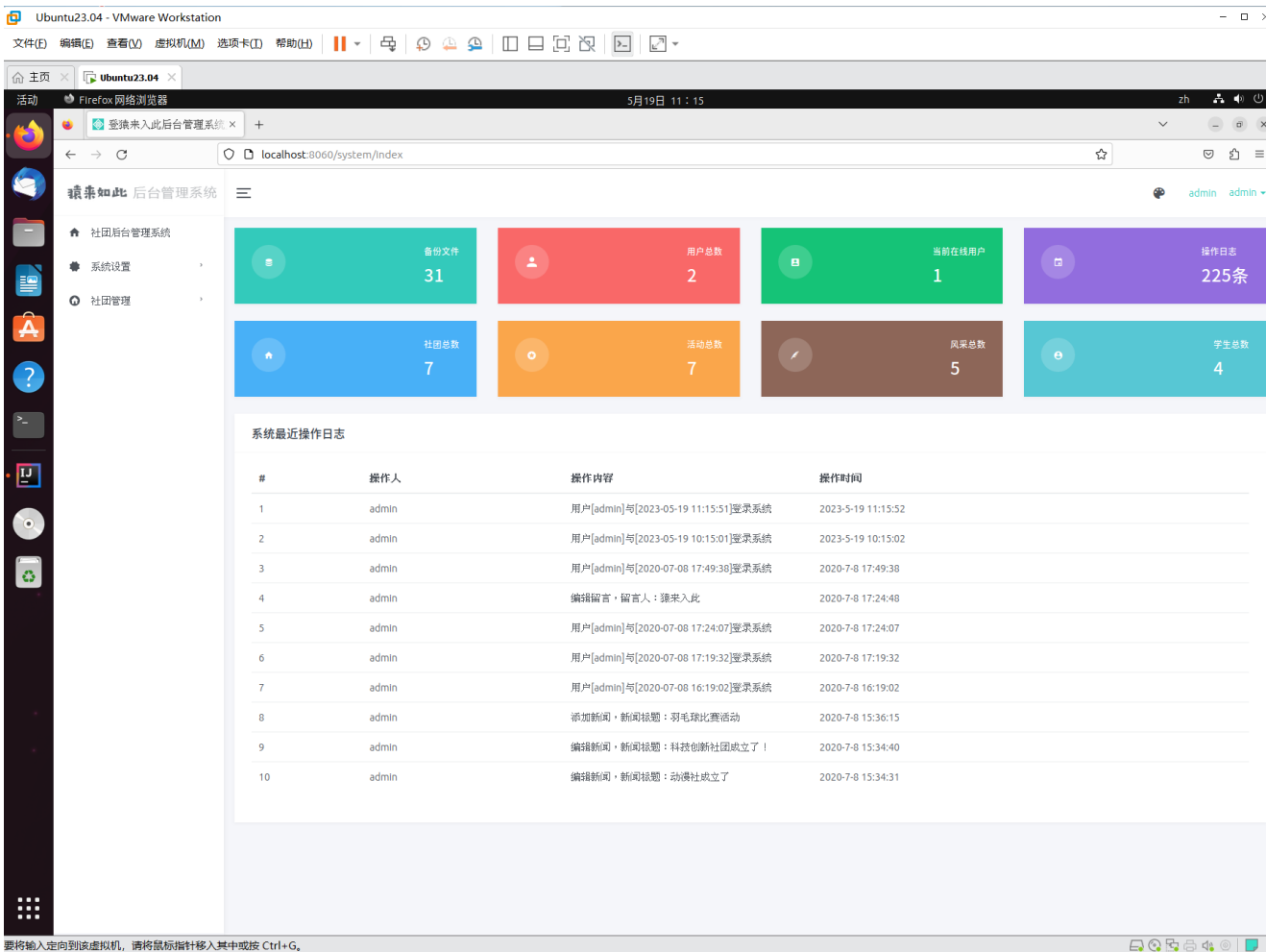
然后我们进行运行：

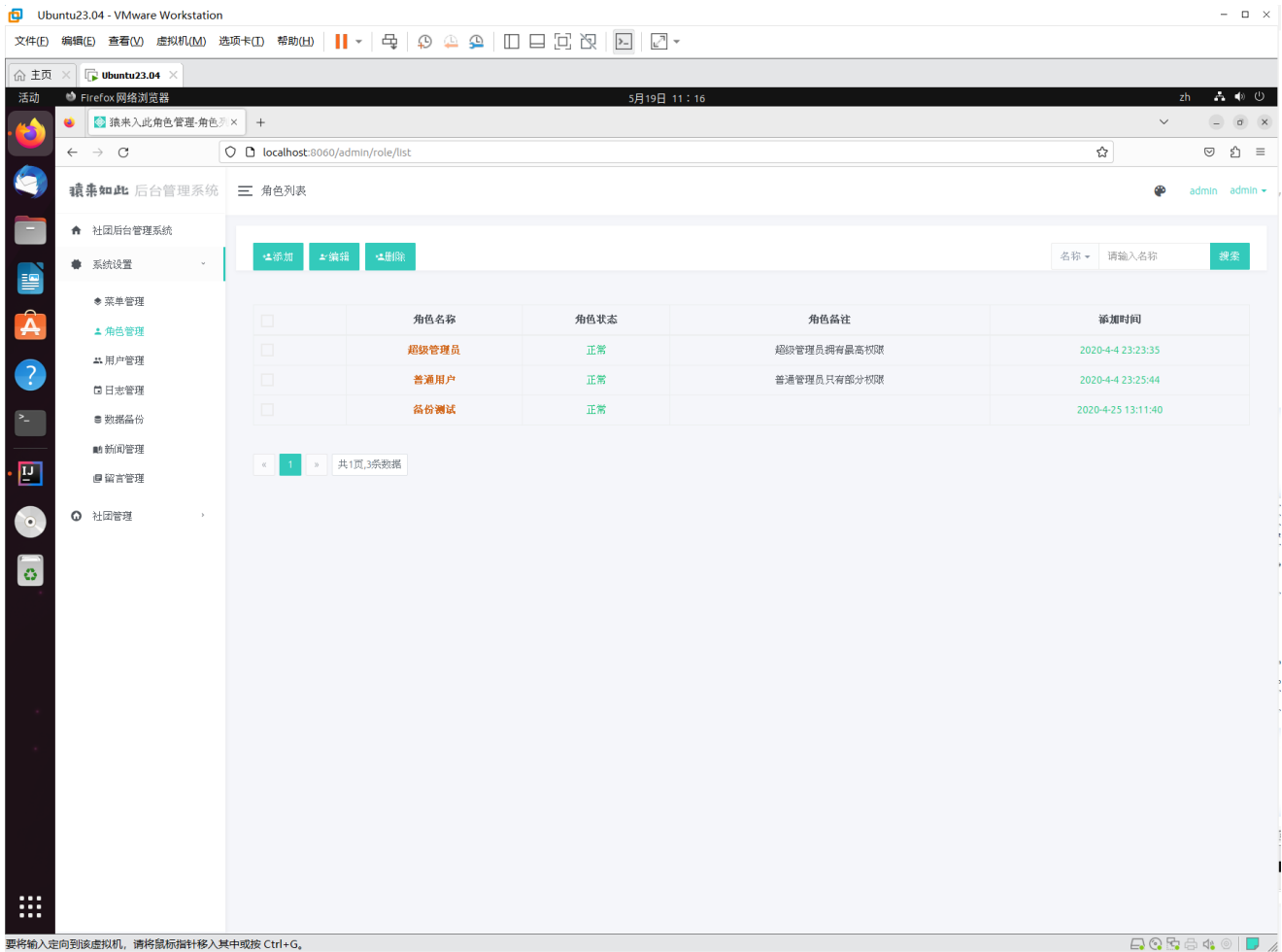
```
2023-05-19 11:12:21.610 INFO 4803 --- [ restartedMain] .RepositoryConfigurationExtensionSupport : Spring Data Redis - Could not safely identify store assignment for repository candidate interface com
2023-05-19 11:12:21.611 INFO 4803 --- [ restartedMain] .RepositoryConfigurationExtensionSupport : Spring Data Redis - Could not safely identify store assignment for repository candidate interface com
2023-05-19 11:12:21.612 INFO 4803 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 67ms. Found 0 Redis repository interfaces.
2023-05-19 11:12:24.797 INFO 4803 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8060 (http)
2023-05-19 11:12:24.840 INFO 4803 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-05-19 11:12:24.841 INFO 4803 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.31]
2023-05-19 11:12:25.255 INFO 4803 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-05-19 11:12:25.260 INFO 4803 --- [ restartedMain] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 8319 ms
2023-05-19 11:12:26.305 INFO 4803 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-05-19 11:12:26.573 INFO 4803 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.4.12.Final
2023-05-19 11:12:27.070 INFO 4803 --- [ restartedMain] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2023-05-19 11:12:27.513 INFO 4803 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-05-19 11:12:32.124 INFO 4803 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-05-19 11:12:32.167 INFO 4803 --- [ restartedMain] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
2023-05-19 11:12:35.871 INFO 4803 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000400: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal
2023-05-19 11:12:35.907 INFO 4803 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-05-19 11:12:35.965 INFO 4803 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-05-19 11:12:40.510 WARN 4803 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during v
2023-05-19 11:12:40.692 INFO 4803 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2023-05-19 11:12:42.559 INFO 4803 --- [ restartedMain] o.s.s.c.ThreadPoolTaskScheduler : Initializing ExecutorService 'taskScheduler'
2023-05-19 11:12:42.845 INFO 4803 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8060 (http) with context path ''
2023-05-19 11:12:42.853 INFO 4803 --- [ restartedMain] c.j.j.s.SpringBootJsJWebApplication : Started SpringBootJsJWebApplication in 28.215 seconds (JVM running for 30.856)
```

成功运行，本地端口为8060.我们在本地浏览器登录8060：



输入账号密码进行登录：





至此完美成功。

## 容器实验（基于Docker）

先在Ubuntu23.04镜像上安装docker：

## 卸载旧版本

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

## 更新apt包索引并安装包以允许apt通过 HTTPS 使用存储库：

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl gnupg
```

## 添加 Docker 的官方 GPG 密钥：

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/etc/apt/keyrings/docker.gpg
```

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

## 使用以下命令设置存储库：

```
echo \  
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/ubuntu \  
"
```

```
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
## 更新apt包索引:
```

```
sudo apt-get update
```

```
## 安装 Docker Engine、containerd 和 Docker Compose。
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-  
compose-plugin
```

```
## 通过运行映像验证 Docker 引擎安装是否成功 hello-world。
```

```
sudo docker run hello-world
```

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

显示我们docker安装成功。接下来我们来形成docker images.

```
##打包springboot项目为jar包
```

```
## 使用maven中的package，如果有问题，直接百度就能解决
```

```
## 设置源 能够加速编译
```

```
{  
"registry-mirrors":[https://drcy3veo.mirror.aliyuncs.com:]  
}
```

```
## 编写Dockerfile文件
```

```
## 用来初始化镜像 和 运行jar包，指定目录文件
```

```
FROM openjdk:11.0.2
```

```
MAINTAINER "shuxiaoningak@163.com"
```

```
ADD SchoolTeamManag.jar service_acl.jar
```

```
EXPOSE 8060
```

```
RUN bash -c 'touch /service_acl.jar'
```

```
ENTRYPOINT ["java", "-jar", "service_acl.jar"]
```

```
## 部署文件
```



将这三个文件放在同一个目录下 我是放在了/etc/docker目录下

## 制作镜像 执行下面命令，看好，最后面有个"."点！ 因为我没有在超级模式下对nn用户使用sudo 权限 所以我的命令全部都是加sudo的

```
sudo docker build -t springboot .
```

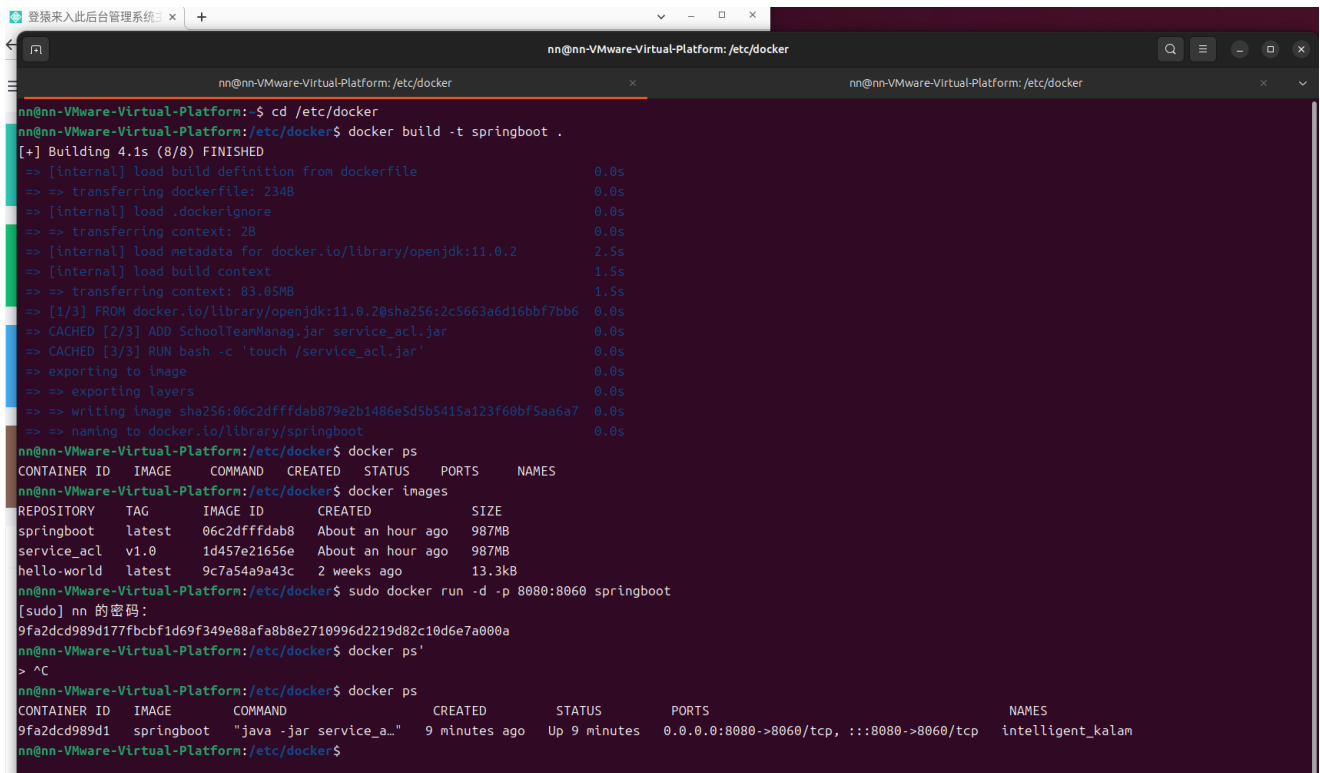
## 启动容器 -d 是后台运行不会产生很多的编译文件显示。因为上面已经有编译文件显示了 所以我直接不显示了

```
sudo docker run -d -p 8080:8060 springboot
```

这是启动容器后的显示。我们再看一下docker ps

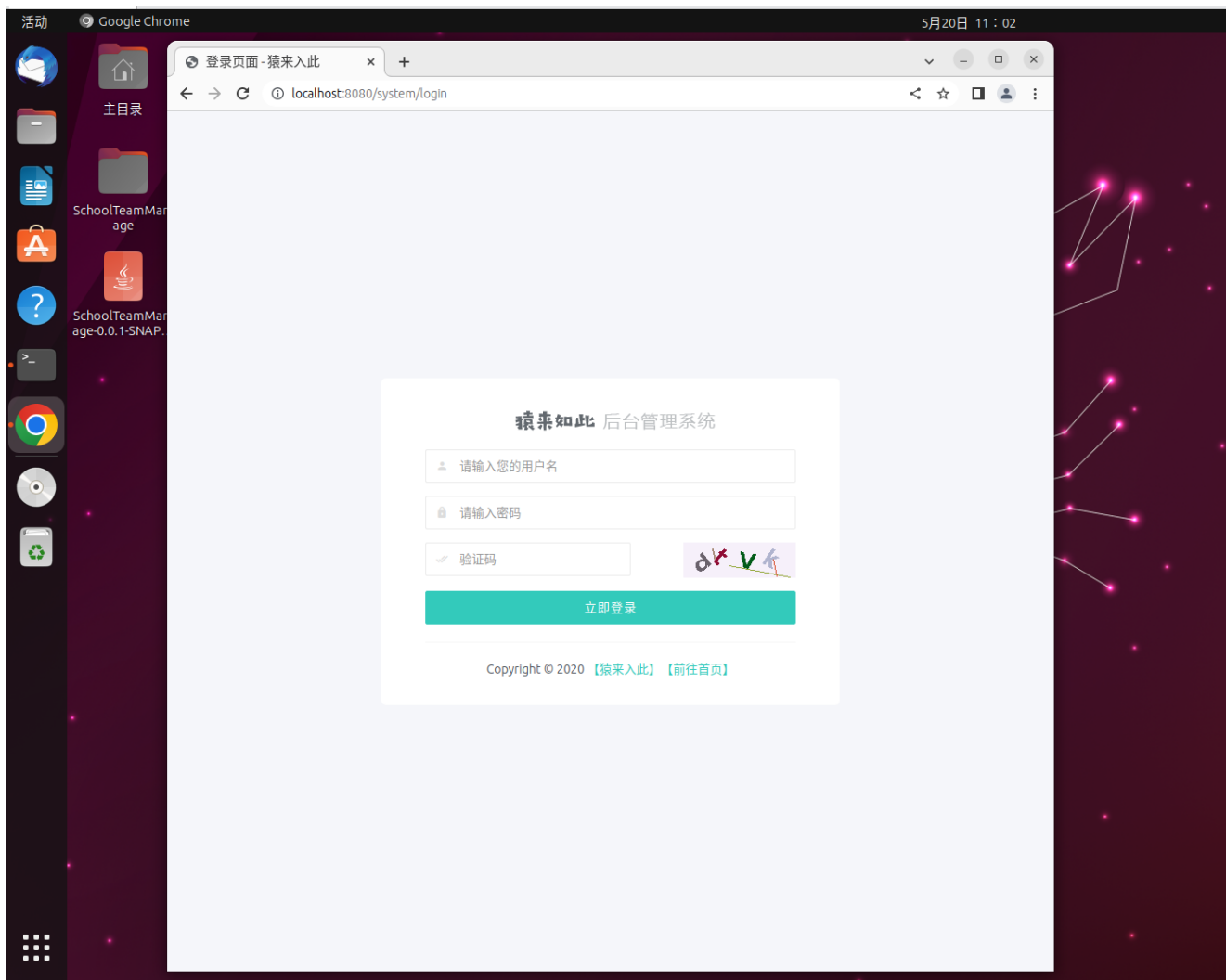
```
nn@nn-VMware-Virtual-Platform:/etc/docker$ sudo docker run -d -p 8080:8060 springboot
[sudo] nn 的密码：
9fa2dcd989d177fbcbf1d69f349e88afa8b8e2710996d2219d82c10d6e7a000a
nn@nn-VMware-Virtual-Platform:/etc/docker$
```

```
nn@nn-VMware-Virtual-Platform:/etc/docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
9fa2dcd989d1   springboot "java -jar service_a..." 9 minutes ago  Up 9 minutes  0.0.0.0:8080->8060/tcp, :::8080->8060/tcp  intelligent_kalam
nn@nn-VMware-Virtual-Platform:/etc/docker$
```

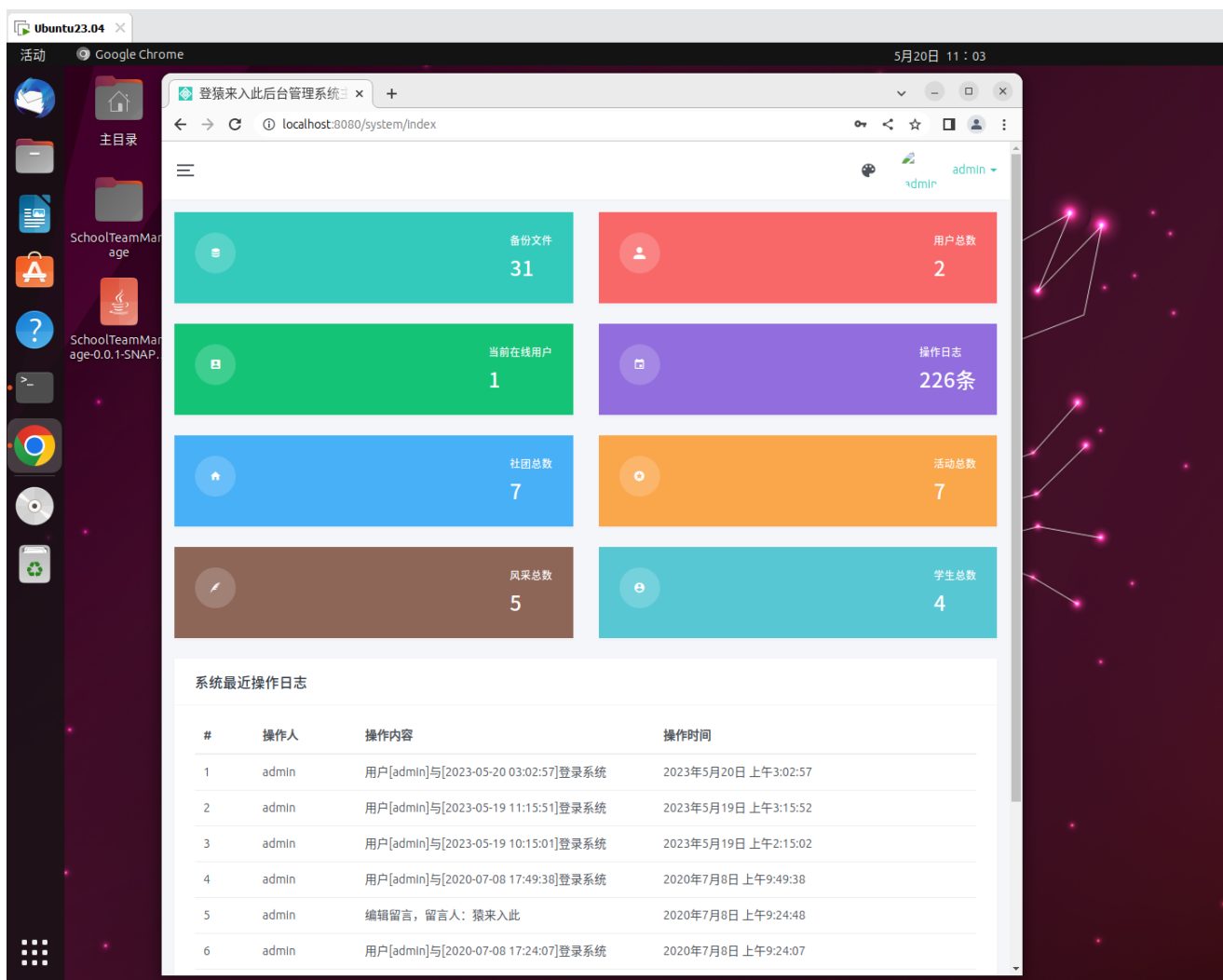


```
nn@nn-VMware-Virtual-Platform:/etc/docker$ cd /etc/docker
nn@nn-VMware-Virtual-Platform:/etc/docker$ docker build -t springboot .
[+] Building 4.1s (8/8) FINISHED
=> [internal] load build definition from dockerfile                                0.0s
=> => transferring dockerfile: 234B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/openjdk:11.0.2              2.5s
=> [internal] load build context                                                1.5s
=> => transferring context: 83.05MB                                             1.5s
=> [1/3] FROM docker.io/library/openjdk:11.0.2@sha256:2c5663a6d16bbf7bb6      0.0s
=> CACHED [2/3] ADD SchoolTeamManag.jar service_acl.jar                      0.0s
=> CACHED [3/3] RUN bash -c 'touch /service_acl.jar'                          0.0s
=> exporting to image                                                         0.0s
=> => exporting layers                                                         0.0s
=> => writing image sha256:06c2dffffdab879e2b1486e5d5b5415a123f60bf5aa6a7    0.0s
=> => naming to docker.io/library/springboot                                  0.0s
nn@nn-VMware-Virtual-Platform:/etc/docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
nn@nn-VMware-Virtual-Platform:/etc/docker$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
springboot    latest   06c2dffffdab8  About an hour ago  987MB
service_acl   v1.0     1d457e21656e  About an hour ago  987MB
hello-world   latest   9c7a54a9a43c  2 weeks ago     13.3kB
nn@nn-VMware-Virtual-Platform:/etc/docker$ sudo docker run -d -p 8080:8060 springboot
[sudo] nn 的密码：
9fa2dcd989d177fbcbf1d69f349e88afa8b8e2710996d2219d82c10d6e7a000a
nn@nn-VMware-Virtual-Platform:/etc/docker$ docker ps
> ^C
nn@nn-VMware-Virtual-Platform:/etc/docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
9fa2dcd989d1   springboot "java -jar service_a..." 9 minutes ago  Up 9 minutes  0.0.0.0:8080->8060/tcp, :::8080->8060/tcp  intelligent_kalam
nn@nn-VMware-Virtual-Platform:/etc/docker$
```

可以看到容器已经开始运行，已经有9分钟，并且本地端口与服务器端口进行了转接。此时我们应该登录localhost:8080链接



这是登录口令后的界面



至此已经全部完成。

这两个界面是完全不同的，在idea上是8060端口，在docker镜像上是8080端口，所以我没有敷衍作业。

一是虚拟机上进行web的开发，二是使用docker镜像对web进行部署。

## 比较分析虚拟机与容器两种技术的差异

VM 和容器都是用于为应用程序和服务提供独立且可移植的计算环境的虚拟化技术：

- VM 模拟整个计算机，包括虚拟化硬件、OS、用户模式及其自身的内核模式。VM 非常灵活，可为应用程序提供广泛的支持；但是，VM 往往比较大，会消耗主机资源。
- 容器（如前所述）基于主机操作系统内核构建，并包含打包的应用的独立用户模式进程。这有助于使容器轻量化且快速启动。

以我的理解，虚拟机是模拟一个完整的操作系统，里面内核与函数，应用等都是实打实的，而docker只是一个轻量化的虚拟机，他不需要进行操作系统的安装，只需要一个封包技术，将需要的配置进行打包，能够在不同的机子上快速传输，而虚拟机不能再不同的机子上快速传输。相当于一个项目，能够打包docker，另一台电脑直接pull images就可以直接运行，而虚拟机中却不能这样。

## 实验说明

本地或网络场景都可，操作系统、Web服务器等平台不限，虚拟机和容器软件与版本自选

Web应用的开发框架、语言等不限，Web应用完整度不做要求，可以正常访问即可，但是至少支持用户名-口令登录操作，用户名和口令存储在MySQL数据库中