2018 年全国职业院校技能大赛（高职组）

“云计算技术与应用”J 卷答案

场景描述

一、需求说明

某企业计划搭建私有云平台，以实现资源的池化弹性管理、企业应用的集中管理、统一安全认证和授权管理。按照给出的云平台架构进行 IaaS、PaaS、大数据系统部署及运维管理，并进行云存储网盘（包括 web 及安卓客户端）开发、大数据分析应用开发和微信小程序开发，最后提交工程文档。

二、云平台架构说明

赛项所采用的云计算系统架构如图 1 所示，IP 地址规划如表 1 所示。

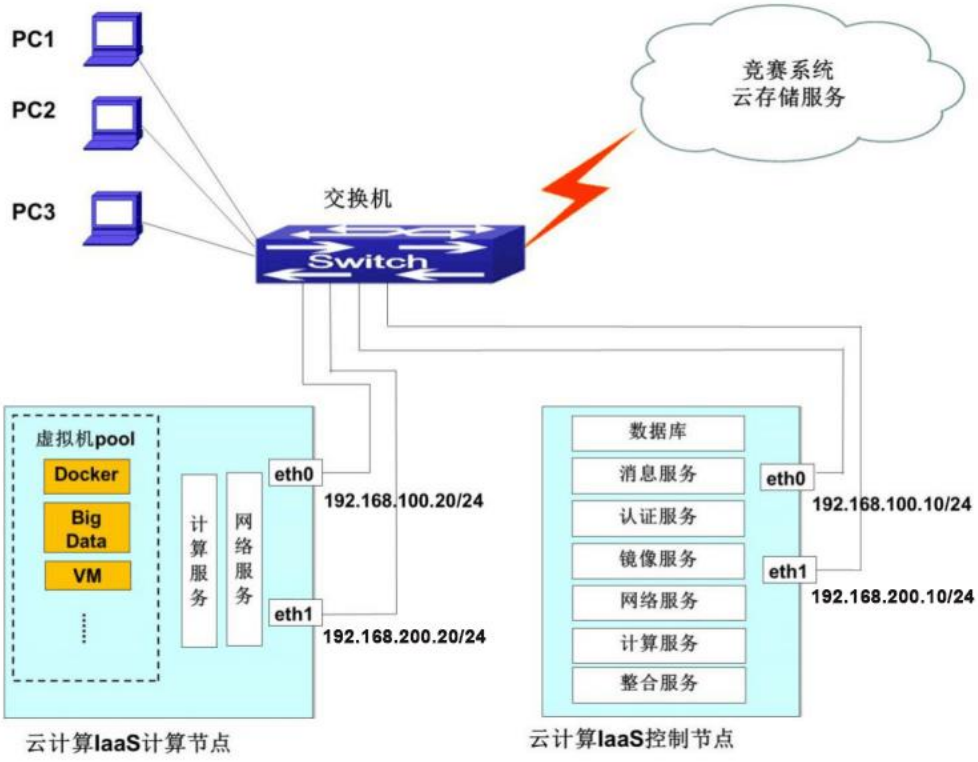


图 1 云计算系统架构图

表 1 IP 地址规划表

设备名称	接口	IP 地址	说明
控制节点服务器	eth0	192.168.100.10/24	Vlan 100
	eth1	192.168.200.10/24（初始 IP）	Vlan 200
计算节点服务器	eth0	192.168.100.20/24	Vlan 100
	eth1	192.168.200.20/24（初始 IP）	Vlan 200
PC-1	本地连接	172.16.x.2/16	Vlan 1
PC-2	本地连接	172.16.x.3/16	Vlan 1
PC-3	本地连接	172.16.x.4/16	Vlan 1
交换机	Vlan 1	172.16.x.1/16	
	Vlan 100	192.168.100.1/24	
	Vlan 200	192.168.200.1/24	

注：表中的 x 为考位号；根据以上信息，检查硬件连线及网络设备配置，确保网络连接正常。

第一部分：IaaS 云计算基础架构平台（共 15 分）

任务一、IaaS 云平台搭建（15 分）

修改云平台 IaaS 各节点的系统配置，按以下步骤搭建云平台，并完成相应的答题。

1.操作系统环境配置（1 分）

按以下要求设置主机名、防火墙及 SELinux：

（1）设置控制节点主机名为 controller，计算节点主机名为 compute；

（2）关闭控制节点和计算节点的防火墙，设置开机不启动；

（3）设置控制节点和计算节点的 SELinux 为 Permissive 模式；

（4）退出 SecureCRT，重新通过 ssh 连接各节点服务器；

使用命令查询控制节点和计算节点的主机名、防火墙是否处于关闭状态及 SELinux 的状态。以文本形式依次将命令行及查询信息提交到答题框。

```
[root@controller ~]# hostname
controller
[root@compute ~]# hostname
```

compute [root@controller ~]#systemctl status firewalld ● firewalld.service - firewalld - dynamic firewall daemon Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled) Active: inactive (dead) Docs: man:firewalld(1) [root@controller ~]# getenforce Permissive
查到两个主机名各得 0.25 分 查到防火墙状态得 0.25 分 查到 SELinux 模式为 permissive 得 0.25 分

2.上传镜像源并挂载（1 分）

通过 SecureFX 上传两个镜像文件 CentOS-7-x86_64-DVD-1511.iso 和 XianDian-IaaS-v2.2.iso 到控制节点的 opt 目录下；在 opt 目录下创建两个子目录 centos 和 iaas，并将镜像文件对应挂载到上述两个目录下；使用 df 命令查看挂载的信息（需显示挂载的文件系统类型）。依次将操作命令及执行结果以文本形式提交到答题框。

[root@controller ~]# mkdir /opt/{centos，iaas}
[root@controller ~]# mount -o loop CentOS-7-x86_64-DVD-1511.iso /opt/centos/
mount: /dev/loop0 is write-protected, mounting read-only
[root@controller ~]# mount -o loop XianDian-IaaS-v2.2.iso /opt/iaas/
mount: /dev/loop1 is write-protected, mounting read-only
[root@controller ~]# df -Th
Filesystem Type Size Used Avail Use% Mounted on
/dev/vda1 xfs 50G 7.6G 43G 16% /
devtmpfs devtmpfs 984M 0 984M 0% /dev
tmpfs tmpfs 1001M 0 1001M 0% /dev/shm
tmpfs tmpfs 1001M 17M 985M 2% /run
tmpfs tmpfs 1001M 0 1001M 0% /sys/fs/cgroup

/dev/loop0	iso9660	4.1G	4.1G	0 100% /opt/centos
/dev/loop1	iso9660	2.7G	2.7G	0 100% /opt/iaas

完成挂载，有正确的挂载返回信息得 0.5 分

使用 df 命令显示文件系统类型得 0.5 分

3.配置本地以及远程 yum 源（1 分）

配置控制节点本地 yum 源文件 local.repo，搭建 ftp 服务并配置根目录为指向存放 yum 源的路径；配置计算节点 yum 源文件 ftp.repo，使用控制节点 ftp 服务作为 yum 源，其中节点的地址以主机名表示；使用 cat 命令查看控制节点和计算节点的 yum 源全路径配置文件。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@controller ~]# cat /etc/yum.repos.d/local.repo
[centos]
name=centos
baseurl=file:///opt/centos
enabled=1
gpgcheck=0
[iaas]
name=iaas
baseurl=file:///opt/iaas/iaas-repo
enabled=1
gpgcheck=0
[root@compute ~]# cat /etc/yum.repos.d/ftp.repo
[centos]
name=centos
baseurl=ftp://controller/centos
enabled=1
gpgcheck=0
[iaas]
name=iaas
```

baseurl=ftp://controller/iaas/iaas-repo enabled=1 gpgcheck=0
配置正确 controller 节点 local.repo 中的 baseurl 得 0.5 分 配置正确 compute 节点 ftp.repo 中的 baseurl 得 0.5 分

4.环境变量配置（1 分）

在控制节点和计算节点分别安装 iaas-xiandian 软件包，根据表 2 完成脚本文件 openrc.sh 的配置。以文本形式提交脚本文件的内容到答题框中。

表 2 变量配置表

服务	变量	参数/密码
Mysql	root	000000
	Keystone	000000
	Glance	000000
	Nova	000000
	Neutron	000000
	Heat	000000
	Trove	000000
Keystone	DOMAIN_NAME	demo
	Admin	000000
	Rabbit	000000
	Glance	000000
	Nova	000000
	Neutron	000000
	Heat	000000
	Trove	000000
Neutron	Metadata	000000
	External Network	enp9s0

[root@controller opt]# cat /etc/xiandian/openrc.sh ##------------------system config----------------## ##Controller Server Manager IP. example:x.x.x.x

```
HOST_IP=192.168.100.10

##Controller Server hostname. example:controller
HOST_NAME=controller

##Compute Node Manager IP. example:x.x.x.x
HOST_IP_NODE=192.168.100.20

##Compute Node hostname. example:compute
HOST_NAME_NODE=compute

##--------------------Rabbit config ----------------##
##user for rabbit. example:openstack
RABBIT_USER=openstack

##Password for rabbit user .example:000000
RABBIT_PASS=000000

##--------------------MySQL config--------------------##
##Password for MySQL root user . exmaple:000000
DB_PASS=000000

##--------------------Keystone config----------------##
##Password for Keystore admin user. exmaple:000000
DOMAIN_NAME=demo
ADMIN_PASS=000000
DEMO_PASS=000000

##Password for Mysql keystore user. exmaple:000000
KEYSTONE_DBPASS=000000
```

```
##------------------Glance config-----------------##
##Password for Mysql glance user. exmaple:000000
GLANCE_DBPASS=000000

##Password for Keystore glance user. exmaple:000000
GLANCE_PASS=000000

##------------------Nova config-----------------##
##Password for Mysql nova user. exmaple:000000
NOVA_DBPASS=000000

##Password for Keystore nova user. exmaple:000000
NOVA_PASS=000000

##------------------Neturon config-----------------##
##Password for Mysql neutron user. exmaple:000000
NEUTRON_DBPASS=000000

##Password for Keystore neutron user. exmaple:000000
NEUTRON_PASS=000000

##metadata secret for neutron. exmaple:000000
METADATA_SECRET=000000

##External Network Interface. example:eth1
INTERFACE_NAME=enp9s0

##First Vlan ID in VLAN RANGE for VLAN Network. exmaple:101
minvlan=101

##Last Vlan ID in VLAN RANGE for VLAN Network. example:200
```

```
maxvlan=200

##-------------------Cinder config--------------------##
##Password for Mysql cinder user. exmaple:000000
CINDER_DBPASS=000000

##Password for Keystore cinder user. exmaple:000000
CINDER_PASS=000000

##Cinder Block Disk. example:md126p3
BLOCK_DISK=md126p4

##-------------------Swift config---------------------##
##Password for Keystore swift user. exmaple:000000
SWIFT_PASS=000000

##The NO1. NODE Object Disk for Swift. example:md126p4. The 2nd will be OBJECT_DISK_2
OBJECT_DISK=md126p5

##The NO1. NODE IP for Swift Storage Network. example:x.x.x.x. The 2nd will be
STORAGE_LOCAL_NET_IP_2
STORAGE_LOCAL_NET_IP=192.168.100.20

##-------------------Heat config----------------------##
##Password for Mysql heat user. exmaple:000000
HEAT_DBPASS=000000

##Password for Keystore heat user. exmaple:000000
HEAT_PASS=000000

##-------------------Ceilometer config----------------##
```

##Password for Mysql ceilometer user. exmaple:000000 CEILOMETER_DBPASS=000000 ##Password for Keystore ceilometer user. exmaple:000000 CEILOMETER_PASS=000000 ##----------AODH config----------## ##Password for Mysql AODH user. exmaple:000000 AODH_DBPASS=000000 ##Password for Keystore AODH user. exmaple:000000 AODH_PASS=000000
配置正确的 domain 得 0.5 分 配置正确的密码得 0.5 分

5.数据库安装（2 分）

使用脚本安装数据库服务，使用 root 用户登录数据库，查看 mysql 的默认存储引擎信息、mysql 支持的存储引擎有哪些。依次将操作命令和查询结果以文本形式提交到答题框。

[root@controller ~]# iaas-install-mysql.sh [root@cloud-controller ~]# mysql -uroot -p000000 Welcome to the MariaDB monitor. Commands end with ; or \g. Your MariaDB connection id is 20952 Server version: 10.1.17-MariaDB MariaDB Server Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement. MariaDB [(none)]> show variables like "storage_engine";

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| storage_engine | InnoDB |
+---------------+-------+
1 row in set (0.00 sec)

MariaDB [(none)]> show variables like "have%";
+-------------------+---------+
| Variable_name | Value |
+-------------------+---------+
| have_compress | YES |
| have_crypt | YES |
| have_dynamic_loading | YES |
| have_geometry | YES |
| have_openssl | YES |
| have_profiling | YES |
| have_query_cache | YES |
| have_rtree_keys | YES |
| have_ssl | DISABLED |
| have_symlink | DISABLED |
+-------------------+---------+
10 rows in set (0.00 sec)
```

查到默认的存储引擎 InnoDB 得 1 分

查到支持的存储引擎得 1 分

6.keystone 安装（2 分）

使用脚本安装 keystone 服务，在 keystone 中创建用户 testuser，密码为 password。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@xiandian ~]# source /etc/keystone/admin-openrc.sh
[root@xiandian ~]# openstack user create --domain demo --password password testuser
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 3ac89594c8e944a9b5bb567fca4e75aa |
| enabled   | True                             |
| id        | fb1f3175d3ef4974a12c8482307c8e24 |
| name      | testuser                         |
+-----------+----------------------------------+
```

使用命令正确的创建用户得 1 分

返回结果正确（除了 ID）得 1 分

7.glance 安装（2 分）

使用脚本安装 glance 服务。首先使用 glance 相关命令上传镜像，镜像源为 CentOS_6.5_x86_64_XD.qcow2，名为 testone；其次使用 openstack 命令修改这个镜像名为 examimage；最后使用 openstack 命令查看镜像列表。依次将操作命令与返回结果以文本形式提交到答题框。

```
[root@xiandian ~]# glance image-create --name "testone" --disk-format "qcow2"
--container-format bare --progress < /root/CentOS_6.5_x86_64_XD.qcow2
[=============================>] 100%
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 3e565ace16066679ea363dde5411ed25     |
| container_format | bare                                 |
| created_at       | 2018-01-17T09:01:36Z                 |
| disk_format      | qcow2                                |
| id               | 3bb63ae0-3129-442b-b19f-9f66298132aa |
```

```
| min_disk          | 0                                    |
| min_ram           | 0                                    |
| name              | testone                              |
| owner             | 0ab2dbde4f754b699e22461426cd0774     |
| protected         | False                                |
| size              | 283181056                            |
| status            | active                               |
| tags              | []                                   |
| updated_at        | 2018-01-17T09:01:38Z                 |
| virtual_size      | None                                 |
| visibility        | private                              |
+-----------------+--------------------------------------+
[root@xiandian ~]# openstack image set testone --name examimage
[root@xiandian ~]# openstack image list
+--------------------------------------+-----------+--------+
| ID                                   | Name      | Status |
+--------------------------------------+-----------+--------+
| 3bb63ae0-3129-442b-b19f-9f66298132aa | examimage | active |
+--------------------------------------+-----------+--------+
```

使用正确命令上传镜像得 1 分

使用 image list 查询到的镜像名为 examimage 得 1 分

8.nova 安装（2 分）

使用脚本安装 nova 服务。查询资源使用情况。依次将操作命令和返回结果以文本形式提交到答题框。

```
[root@controller ~]# nova usage-list
Usage from 2017-12-19 to 2018-01-17:
+----------------------------+---------+-------------+----------+--------------+
```

```
| Tenant ID                        | Servers | RAM MB-Hours | CPU Hours | Disk
GB-Hours |
+----------------------------------+---------+--------------+-----------+--------------+
| 38e4fd41edaf40189d152dda18935b97 | 121     | 9403874.30   | 4575.92   | 91946.84
|
+----------------------------------+---------+--------------+-----------+--------------+
```

使用命令查询使用情况得 1 分

有正确的返回得 1 分

9.网络创建（2 分）

使用脚本安装 neutron 服务，并配置为 GRE 网络：

（1）创建云主机外部网络为 ext-net，子网为 ext-subnet，虚拟机浮动 IP 网段为 192.168.200.0/24，网关为 192.168.200.1;

（2)创建云主机隧道网络 int-net1，子网为 int-subnet1，虚拟机子网 IP 网段为 10.0.0.0/24，网关为 10.0.0.1；

（3)创建云主机隧道网络 int-net2，子网为 int-subnet2，虚拟机子网 IP 网段为 10.0.1.0/24，网关为 10.0.1.1;

（4）添加名为 ext-router 的路由器，配置路由接口地址，完成隧道网络 int-net1 和外部网络 ext-net 的连通。

使用 neutron 相关命令查询所创建路由的详细信息。依次将操作命令和返回结果以文本形式提交到答题框。

```
[root@controller ~]# neutron router-list
+--------------------------------------+-------+-----------------------------------------------------------
----------+-----------+-------+
| id                                   | name  | external_gateway_info
| distributed | ha    |
+--------------------------------------+-------+-----------------------------------------------------------
----------+-----------+-------+
```

```
| 9d0e62e4-6340-4686-8d99-3fc715166302 | route | {"network_id":
"49af02fe-8496-4337-9817-eee1a72feef0", "enable_snat": true, | False | False |
| | | "external_fixed_ips": [{"subnet_id":
"1e097cb7-9630-418d-bfd1-57e80354b39a", | | |
| | | "ip_address": "172.30.24.3"}]}
| | |
+--------------------------------------+-------+-----------------------------------------------------
----------+-----------+-------+
[root@controller ~]# neutron router-show 9d0e62e4-6340-4686-8d99-3fc715166302
+-----------------------+-----------------------------------------------------------------------------
-----------------------------------------------------------+
| Field | Value
|
+-----------------------+-----------------------------------------------------------------------------
-----------------------------------------------------------+
| admin_state_up | True
|
| availability_zone_hints |
|
| availability_zones | nova
|
| description |
|
| distributed | False
|
| external_gateway_info | {"network_id": "49af02fe-8496-4337-9817-eee1a72feef0",
"enable_snat": true, "external_fixed_ips": [{"subnet_id":
"1e097cb7-9630-418d-bfd1-57e80354b39a", |
| | "ip_address": "172.30.24.3"}]}
|
| ha | False
```

```
|
| id                                    | 9d0e62e4-6340-4686-8d99-3fc715166302
|
| name                                                    | route
|
| routes                                                                |
|
| status                                                  | ACTIVE
|
| tenant_id                             | 38e4fd41edaf40189d152dda18935b97
|
+---------------------+-------------------------------------------------------------------
-------------------------------------------------------+
```

正确的查询命令查询路由器列表得 1 分

返回结果有正确的名字和状态得 1 分

10.trove 配置（1 分）

使用脚本在控制节点安装 trove 服务，链接的网络为 int-net1。完成后查询所有的数据库实例列表。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@controller ~]# trove list
+--------------------------------------+--------+----------+----------------+--------+-----------+------+
| ID                                   | Name   | Datastore | Datastore Version | Status | Flavor ID
| Size |
+--------------------------------------+--------+----------+----------------+--------+-----------+------+
| a564b6de-12a6-4b70-b1e6-af2e9395615f | mysql-1 | mysql     | mysql-5.6         | ACTIVE |
2          |     5 |
+--------------------------------------+--------+----------+----------------+--------+-----------+------+
```

正确的查询命令得 0.5 分 正确的返回结果得 0.5 分

第二部分：PaaS 服务平台（共 5 分）

任务一、PaaS 平台搭建（5 分）

1.容器平台搭建（2 分）

容器平台架构如图 2 所示。

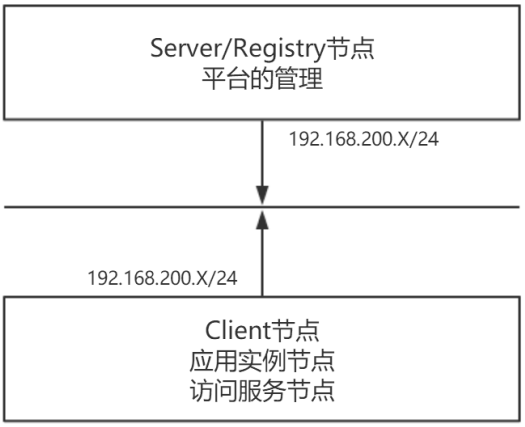


图 2 容器平台架构

根据 PaaS 平台的部署架构，PaaS 平台部署在 IaaS 平台的 2 台云主机上。其中 Server 云主机上部署 Server 节点和 Registry 节点，Client 云主机上部署 Client 节点。每个云主机配置如下：

（1）系统配置

Server/Registry 节点：2CPU，4G 内存，60G 硬盘；

Client 节点：2CPU，4G 内存，60G 硬盘；

（2）操作系统：centos_7-x86_64；

（3）IP：Server 和 Client 云主机 ip 动态分配；

（4）Server 节点的主机名：Server；

（5）Client 节点的主机名：Client。

根据配置要求，完成配置文件的自定义与修改，搭建容器平台。登录容器平台，通过

curl 命令查询 Rancher 管理平台首页。依次将操作命令及返回结果以文本形式提交到答题框。

[root@server images]# curl http://10.0.6.126:8080/env/1a5/apps/stacks Loading
正确的 curl 地址（除了 IP）得 1 分 正确的返回结果得 1 分

2.应用模板部署（3 分）

根据提供的软件包，通过“应用商店”部署 Gogs，修改网页访问端口为 9093，通过 curl 命令访问 Gogs 用户列表。依次将操作命令及返回结果以文本形式提交到答题框。

[root@server images]# curl http://10.0.6.127:9093/explore/users Explore - Gogs Please enable JavaScript in your browser!
正确的 curl 地址（除了 IP）得 1 分 正确的返回结果得 2 分

第三部分：云计算平台运维管理（共 35 分）

任务一、IaaS 云平台运维（20 分）

IaaS 平台运维准备工作：

•上传 PC 上的镜像文件 Xiandian-IaaS-All.qcow2 到控制节点，创建名为 iaas-all 的 glance 镜像；

• 解压 PC 上的文件 iaas-error-n001.zip（解压密码为 n001qcow2），上传解压出的镜像文件 iaas-error-n001.qcow2 到控制节点，创建名为 iaas-error-n001 的 glance 镜像；

• 解压 PC 上的文件 iaas-error-z001.zip（解压密码为 z001qcow2），上传解压出的镜像文件 iaas-error-z001.qcow2 到控制节点，创建名为 iaas-error-z001 的 glance 镜像；

• 解压 PC 上的文件 iaas-error-520.zip（解压密码为 520qcow2），上传解压出的镜像文

件 iaas-error-520.qcow2 到控制节点，创建名为 iaas-error-520 的 glance 镜像。

按以下配置在云平台中创建云主机：

（1）名称：iaas_all；

（2）镜像文件：iaas-all；

（3）云主机类型：m1.large；

（4）网络 1：int-net1，绑定浮动 IP；

（5）网络 2：int-net2。

1.MongoDB 管理（2 分）

登录“iaas_all”云主机，使用 mongoimport 命令，将附件中的 test.dat 文件，导入至 MongoDB 下的 basketball 数据库中的 player 集合中。导入后登录 MongoDB 数据库。

（1）使用命令查询 player 集合中 MPG 大于 35 分钟，且 PPG 在 20-25 分之间（包括 20 和 25）的所有数据，并按照 PTS 的降序排序。

（2）使用命令查询 player 集合中 PTS 超过 2000 分，且 3P%大于等于 33 的前两条数据。

依次将操作命令及返回结果以文本形式提交到答题框。

PPG--场均得分；PTS--总得分；FG%--投篮命中率；3P%--三分命中率；MPG--平均上场时间

```
[root@xiandian ~]# mongoimport -d basketball -c player test.dat
connected to: 127.0.0.1
2018-03-16T23:12:06.364+0000 check 9 25
2018-03-16T23:12:06.365+0000 imported 25 objects
> use basketball
switched to db basketball
> db.player.find({"MPG":{$gt:"35"},"PPG":{$gte:"20"},"PPG":{$lte:"25"}}).sort({"PTS":-1})
{ "_id" : ObjectId("5aac4f461c7c77fbfa3fc096"), "name" : "tangsi", "PPG" : "21.3", "PTS" : "1743", "FG%" : "54.5", "3P%" : "42.1", "MPG" : "35.6" }
{ "_id" : ObjectId("5aac4f461c7c77fbfa3fc093"), "name" : "qiaozhi", "PPG" : "21.9", "PTS" : "1734", "FG%" : "43", "3P%" : "40.1", "MPG" : "36.6" }
{ "_id" : ObjectId("5aac4f461c7c77fbfa3fc095"), "name" : "maikelemu", "PPG" : "21.4", "PTS" : "1732", "FG%" : "44.3", "3P%" : "39.7", "MPG" : "36.1" }
```

{ "_id" : ObjectId("5aac4f461c7c77fbfa3fc098"), "name" : "mideerdun", "PPG" : "20.1", "PTS" : "1652", "FG%" : "46.6", "3P%" : "35.9", "MPG" : "36.4" } { "_id" : ObjectId("5aac4f461c7c77fbfa3fc091"), "name" : "batele", "PPG" : "22.2", "PTS" : "1307", "FG%" : "47.4", "3P%" : "35", "MPG" : "36.7" } > db.player.find({"PTS":{$gt:"2000"},"3P%":{$gte:"33"}}).limit(2) { "_id" : ObjectId("5aac4f451c7c77fbfa3fc083"), "name" : "hadeng", "PPG" : "30.4", "PTS" : "2191", "FG%" : "44.9", "3P%" : "36.7", "MPG" : "35.4" } { "_id" : ObjectId("5aac4f461c7c77fbfa3fc084"), "name" : "daiweisi", "PPG" : "28.1", "PTS" : "2110", "FG%" : "53.4", "3P%" : "34", "MPG" : "36.4" }
正确的导入 test.dat 脚本得 0.2 分 正确的查询条件得 1 分 正确的返回结果（球员名字）得 0.8 分

2.cinder 管理（3 分）

（1）登录 http://192.168.100.10/dashboard，创建云主机名为“vm_extend”，镜像使用“centos6.5”，flavor 使用“m1.medium”；

（2）登录“vm_extend”云主机，从该主机的硬盘“/dev/vda”中分出一个 10G 的分区，使用这个分区将云主机“vm_extend”根目录所在逻辑分区扩容 5G；

（3）在云主机上用 df -h 命令查看挂载信息。

依次将操作命令及返回结果以文本形式提交到答题框。

[root@host-10-10-10-115 ~]# lsblk NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT vda 252:0 0 40G 0 disk ├─vda1 252:1 0 500M 0 part /boot ├─vda2 252:2 0 7.5G 0 part │ ├─VolGroup-lv_root (dm-0) 253:0 0 6.7G 0 lvm / │ └─VolGroup-lv_swap (dm-1) 253:1 0 816M 0 lvm [SWAP]

```
└─vda3                          252:3    0    10G  0 part
[root@host-10-10-10-115 ~]# vgs
  VG          #PV #LV #SN Attr   VSize  VFree
  VolGroup      1   2   0 wz--n- 7.51g      0
[root@host-10-10-10-115 ~]# lvs
  LV       VG          Attr       LSize   Pool Origin Data%  Move Log Cpy%Sync
Convert
  lv_root VolGroup -wi-ao----   6.71g
  lv_swap VolGroup -wi-ao---- 816.00m
[root@host-10-10-10-115 ~]# pvcreate /dev/vda3
  Physical volume "/dev/vda3" successfully created
[root@host-10-10-10-115 ~]# vgextend VolGroup /dev/vda3
  Volume group "VolGroup" successfully extended
[root@host-10-10-10-115 ~]# vgs
  VG          #PV #LV #SN Attr   VSize  VFree
  VolGroup      2   2   0 wz--n- 17.50g 10.00g
[root@host-10-10-10-115 ~]# lvextend -L +5G /dev/mapper/VolGroup-lv_root
  Extending logical volume lv_root to 11.71 GiB
  Logical volume lv_root successfully resized
[root@host-10-10-10-115 ~]# df -h
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root    6.7G  825M  5.5G  13% /
tmpfs                           1.9G     0  1.9G   0% /dev/shm
/dev/vda1                       485M   32M  428M   7% /boot
[root@host-10-10-10-115 ~]# resize2fs /dev/mapper/VolGroup-lv_root
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/mapper/VolGroup-lv_root is mounted on /; on-line resizing required
old desc_blocks = 1, new_desc_blocks = 1
Performing an on-line resize of /dev/mapper/VolGroup-lv_root to 3069952 (4k) blocks.
The filesystem on /dev/mapper/VolGroup-lv_root is now 3069952 blocks long.
```

```
[root@host-10-10-10-115 ~]# df -h
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root     12G  826M   11G   8% /
tmpfs                           1.9G     0  1.9G   0% /dev/shm
/dev/vda1                       485M   32M  428M   7% /boot
```

创建物理卷成功得 0.6 分

扩容卷组成功得 0.6 分

扩容根目录所在的逻辑卷得 0.6 分

扩容文件系统得 0.6 分

使用 df -h 得到正确的返回结果得 0.6 分

3.防火墙管理（1 分）

登录 http://192.168.100.10/dashboard，添加名为 fw_ruler 的防火墙规则，拒绝所有源 IP、源端口使用 TCP 协议访问“iaas_all”云主机。使用 neutron 命令查询规则列表信息。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@controller ~]# neutron firewall-rule-list
+--------------------------------------+------------+--------------------------------------+------------------
+---------+
| id                                   | name       | firewall_policy_id
| summary                    | enabled |
+--------------------------------------+------------+--------------------------------------+------------------
+---------+
| 50bb0e10-18b3-4ae5-8918-353992b02833 | fw_ruler | 9b9d1d60-6c9b-40e8-b49e-3df793ba2183
| ICMP,                      | True    |
|                                      |                                      |
|   source: none(none), |            |
|                                      |                                      |
|   dest: none(none),   |            |
```

```
|                                                                    |                        |
|    reject                      |              |
+----------------------------------+------------+---------------------------------+--------------------
+---------+
```

正确的查询命令得 0.5 分

正确的返回结果得 0.5 分

4.nova 管理（1 分）

登录“iaas_all”云主机,通过 nova 的相关命令创建名为 exam、ID 为 1234、内存为 1024M、硬盘为 20G、虚拟内核数量为 2 的云主机类型，查看 exam 的详细信息。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@xiandian ~]# nova flavor-create exam 1234 1024 20 2
+------+------+-----------+------+-----------+------+-------+-------------+-----------+
| ID   | Name | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+------+------+-----------+------+-----------+------+-------+-------------+-----------+
| 1234 | exam | 1024      | 20   | 0         |      | 2     | 1.0         | True      |
+------+------+-----------+------+-----------+------+-------+-------------+-----------+
[root@xiandian ~]# nova flavor-show 1234
+----------------------------+-------+
| Property                   | Value |
+----------------------------+-------+
| OS-FLV-DISABLED:disabled   | False |
| OS-FLV-EXT-DATA:ephemeral  | 0     |
| disk                       | 20    |
| extra_specs                | {}    |
| id                         | 1234  |
| name                       | exam  |
| os-flavor-access:is_public | True  |
```

ram	1024
rxtx_factor	1.0
swap	
vcpus	2

+-----------------------+------+

正确的创建 flavor 命令得 0.5 分

正确的 flavor 详情信息得 0.5 分

5.KVM 管理（1 分）

登录 compute 节点，使用命令将 KVM 进程绑定到特定的 0-5 号 CPU 上。依次将操作命令及返回结果以文本形式提交到答题框。

[root@controller ~]# ps -e |grep kvm

765 ? 00:00:00 kvm-irqfd-clean

7490 ? 00:00:12 qemu-kvm

7497 ? 00:00:00 kvm-pit/7490

[root@controller ~]# taskset -cp 0-5 7490

pid 7490 's current affinity list: 0-15

pid 7490 's new affinity list: 0-5

查询到 kvm 所在的进程得 0.5 分

绑定到 0-5 号 CPU 上的返回信息得 0.5 分

6.Nova 排错（4 分）

使用镜像 iaas-error-n001 重建云主机“iaas_all”（账号：root 密码：000000）。重建后的云主机内有错误的 openstack 平台，其中有已经创建好的云主机 vm-test，错误现象为无法

获取云主机的列表和详细信息，找出错误原因，并进行排错。将 nova show 命令查询到的云主机 vm-test 详细信息以文本形式提交到答题框。

```
[root@xiandian ~]# vi /etc/my.cnf.d/mariadb-server.cnf
[root@xiandian ~]# chmod 755 /var/lib/mysql/
[root@xiandian ~]# openstack endpoint create --region RegionOne compute public http://xiandian:8774/v2.1/%\(tenant_id\)s
+--------------+----------------------------------------+
| Field        | Value                                  |
+--------------+----------------------------------------+
| enabled      | True                                   |
| id           | 483e7683e5954e0c8fa660afda5c7b16       |
| interface    | public                                 |
| region       | RegionOne                              |
| region_id    | RegionOne                              |
| service_id   | 1af6174c6dd043bf8f16adcb97bbad28       |
| service_name | nova                                   |
| service_type | compute                                |
| url          | http://xiandian:8774/v2.1/%(tenant_id)s |
+--------------+----------------------------------------+
[root@xiandian ~]# source /etc/xiandian/openrc.sh
[root@xiandian ~]# openstack endpoint create --region RegionOne compute internal http://$HOST_NAME:8774/v2.1/%\(tenant_id\)s
+--------------+----------------------------------------+
| Field        | Value                                  |
+--------------+----------------------------------------+
| enabled      | True                                   |
| id           | 0286d3d4cc7946978308100e1c31cf69       |
| interface    | internal                               |
| region       | RegionOne                              |
| region_id    | RegionOne                              |
| service_id   | 1af6174c6dd043bf8f16adcb97bbad28       |
```

```
| service_name | nova                                                |
| service_type | compute                                             |
| url          | http://xiandian:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
[root@xiandian ~]# openstack endpoint create --region RegionOne compute admin http://$HOST_NAME:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------+
| Field        | Value                             |
+--------------+-----------------------------------+
| enabled      | True                              |
| id           | 7e7eae61f1aa467e9516ffc81acff4b1  |
| interface    | admin                             |
| region       | RegionOne                         |
| region_id    | RegionOne                         |
| service_id   | 1af6174c6dd043bf8f16adcb97bbad28  |
| service_name | nova                              |
| service_type | compute                           |
| url          | http://xiandian:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------+
MariaDB [(none)]> create database nova_api;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY '000000' ;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY '000000' ;
Query OK, 0 rows affected (0.01 sec)
[root@xiandian ~]# su -s /bin/sh -c "nova-manage api_db sync" nova
Option "notification_driver" from group "DEFAULT" is deprecated. Use option "driver" from
```

```
group "oslo_messaging_notifications".

[root@xiandian ~]# openstack user set --password 000000 nova
[root@xiandian ~]# openstack-service restart nova-compute
[root@xiandian ~]# openstack-service restart nova-api
[root@xiandian ~]# nova show 32ca829b-133c-4026-b0ea-3ac9604ece50
+--------------------------------------+----------------------------------------------------------+
| Property                             | Value                                                    |
+--------------------------------------+----------------------------------------------------------+
| OS-DCF:diskConfig                    | AUTO                                                     |
| OS-EXT-AZ:availability_zone          | nova                                                     |
| OS-EXT-SRV-ATTR:host                 | xiandian                                                 |
| OS-EXT-SRV-ATTR:hostname             | vm-test                                                  |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | xiandian                                                 |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000001                                        |
| OS-EXT-SRV-ATTR:kernel_id            |                                                          |
| OS-EXT-SRV-ATTR:launch_index         | 0                                                        |
| OS-EXT-SRV-ATTR:ramdisk_id           |                                                          |
| OS-EXT-SRV-ATTR:reservation_id       | r-c7fl7h34                                               |
| OS-EXT-SRV-ATTR:root_device_name     | /dev/vda                                                 |
```

```
|
|     OS-EXT-SRV-ATTR:user_data                                      |     -
|
|     OS-EXT-STS:power_state                                         |     1
|
|     OS-EXT-STS:task_state                                          |     -
|
|     OS-EXT-STS:vm_state                                            |     active
|
|   OS-SRV-USG:launched_at                                 |   2018-05-04T00:53:13.000000
|
|     OS-SRV-USG:terminated_at                                       |     -
|
|    accessIPv4                                                                |
|
|    accessIPv6                                                                |
|
|    config_drive                                                              |
|
|   created                                                  |   2018-05-04T00:53:01Z
|
|    description                                                     |    vm-test
|
|   flavor                                                   |   m1.tiny   (1)
|
|    hostId                                                                    |
991e57844e869d97b120c16afc7f4820ab6361ed1904375f832453e4 |
|    host_status                                                     |    UP
|
|  id                                                |  32ca829b-133c-4026-b0ea-3ac9604ece50
|
```

```
| image                                | Image not found
(33c11940-11d3-4a94-b976-392837d4dfed)  |
| key_name                             | -
|
| locked                               | False
|
| metadata                             | {}
|
| name                                 | vm-test
|
| os-extended-volumes:volumes_attached | []
|
| progress                             | 0
|
| security_groups                      | default
|
| sharednet1 network                   | 10.0.0.3
|
| status                               | ACTIVE
|
| tenant_id                            | 0ab2dbde4f754b699e22461426cd0774
|
| updated                              | 2018-05-04T00:53:13Z
|
| user_id                              | 53a1cf0ad2924532aa4b7b0750dec282
|
+--------------------------------------+----------------------------------------------------+
```

得到云主机的详细信息得 4 分，结果导向，过程不得分

7.综合排错（4 分）

使用镜像 iaas-error-z001 重建“iaas_all”云主机（账号：root 密码：000000）。重建后的云主机内有错误的 openstack 平台，现象为创建云硬盘错误，创建云主机错误，无法将云硬盘挂载到云主机上。进行排错，排错之后创建大小为 1G，类型为 luks 的云硬盘 block1；使用镜像 cirros，创建名字为 vm-test 的云主机，创建完成后，将 block1 挂载到 vm-test 上。以文本形式依次提交云硬盘、云主机的排错命令和 cinder list 查询结果到答题框。

```
fixed_key=cinder-volume-key
fixed_key=0000000000000000
[root@xiandian ~]# umount /media/
[root@xiandian ~]# pvcreate /dev/vda7
WARNING: ext4 signature detected on /dev/vda7 at offset 1080. Wipe it? [y/n]: y
  Wiping ext4 signature on /dev/vda7.
  Physical volume "/dev/vda7" successfully created
[root@xiandian ~]# vgcreate cinder-volumes /dev/vda7
  Volume group "cinder-volumes" successfully created
[root@xiandian ~]# vgs
  VG                  #PV #LV #SN Attr      VSize  VFree
  cinder-volumes      1   0   0 wz--n- 4.75g 4.75g
[root@xiandian ~]# openstack user show nova
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 3ac89594c8e944a9b5bb567fca4e75aa |
| enabled   | False                            |
| id        | a0024f4b405745b999aaa1276844dec6 |
| name      | nova                             |
+-----------+----------------------------------+
fixed_key=cinder-volume-key
virt_type = kvm
```

```
fixed_key=0000000000000000
virt_type = qemu
[root@xiandian ~]# openstack user set --enable nova
[root@xiandian ~]# openstack user set --password 000000 nova
[root@xiandian ~]# cinder list
+--------------------------------------+--------+--------+------+-------------+----------+--------------------
--------------+
|                  ID                  | Status |  Name  | Size | Volume Type | Bootable
|             Attached to              |
+--------------------------------------+--------+--------+------+-------------+----------+--------------------
--------------+
| 01d149ad-2e34-4bde-b4d9-f42e1f64956a | in-use | block1 |  1   |      luks   |  false   |
231f3e6d-4c81-4eb1-9b16-dfb9a7c7e7cf |
+--------------------------------------+--------+--------+------+-------------+----------+--------------------
--------------+
```

成功创建 cinder-volumes 卷组得 2 分

设置 nova 用户 enable 得 1 分

成功将云硬盘挂载到云主机上得 1 分

8.Glance 排错（4 分）

使用镜像 iaas-error-520 重建“iaas_all”云主机（账号：root 密码：000000）并排除重建过程中的错误。重建后的云主机内有错误的 openstack 平台，错误现象为无法获取镜像信息，试排错。排错完之后通过 glance 命令查询该镜像详细信息。依次将错误的信息、排错的操作命令及查询到的信息以文本形式提交到答题框。

```
[root@xiandian ~]# glance image-list
An unexpected error prevented the server from fulfilling your request. (HTTP 500) (Request-ID:
req-7339bd41-bf22-49c4-89a4-3fbfd20568c6)
[root@xiandian ~]# vi /etc/my.cnf
```

将 max_connections=1 注释掉

[root@xiandian ~]# systemctl restart mariadb

[root@xiandian ~]# openstack image show $(glance image-list | awk -F '|' '/\ xiandian\s/{print$2}')

```
+-----------------+------------------------------------------------------+
| Field           | Value                                                |
+-----------------+------------------------------------------------------+
| checksum        | ee1eca47dc88f4879d8a229cc70a07c6                     |
| container_format| bare                                                 |
| created_at      | 2018-05-18T10:05:58Z                                 |
| disk_format     | qcow2                                                |
| file            | /v2/images/ea6ed9be-5f5b-40d2-b3a9-e3500b59a535/file |
| id              | ea6ed9be-5f5b-40d2-b3a9-e3500b59a535                 |
| min_disk        | 0                                                    |
| min_ram         | 0                                                    |
| name            | xiandian                                             |
| owner           | 0ab2dbde4f754b699e22461426cd0774                     |
| protected       | False                                                |
| schema          | /v2/schemas/image                                    |
| size            | 13287936                                             |
| status          | active                                               |
| tags            | paicuo                                               |
| updated_at      | 2018-05-18T23:38:28Z                                 |
| virtual_size    | None                                                 |
| visibility      | private                                              |
+-----------------+------------------------------------------------------+
```

贴出 max_connections=1 得 2 分

得到正确的镜像 checksum 值得 2 分

任务二、容器运维（15 分）

1.容器底层服务（2 分）

在容器 server 节点上创建 memory 控制的 cgroup，名称为 xiandian，将当前 shell 的进程移动到 cgroup 中，通过 cat 相关命令查询 cgroup 中的进程。依次将操作命令及返回结果以文本形式提交到答题框。

[root@server ~]# mkdir /sys/fs/cgroup/memory/xiandian [root@server ~]# echo $$ 956 [root@server ~]# sudo sh -c "echo $$ >> /sys/fs/cgroup/memory/xiandian/tasks" [root@server ~]# cat /proc/956/cgroup 10:hugetlb:/ 9:perf_event:/ 8:blkio:/system.slice/sshd.service 7:net_cls:/ 6:freezer:/ 5:devices:/system.slice/sshd.service 4:memory:/xiandian 3:cpuacct,cpu:/system.slice/sshd.service 2:cpuset:/ 1:name=systemd:/system.slice/sshd.service
创建正确的 cgroup 得 0.5 分 将当前 shell 的进程移到 cgroup 中得 1 分 查询得到正确的结果得 0.5 分

2.容器存储配置（2 分）

在容器 server 节点创建/opt/xiandian-ro 目录，使用镜像 nginx:latest 创建名为 xiandian 的容器，将/opt/xiandian-ro 目录挂载到容器内部/opt 下，并设置为只读模式。通过 inspect 命令

查看 HostConfig 内的 Binds 信息。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@server ~]# mkdir /opt/xiandian-ro
[root@server ~]# docker run -d -it  -P --name xiandian -v /opt/xiandian-ro:/opt:ro nginx:latest /bin/bash
a85027b37c31512bff91cdb915f179277f20abb906b85153c1d5a94e941c8a4b
[root@server ~]# docker inspect -f '{{.HostConfig.Binds}}' xiandian
[/opt/xiandian-ro:/opt:ro]
```

创建目录得 0.5 分

正确容器目录挂载参数得 1 分

正确的查询返回结果得 0.5 分

3.容器网络（2 分）

（1）在容器 server 节点，使用 docker 命令创建名为 xd_net 的网络，网络网段为 192.168.3.0/24，网关为 192.168.3.1；

（2）使用镜像 centos:latest 和 xd_net 网络，创建名为 centos_net 的容器；

（3）使用 inspect -f 命令查询容器使用的网络名称；

（4）查询容器的运行状态。

依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@server ~]# docker network create --subnet=192.168.3.0/24 --ip-range=192.168.3.0/24 --gateway=192.168.3.1  xd_net
b8199203dbfeeee03956ab851ccaaa9753b83955a82abf91114260fc43715ad2
[root@server ~]# docker run -itdP --net=xd_net --name centos_net centos:latest
6783d9fb4510df5208e89f45afc477643dfba662d118aa4b1dc00eb1fd5aeb67
[root@server ~]# docker inspect -f '{{.NetworkSettings.Networks}}' centos
map[xd_net:0xc4200bc0c0]
[root@server ~]# docker ps -a
CONTAINER ID        IMAGE                                  COMMAND
CREATED                         STATUS                          PORTS
```

NAMES 6783d9fb4510 centos:latest "/bin/bash" 27 seconds ago Up 23 seconds centos
创建正确的 docker 网络得 1 分 创建容器使用创建的网络得 0.5 分 查询得到正确的返回结果得 0.5 分

4.容器构建（3 分）

在容器 server 节点，使用 supermin5 命令（若命令不存在，则自己安装）构建名为 centos-7 的 centos7 系统 docker 镜像，镜像预装 yum、net-tools、initscripts 和 vi 命令。构建完成后提交镜像至容器仓库，并查看此镜像。依次将操作命令及返回结果以文本形式提交到答题框。

[root@server ~]# supermin5 -v --prepare bash yum net-tools initscripts vi coreutils -o supermin.d [root@server ~]# supermin5 -v --build --format chroot supermin.d -o appliance.d [root@server ~]# echo 7 > appliance.d/etc/yum/vars/releasever [root@server ~]# tar --numeric-owner -cpf centos-7.tar -C appliance.d . [root@server ~]# cat centos-7.tar | docker import - 10.0.6.126:5000/centos-7 sha256:75b27ee7851e297faa2a085c7dac3c0f25bc4ac63949b04616fa3d247e52d007 [root@server ~]# docker images centos-7 REPOSITORY TAG IMAGE ID CREATED SIZE 10.0.6.126:5000/centos-7 latest 75b27ee7851e 50 seconds ago 261.5 MB
使用 supermin5 命令预装正确的软件得 1 分 构建镜像正确得 1 分 查看构建的镜像 1 分

5.Dockerfile 编写（3 分）

以上题构建的 centos-7 镜像为基础，构建数据库镜像 centos-mariadb:v1.0，其要求为：

（1）删除镜像的本地 yum 源，使用容器 server 节点的 yum 源文件；

（2）安装 mariadb 服务，使用 mysql 用户初始化数据库；

（3）设置 MYSQL_USER=xiandian、MYSQL_PASS=xiandian 环境变量；

（4）数据库支持中文；

（5）暴露 3306 端口；

（6）启动容器时能自动运行 mysld_safe 命令。

使用 cat 命令查看 Dockerfile 文件并构建镜像。依次将操作命令及返回结果以文本形式提交到答题框。

```
# cat Dockerfile
FROM 10.0.6.126:5000/centos-7
MAINTAINER  Xiandian
RUN  rm -fv /etc/yum.repos.d/*
ADD  local.repo /etc/yum.repos.d/
RUN  yum install  -y mariadb-server
RUN  mysql_install_db  --user=mysql
ENV  LC_ALL  en_US.UTF-8
ENV  MYSQL_USER  xiandian
ENV  MYSQL_PASS  xiandian
EXPOSE 3306
CMD  mysqld_safe
# docker build -t 10.0.6.126:5000/centos-mariadb:v1.0 .
[root@server mysql]# docker images
REPOSITORY                                                                  TAG
IMAGE ID                CREATED                 SIZE
10.0.6.126:5000/centos-mariadb                                              v1.0
36dfc7b3a525            15 seconds ago          685 MB
```

正确的 dockerfile 内容得 2 分

构建正确的镜像得 1 分

6.容器综合（3 分）

（1）在容器 server 节点，创建/root/www1 目录，在 www1 目录下编写 index.jsp 文件；

（2）使用 tomcat 镜像创建容器，容器的默认项目地址连接到/root/www1 目录，要求访问 tomcat 的时候输出语句为“this is Tomcat1”；

（3）进入容器启动 tomcat 服务，使用 curl 命令查询 tomcat 的首页。

依次将操作命令及返回结果以文本形式提交到答题框。

[root@server ~]# docker run -it -d -P -h tomcat1 -v /root/www1:/usr/local/tomcat/webapps/ROOT 10.0.6.126:5000/tomcat:latest /bin/bash f3f8c24dfb0adf112e8a6fdcb49fbd584fd5f9fbbf454146d369e97c3366985c [root@server ~]# docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES f3f8c24dfb0a 10.0.6.126:5000/tomcat:latest "/bin/bash" 14 seconds ago Up 11 seconds 0.0.0.0:32768->8080/tcp adoring_bose [root@server ~]# docker exec -it f3f8c24dfb0a /bin/bash root@tomcat1:/usr/local/tomcat# cd bin/ root@tomcat1:/usr/local/tomcat/bin# startup.sh Tomcat started. [root@server ~]# curl http://10.0.6.126:32768 <title>Tomcat1</title> <body> this is Tomcat1 </body>
挂载本地目录到容器 tomcat 工程目录参数得 1 分 进入容器启动 tomcat 服务得 1 分

Curl 到正确的返回结果得 1 分

第四部分：大数据平台（共 15 分）

任务一、大数据平台搭建（5 分）

大数据平台的搭建采用分布式部署，部署在云平台的两台虚拟机上，在云主机 1 上部署大数据平台 master 节点，在云主机 2 上部署大数据平台 slaver 节点：

云主机 1：

（1）名称：master；

（2）镜像文件：hadoop_master_centos7_x86_xiandian_images-v05.qcow2；

（3）类型：4CPU、8G 内存、100G 硬盘；

（4）网络 1：int-net1，绑定浮动 IP。

云主机 2：

（1）名称：slaver；

（2）镜像文件：hadoop_slaver1_centos7_x86_xiandian_images-v05.qcow2；

（3）类型：4CPU、8G 内存、100G 硬盘；

（4）网络 1：int-net1，绑定浮动 IP。

1.主机名配置（1 分）

使用 cat 命令查看云主机 master 和 slaver 的 hosts 文件。依次将操作命令及返回结果以文本形式提交到答题框。

```
[root@master ~]# cat /etc/hosts
127.0.0.1     localhost localhost.localdomain localhost4 localhost4.localdomain4
::1              localhost localhost.localdomain  localhost6 localhost6.localdomain6
10.0.0.10           master.hadoop master
10.0.0.11           slaver1.hadoop
[root@slaver1 ~]# cat /etc/hosts
127.0.0.1     localhost localhost.localdomain localhost4 localhost4.localdomain4
```

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6 10.0.0.10 master.hadoop 10.0.0.11 slaver1.hadoop slaver1
查询 hosts 文件得 1 分

2.内存大页配置（1 分）

检查云主机 master 和 slaver 的内存配置文件，查看 Transparent Huge Pages 状态。依次将操作命令及返回结果以文本形式提交到答题框。

echo never > /sys/kernel/mm/transparent_hugepage/enabled # echo never > /sys/kernel/mm/transparent_hugepage/defrag # cat /sys/kernel/mm/transparent_hugepage/enabled always madvise [never]
查询到状态得 1 分

3.大数据平台环境配置（1 分）

在 master 节点启动 HTTP 服务，使用 systemctl 命令查看 HTTP 服务的状态。依次将操作命令及返回结果以文本形式提交到答题框。

[root@master ~]# systemctl status httpd httpd.service - The Apache HTTP Server Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled) Active: active (running) since Wed 2017-05-03 01:14:13 UTC; 3 days ago Docs: man:httpd(8) man:apachectl(8) Main PID: 1277 (httpd) Status: "Total requests: 114; Current requests/sec: 0; Current traffic: 0 B/sec" CGroup: /system.slice/httpd.service ├─1277 /usr/sbin/httpd -DFOREGROUND ├─1278 /usr/sbin/httpd -DFOREGROUND

├─1279 /usr/sbin/httpd -DFOREGROUND ├─1280 /usr/sbin/httpd -DFOREGROUND ├─1281 /usr/sbin/httpd -DFOREGROUND ├─1282 /usr/sbin/httpd -DFOREGROUND ├─2350 /usr/sbin/httpd -DFOREGROUND ├─2351 /usr/sbin/httpd -DFOREGROUND └─2352 /usr/sbin/httpd -DFOREGROUND May 03 01:14:13 master httpd[1277]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using ... message May 03 01:14:13 master systemd[1]: Started The Apache HTTP Server. Hint: Some lines were ellipsized, use -l to show in full.
使用 systemctl 命令查询状态得 0.5 分 得到正确的返回结果得 0.5 分

4.启动大数据平台（2 分）

（1）启动大数据平台并登录 http://{master-ip}:8080，用户名：admin，密码：admin；

（2）平台中已经安装了以下服务组件：HDFS、MapReduce2、YARN、Tez、Hive、HBase、Pig、Zookeeper、Mahout。其中 master 节点 Mariadb 数据库用户密码配置如表 3 所示。

表 3 数据库用户密码配置表

用户名	密码
root	bigdata
ambari	bigdata
hive	bigdata

启动平台中安装的 HDFS、MapReduce2、YARN、Zookeeper 等服务；

（3）先后在 master 节点和 slaver 节点的 Linux Shell 中查看 Hadoop 集群的服务进程信息。

依次将操作命令及返回结果以文本形式提交到答题框。

[root@master ~]# jps 16145 Jps 18195 Bootstrap 9668 DataNode 23796 QuorumPeerMain 10053 NameNode 11285 NodeManager 3340 HMaster 1374 AmbariServer 3375 ApplicationHistoryServer [root@slaver1 ~]# jps 20529 RunJar 8242 NodeManager 7907 ApplicationHistoryServer 19509 RunJar 5830 SecondaryNameNode 17494 Jps 5464 DataNode 21129 RunJar 9513 QuorumPeerMain 7117 JobHistoryServer 8637 ResourceManager
使用 jps 命令得 0.5 分 查到正确的进程信息得 1.5 分

任务二、大数据平台运维（10 分）

1.Sqoop 管理（2 分）

使用 Sqoop 工具查询 master 节点 MySQL 中的所有数据库。依次将操作命令及返回结果

以文本形式提交到答题框。

[root@master ~]# sqoop list-databases --connect jdbc:mysql://localhost --username root --password bigdata Warning: /usr/hdp/2.4.3.0-227/accumulo does not exist! Accumulo imports will fail. Please set $ACCUMULO_HOME to the root of your Accumulo installation. 17/05/07 07:02:34 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6.2.4.3.0-227 17/05/07 07:02:34 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead. 17/05/07 07:02:34 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset. information_schema ambari mysql performance_schema
使用正确的 sqoop 命令连接数据库得 1 分 查询到数据库信息得 1 分

2.Spark 案例（3 分）

登录 spark-shell：

（1）定义值为（1,2,3,4,5,6,7,8,9）的 Int 类型 List 变量，使用 map 函数，对该变量进行元素乘以 2 的操作；

（2）定义值为（"Hadoop","Java","Spark"）的 String 类型 List 变量，使用 flatmap 函数，将该变量转换成 Char 类型 List 变量，并将小写字母转换成大写。

依次将操作命令及返回结果以文本形式提交到答题框。

scala> import scala.math._ import scala.math._ scala> val nums=List(1,2,3,4,5,6,7,8,9) nums: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9) scala> nums.map(x=>x*2)

```
res18: List[Int] = List(2, 4, 6, 8, 10, 12, 14, 16, 18)


scala> val data = List("Hadoop","Java","Spark")
data: List[String] = List(Hadoop, Java, Spark)
scala> data.flatMap(_.toUpperCase)
res0: List[Char] = List(H, A, D, O, O, P, J, A, V, A, S, P, A, R, K)
```

定义 List(1,2,3,4,5,6,7,8,9)得 0.5 分

使用 map 函数正确转换得 0.5 分

定义 List("Hadoop","Java","Spark")得 0.5 分

使用 flatmap 函数参数得 1 分

得到正确的结果得 0.5 分

3.HBase 管理（2 分）

登录大数据 master 节点，新建 hbasetest.txt 文件，编写内容，要求如下：

（1）新建一张表为'test', 列族为'cf'；

（2）向这张表批量插入如下数据；

'row1', 'cf:a', 'value1'

'row2', 'cf:b', 'value2'

'row3', 'cf:c', 'value3'

'row4', 'cf:d', 'value4'

（3）插入数据完毕后用 scan 命令查询表内容；

（4）用 get 命令只查询 row1 的内容；

（5）最后退出 hbase shell。

使用命令运行 hbasetest.txt。依次将 hbasetest.txt 的内容、执行命令和返回结果以文本形式提交到答题框。

```
[root@exam1 ~]# cat hbasetest.txt
create 'test', 'cf'
list 'test'
```

```
put 'test', 'row1', 'cf:a', 'value1'
put 'test', 'row2', 'cf:b', 'value2'
put 'test', 'row3', 'cf:c', 'value3'
put 'test', 'row4', 'cf:d', 'value4'
scan 'test'
get 'test', 'row1'
exit

[root@exam1 ~]# hbase shell hbasetest.txt
0 row(s) in 1.5010 seconds
TABLE
test
1 row(s) in 0.0120 seconds
0 row(s) in 0.1380 seconds
0 row(s) in 0.0090 seconds
0 row(s) in 0.0050 seconds
0 row(s) in 0.0050 seconds
ROW                              COLUMN+CELL
 row1                            column=cf:a, timestamp=1522314428726, value=value1
 row2                            column=cf:b, timestamp=1522314428746, value=value2
 row3                            column=cf:c, timestamp=1522314428752, value=value3
 row4                            column=cf:d, timestamp=1522314428758, value=value4
4 row(s) in 0.0350 seconds
COLUMN                           CELL
 cf:a                            timestamp=1522314428726, value=value1
1 row(s) in 0.0190 seconds
```

hbasetest.txt 文件中有 exit 得 0.5 分

正确的使用 hbasetest.txt 文件得 0.5 分

正确的返回结果得 1 分

4.Mahout 案例（3 分）

使用 Mahout 挖掘工具对数据集 user-item-score.txt（用户-物品-得分）进行物品推荐；将 txt 文档放在 hdfs 的/data/mahout/project 目录下（可自行创建）；采用基于项目的协同过滤算法和欧几里得距离公式、每位用户推荐的物品数为 3、最大偏好值为 4、最小偏好值为 1；将推荐输出结果保存到/data/mahout/project/output 目录中，通过命令查询 part-r-00000 中的内容。依次将执行推荐算法的命令和查询结果以文本形式提交到答题框中。

```
[hdfs@master ~]$ hadoop fs -mkdir -p /data/mahout/project
[hdfs@master ~]$ hadoop fs -put user-item-score.txt /data/mahout/project
[hdfs@master ~]$ mahout recommenditembased -i /data/mahout/project/user-item-score.txt -o /data/mahout/project/output -n 3 -b false -s SIMILARITY_EUCLIDEAN_DISTANCE --maxPrefsPerUser 4 --minPrefsPerUser 1 --maxPrefsInItemSimilarity 4 --tempDir /data/mahout/project/temp
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
Running on hadoop, using /usr/hdp/2.4.3.0-227/hadoop/bin/hadoop and HADOOP_CONF_DIR=/usr/hdp/2.4.3.0-227/hadoop/conf
MAHOUT-JOB: /usr/hdp/2.4.3.0-227/mahout/mahout-examples-0.9.0.2.4.3.0-227-job.jar
WARNING: Use "yarn jar" to launch YARN applications.
17/05/15 19:33:06 WARN driver.MahoutDriver: No recommenditembased.props found on classpath, will use command-line arguments only
17/05/15 19:33:07 INFO common.AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/data/mahout/project/user.txt], --maxPrefsInItemSimilarity=[4], --maxPrefsPerUser=[4], --maxSimilaritiesPerItem=[100], --minPrefsPerUser=[1], --numRecommendations=[3], --output=[/data/mahout/project/output], --similarityClassname=[SIMILARITY_EUCLIDEAN_DISTANCE], --startPhase=[0], --tempDir=[/data/mahout/project/temp]}
17/05/15 19:33:07 INFO common.AbstractJob: Command line arguments: {--booleanData=[false], --endPhase=[2147483647], --input=[/data/mahout/project/user.txt], --minPrefsPerUser=[1], --output=[/data/mahout/project/temp/preparePreferenceMatrix], --ratingShift=[0.0], --startPhase=[0], --tempDir=[/data/mahout/project/temp]}
17/05/15 19:33:08 INFO impl.TimelineClientImpl: Timeline service address:
```

http://slaver1:8188/ws/v1/timeline/

17/05/15 19:33:08 INFO client.RMProxy: Connecting to ResourceManager at slaver1/172.16.180.123:8050

17/05/15 19:33:10 INFO input.FileInputFormat: Total input paths to process : 1

17/05/15 19:33:10 INFO mapreduce.JobSubmitter: number of splits:1

17/05/15 19:33:10 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1494874269419_0013

17/05/15 19:33:11 INFO impl.YarnClientImpl: Submitted application application_1494874269419_0013

17/05/15 19:33:11 INFO mapreduce.Job: The url to track the job: http://slaver1:8088/proxy/application_1494874269419_0013/

17/05/15 19:33:11 INFO mapreduce.Job: Running job: job_1494874269419_0013

17/05/15 19:33:18 INFO mapreduce.Job: Job job_1494874269419_0013 running in uber mode : false

17/05/15 19:33:18 INFO mapreduce.Job: map 0% reduce 0%

17/05/15 19:33:25 INFO mapreduce.Job: map 100% reduce 0%

17/05/15 19:33:33 INFO mapreduce.Job: map 100% reduce 100%

17/05/15 19:33:36 INFO mapreduce.Job: Job job_1494874269419_0013 completed successfully

[hdfs@master ~]$ hadoop fs -cat /data/mahout/project/output/part-r-00000

1 [105:3.5941463,104:3.4639049]

2 [106:3.5,105:2.714964,107:2.0]

3 [103:3.59246,102:3.458911]

4 [107:4.7381864,105:4.2794304,102:4.170158]

5 [103:3.8962872,102:3.8564017,107:3.7692602]

创建/data/mahout/project 目录得 0.6 分

将 txt 文档放到/data/mahout/project 目录下得 0.6 分

使用正确的 mahout 命令和参数得 1.2 分

得到正确的结果得 0.6 分

第五部分：SaaS 云应用开发（共 20 分）

任务一、云存储 WEB 应用开发（5 分）

云存储网盘 Web 应用的开发，选用 javaEE 技术平台，使用集中部署的云存储服务。

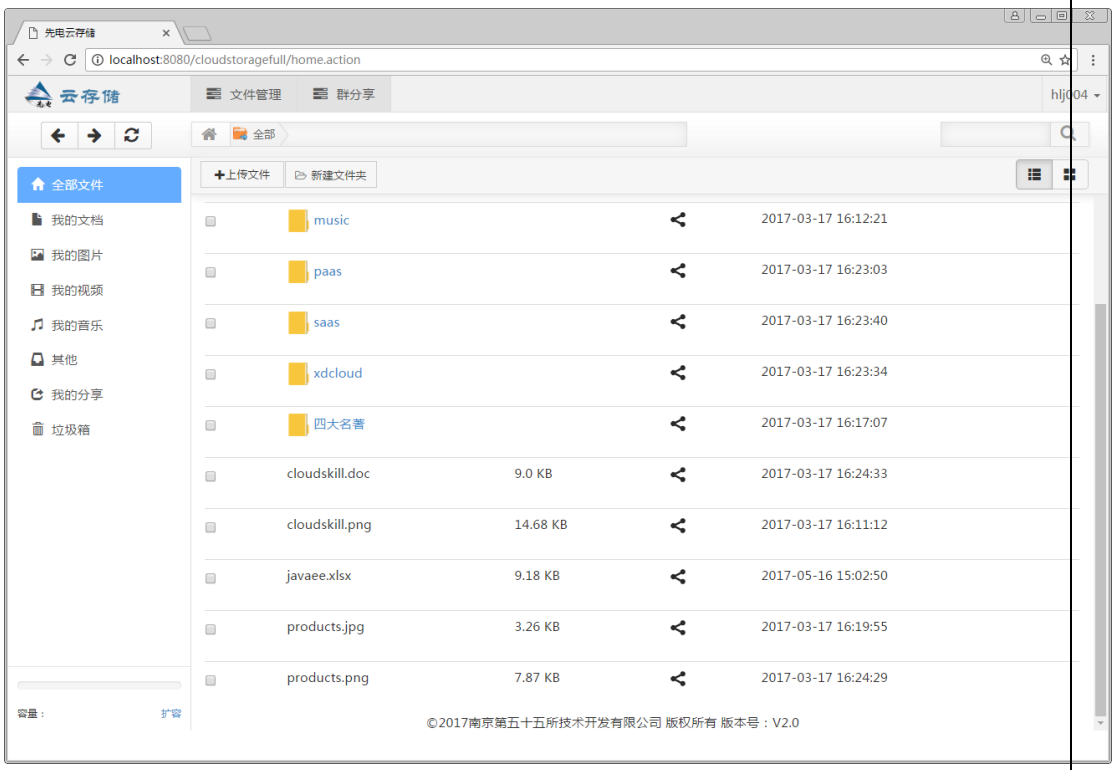
开发环境：Eclipse + JDK + Tomcat + Mysql + swift.sql + 案例 cloudstorage_web.zip。

账户名、密码等配置信息参见考位信息表。

1.搭建开发环境和导入项目（1 分）

根据指定的账户名、密码等信息修改链接云存储平台的配置并运行。按顺序提交浏览器登录后“全部文件”界面截图、修改的配置及修改的配置代码到答题框。

1.效果截图



登录后全部文件截图

2.代码

（1）修改数据库配置文件：jdbc.properties

jdbc.driverClassName=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/swift?useUnicode=true&characterEncoding=UTF-8

jdbc.username=root

jdbc.password=123456

（2）修改云存储配置文件

#1 Config 1 （填写时选手自己的考号和密码，案例为 gw001，000000）

USERNAME=gw001

PASSWORD=000000

AUTHURL=http://swift:35357/v2.0/tokens

TENANTNAME=gw001

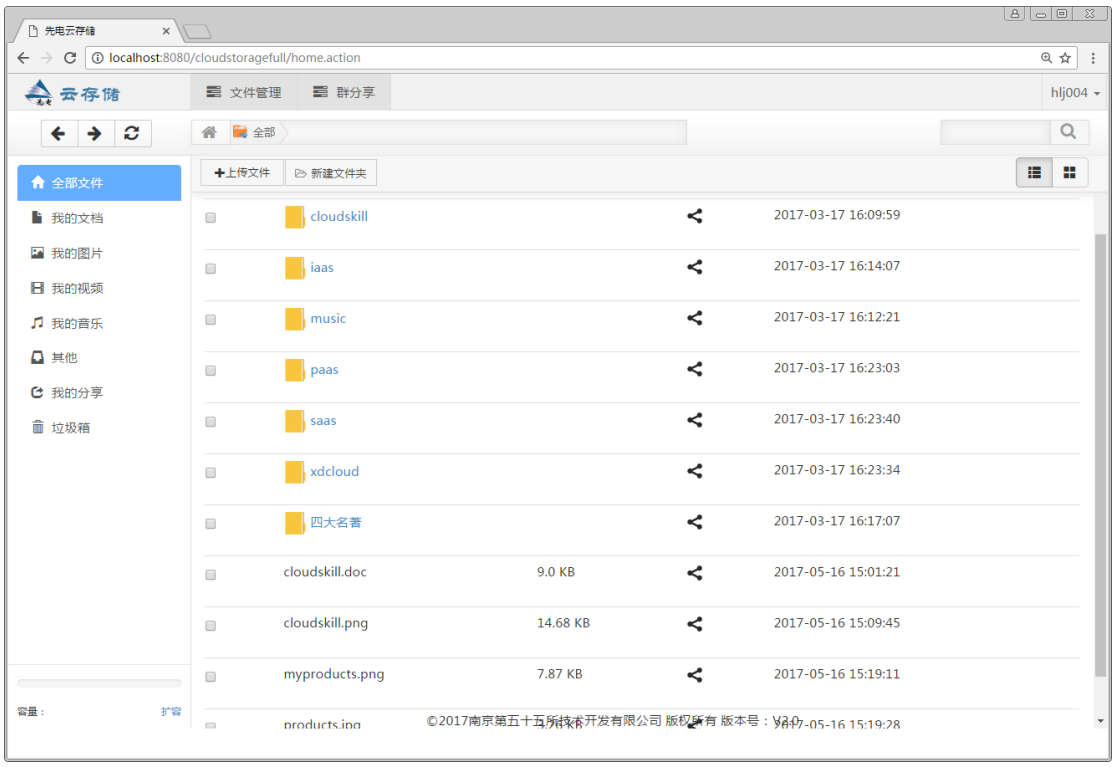
有正确的全部文件截图计 0.5 分。若没有代码计 0 分。

有合理的代码计 0.5 分

2.文件的重命名功能（2 分）

基于 SDK 提供的接口，实现云存储网盘的重命名的功能，将网盘根目录下 products.png 文件重命名为 myproducts.png。实现后按顺序提交重命名后的网页截图和增改的相关代码到答题框。

1.效果截图



2.修改代码

（1）Jsp 页面

```
function sure() {
    var objTable = document.getElementById("tab");
    for ( var y = 0; y < objTable.rows.length; y++) {
        var checkbox = objTable.rows[y].childNodes[1].childNodes[0].childNodes[5];
        if (checkbox.checked == true) {
            var changename =
checkbox.parentNode.parentNode.parentNode.childNodes[3].childNodes[2].childNodes[1]
            var name = $(changename).val();
            var path =
checkbox.parentNode.parentNode.parentNode.childNodes[5].innerHTML;
            path = decodeURIComponent(path);
            var imgpic =
checkbox.parentNode.parentNode.parentNode.childNodes[3]
                .getElementsByTagName("img");
            if (imgpic.length > 0) {
                isDir = true;
            } else {
                isDir = false;
            }
            var data = {
                "path" : path,
                "name" : name,
                "isDir" : isDir
            };
        }
    }
    $.ajax({
        url : "updatefile.action",
        type : "post",
        data : data,
        success : function(s) {
            if (s.success) {
                alert(s.msg)
            } else {
                alert(s.msg);
            }
            location.reload();
        }
    });
}
```

（2）Controller 层

```
/**
 * 更新文件
```

```
     *
     * @param request
     * @param response
     * @param name
     */
    @RequestMapping("/updatefile")
    @ResponseBody
    public Object updatefile(HttpServletRequest request,
            HttpServletResponse response, String path, String name,
            boolean isDir) {
        User user = getSessionUser(request);
        boolean flag = storage
                .updatefile(user.getUsername(), path, name, isDir);
        return new MessageBean(flag, flag ? Constants.SUCCESS_7
                : (isDir ? Constants.ERROR_4 : Constants.ERROR_5));
    }
```

（3）Service 层

```
    /**
     * 重命名文件
     *
     * @param email
     * @param path
     */
    public boolean updatefile(String username, String path, String name,
            boolean isdir) {
        SwiftDFS swiftdfs = new SwiftDFS();
        if (isdir) {
            return swiftdfs.renameDir(username + "/" + path, name);
        } else {
            return swiftdfs.renameFile(username + "/" + path, name);
        }
    }
```

考点：（以上是参考答案不是唯一答案）

SwiftDFS 内的 renameFile 方法

```
swiftdfs.renameFile(username + "/" + path, name);
```

有正确的截图计 1 分，若没有代码按 0 分计。

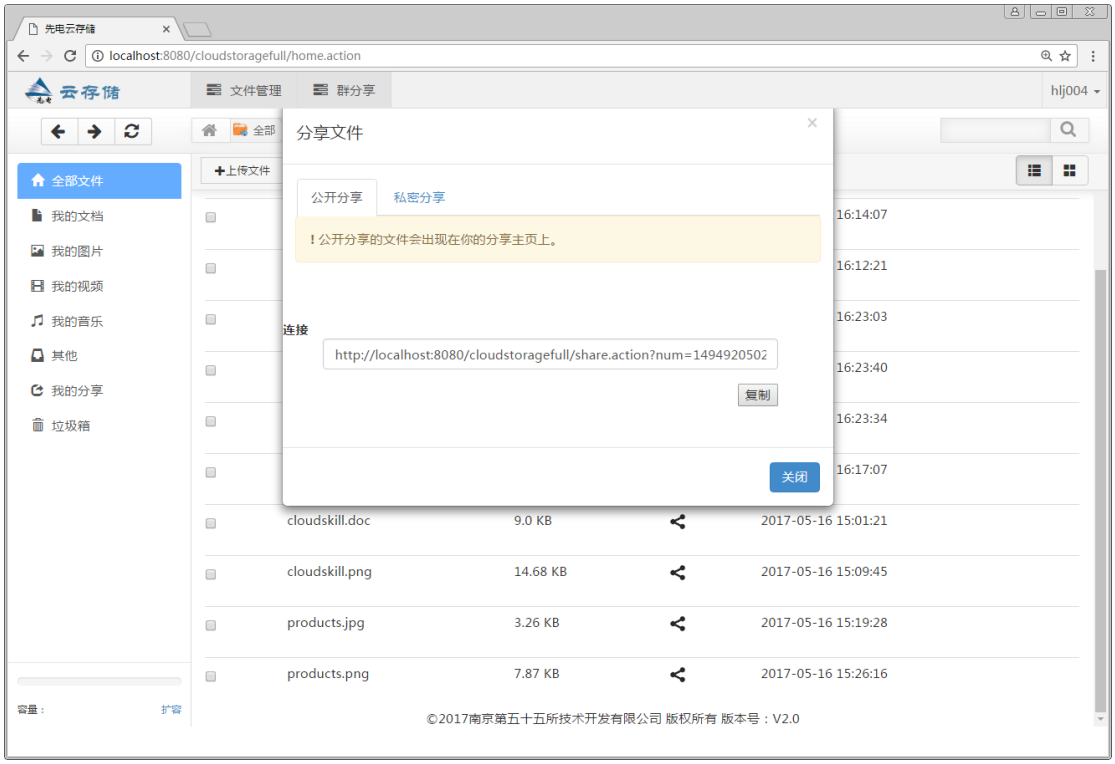
有合理的代码 1 分。

3.实现对单个文件的分享功能（2 分）

基于 SDK 接口,实现完成云网盘文件的分享的功能，分享 cloudskill.png 文件并生成分享

链接。实现后按顺序提交分享的网页截图和增改的 java 代码到答题框。

1.效果截图



2.修改代码

（1）Jsp 页面

```
function createlink(type) {
    var path = "${path}";
    var data = {
        'type' : type,
        'filename' : filename,
        'filepath' : filepath,
        'isdir' : isdir_share,
        'length' : filelength
    };
    $
            .ajax({
                url : 'createLink.action',
                type : 'post',
                data : data,
```

```
success : function(s) {
if (s.success) {
if (type == '1') {
var container = '<div class="panel-heading">'
+ '</div>'
+ '<div class="panel-body" style=" width: 524px;"> '
+ '<div class="form-horizontal" role="form">'
+ '<div class="form-group">'
+ ' <label for="1" style=" margin-left: -44px;" class="col-lg-2 control-label">连接</label>'
+ '<div class="col-lg-10">'
+ ' <input class="form-control" id="1" value="'+s.other.http+'" type="text">'
+ ' </div>'
+ ' </div>'
+ '<div class="form-group" style="display:none">'
+ ' <label for="inputPassword1" style="margin-left: -44px;" class="col-lg-2 control-label">密码</label>'
+ ' <div class="col-lg-4">'
+ ' <input class="form-control" id="2" value="'+s.other.pwd+'" type="text">'
+ ' </div>'
+ ' </div>'
+ ' <div class="form-group">'
+ ' <div class="col-lg-offset-2 col-lg-10" style="text-align:right">'
+ ' <button type="submit" class="btn-u btn-u-green" onclick="copyToClipBoard(3,2,1)" id="3">复制</button>'
```

```
                                        + '          </div>'
                                        + '      </div>'
                                        + '    </div>' + '    </div>';
                                $("#publicLink").empty().append(container);
                                copyToClipBoard(3, 2, 1);
                            } else {
                                var container = '<div class="panel-heading">'
                                        + '</div>'
                                        + '<div class="panel-body" style=" width:
524px;">    '
                                        +       '<div       class="form-horizontal"
role="form">'
                                        + '<div class="form-group">'
                                        +  '  <label  for="4"  style="margin-left:
-44px;" class="col-lg-2 control-label">连接</label>'
                                        + '<div class="col-lg-10">'
                                        +  '    <input class="form-control" id="4"
value="'+s.other.http+'" type="text">'
                                        + '</div>'
                                        + '</div>'
                                        + '<div class="form-group">'
                                        +  '        <label  for="inputPassword1"
style="margin-left: -44px;" class="col-lg-2 control-label">密码</label>'
                                        + '   <div class="col-lg-4">'
                                        +  '            <input  class="form-control"
id="5" value="'+s.other.pwd+'" type="text">'
                                        + '      </div>'
                                        + '</div>'
                                        + '<div class="form-group">'
                                        +  '            <div  class="col-lg-offset-2
col-lg-10" style="text-align:right">'
```

```
                                                + '                    <button type="submit"
class="btn-u btn-u-green" id="6" onclick="copyToClipBoard(6,5,4)">复制</button>'
                                                + '            </div>'
                                                + '        </div>'
                                                + '    </div>' + '    </div>';
                                        $("#securitLink").empty().append(container);
                                        copyToClipBoard(6, 5, 4);
                                    }
                                }
                            }
                        });
            }
```

（2）Controller 层

```
    @RequestMapping("/createLink")
    @ResponseBody
    public Object createLink(HttpServletRequest request,
            HttpServletResponse response, int type, String filename,
            String filepath, String isdir, String length) {
        User sessionUser = getSessionUser(request);
        String Username = sessionUser.getUsername();
        Integer id = sessionUser.getId();
        String scheme = request.getScheme();
        String serverName = request.getServerName();
        int serverPort = request.getServerPort();
        long currentTimeMillis = System.currentTimeMillis();
        String url = request.getContextPath();
        String port = serverPort == 80 ? "" : ":" + serverPort;
        String urlpath = scheme + "://" + serverName + port + url
                + "/share.action?num=" + currentTimeMillis;
        String dateToString = DateUtil.DateToString("yyyy-MM-dd HH:mm:ss",
```

```
                new Date());
        ShareBean shareBean = new ShareBean();
        shareBean.setUserid(id);
        shareBean.setIsdir(isdir);
        shareBean.setFilename(filename);
        shareBean.setFilepath(filepath);
        shareBean.setType(type);
        shareBean.setFilelength(length);
        shareBean.setHttp(urlpath);
        shareBean.setData(dateToString);
        if (type == 2) {
            shareBean.setPwd(Code.getCode());
        }
        boolean b = shareService.saveShareFile(shareBean);
        return new MessageBean(b, "",
                new ShareBean(urlpath, shareBean.getPwd()));
    }
    /**
     * 复制链接-->地址栏
     *
     * @param request
     * @param response
     * @param num
     * @return
     */
    @RequestMapping("/share")
    public ModelAndView sharefile(HttpServletRequest request,
            HttpServletResponse response, String num) {
        String url = null;
        String number = "";
        ModelAndView view = new ModelAndView();
```

```
// List<FileBean>  list = new ArrayList<FileBean>();
Object unp = shareService.getUsernameBynum(num);
if (unp == null) {
    url = "/shareinput";
    number = "-1";
    view.addObject("num", number);
    view.setViewName(url);
    return view;
} else {
    Object[] unplist = (Object[]) unp;
    int userid = (int) unplist[3];
    String pwd = (String) unplist[5];

    if (pwd != null) {
        url = "/shareinput";
        number = num;
    } else {
        url = "/share";
        number = num;
    }
    User user = userService.getUserByid(userid);
    String username = user.getUsername();
    String rpath = unplist[2].toString();
    List list = new ArrayList();
    FileBean fileBean=new FileBean();
    fileBean.setFilepath(unplist[2].toString());
    fileBean.setIsdirectory(Boolean.parseBoolean(unplist[6].toString()));
    fileBean.setLastmodified(unplist[8].toString());
    fileBean.setLength(unplist[7].toString());
    fileBean.setName(unplist[1].toString());
    fileBean.setPath(unplist[2].toString());
```

```
                list.add(fileBean);
//              SwiftDFS swiftDFS = new SwiftDFS();
//              List lis = swiftDFS.getShareFile(username, rpath);
                view.setViewName(url);
                view.addObject("list", list);
                view.addObject("num", number);
                return view;
            }
        }
```

（3）Service 层

```
    public boolean saveShareFile(ShareBean shareBean){
        shareDao.save(shareBean);
        return true;
    }
```

有正确的截图计 1 分，若没有代码按 0 分计。

有合理的代码 1 分。

任务二、云存储网盘客户端开发（5 分）

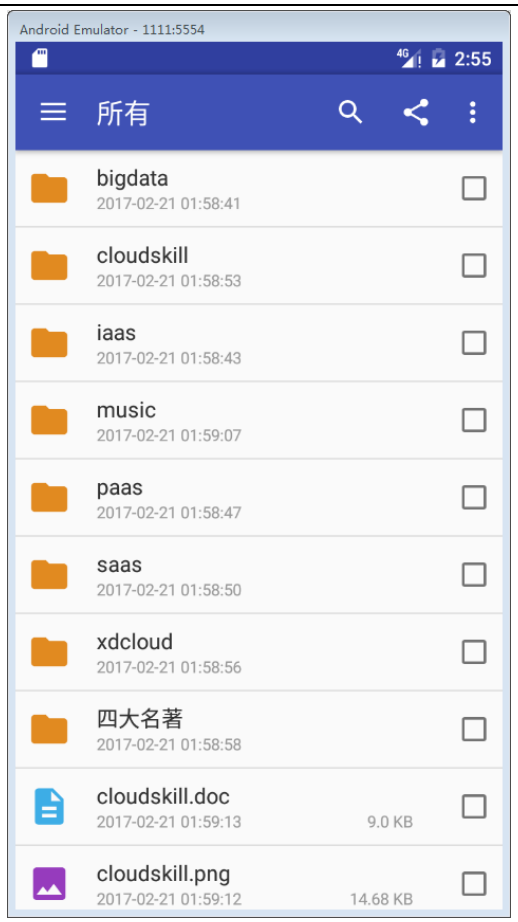
云存储网盘客户端 APP 的开发，选用 Android 开源技术平台，使用集中部署的云存储服务。

开发环境：SwiftSDK(openstack-java-sdk） + Android 开发环境（Android Studio） + JDK + 案例工程 swiftstorage，程序的运行采用 Android Studio 默认模拟器。

1.搭建开发环境和导入项目（1 分）

根据指定的账户名、密码等信息修改链接云存储平台的配置并运行。按顺序提交模拟器登录后“所有”界面截图、修改的配置及修改的配置代码到答题框。

1.效果截图



2.修改代码

需要修改，改变 IP 地址修改为公共云存储 IP192.168.1.11 改变的类包括

Appstate.java

private String openstack_ip = "192.168.1.11";

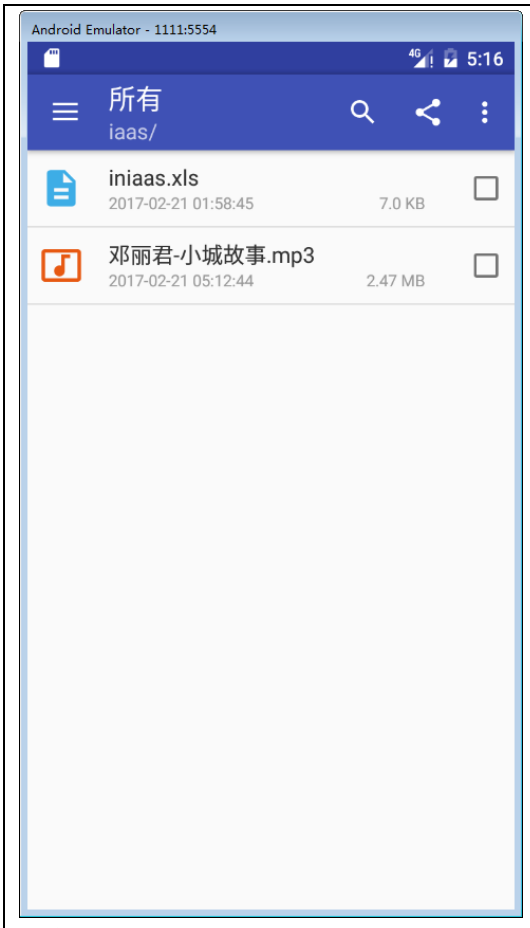
有正确的所有文件截图计 0.5 分。若没有代码计 0 分。

有合理的代码 0.5 分

2.文件的移动功能（2 分）

完善 openstack-java-sdk 中文件移动接口的底层代码，实现网盘 APP 的文件“移动”功能，将“music”文件夹中的“邓丽君-小城故事.mp3”文件移至“iaas”文件夹中。按顺序提交“iaas”文件夹截图和增改的 java 代码到答题框。

1.效果截图



2.修改代码

（1）底层添加的代码：

代码：OpenStackClientService

OpenStackClientService:

```
/**
 * 移动。
 * @param srcPath        原路径
 * @param desPath        目标路径
 * @param contentType  文件类型
 */
public void move(String containerName, String srcPath, String desPath, String contentType) {
    try {
        srcPath = URLEncoder.encode(srcPath, "utf-8");
        desPath = URLEncoder.encode(desPath, "utf-8");
        containerName = URLEncoder.encode(containerName, "utf-8");
        contentType = URLEncoder.encode(contentType, "utf-8");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    getSwift(tenantID).containers().container(containerName).move(srcPath, desPath,
contentType).execute();
```

（2）MainFragment.java

```
    /**
```

```
     * 移动
     *
     * @param fromPath：初始目录
     * @param toPath：最终目录
     */
    @Override
    public void move(String fromPath, String toPath) {
        if (getFirstSelected() != null) {
            this.ismove = true;
            this.moveFileName = getFirstSelected().getName();
            this.moveFileType = getFirstSelected().getContentType();
            fileActionBar.setVisibility(View.VISIBLE);
        }
    }



    if (ismove) {
        //是移动
        moveToFileName = getAppState().getSelectedDirectory().getName() +
cleanName(moveFileName);
        System.out.println("moveToFileName:" + moveToFileName);
        MoveObjectTask moveObjectTask = new MoveObjectTask(moveFileName,
moveToFileName, moveFileType);
        moveObjectTask.execute();
    }



    /**
     * 移动功能的线程
     */
    private class MoveObjectTask extends AsyncTask<String, Object, TaskResult<Object>> {
```

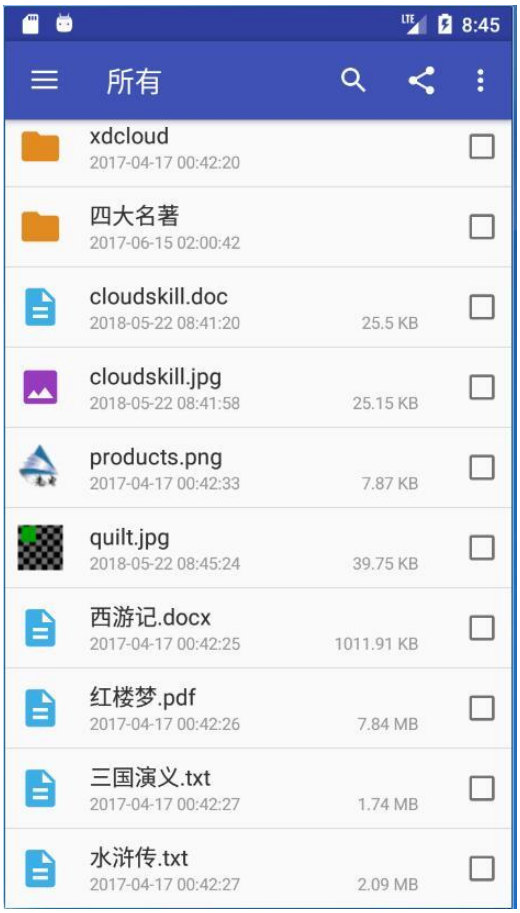
```
        private String _path, _pathTo, _type;

        private MoveObjectTask(String path, String pathTo, String type) {
            this._path = path;
            this._pathTo = pathTo;
            this._type = type;
        }

        @Override
        protected TaskResult<Object> doInBackground(String... params) {
            try {
                String
containName=getAppState().getSelectedContainer().getName().toString();
                getService().move(containName,_path,_pathTo,_type);
                return new TaskResult<Object>((Object) getAppState().getSelectedObject());
            } catch (Exception e) {
                return new TaskResult<Object>(e);
            }
        }

        /**
         * 移动完成后刷新
         *
         * @param result
         */
        @Override
        protected void onPostExecute(TaskResult<Object> result) {
            ismove = false;
            fileActionBar.setVisibility(View.GONE);
            GetOSSObjectsTask getObjectsTask = new GetOSSObjectsTask();
            getObjectsTask.execute();
```

} }
有正确的“iaas”文件夹截图计 1 分。若没有代码计 0 分。 有正确底层和实现代码各 0.5 分。

3.拍照识别上传功能（2 分）

基于 TensorFlow 提供的接口，实现网盘 APP 的“拍照识别”上传功能。使用安卓模拟器拍一张照片，识别该照片并按相似度从高到低排序（展示三个相似度高的结果），将识别度最高的结果作为照片文件名上传至根目录下。按顺序提交模拟器照片识别截图、根目录截图和增改的 java 代码到答题框。

1.效果截图



2.修改代码

（1）MainFragment:

```
@Override
public void takePhoto_upload() {
    runtakePhoto();
}

private static final int ACTION_SELECT_CONTENT_FROM_CAMERA = 3;
//拍照临时存储路径
private Uri mImageUri;
private File image_uri;

/**
 * 拍照
 */
```

```
    private void runtakePhoto() {
        try {
            //创建 Intent
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            //临时文件传递参数
            mImageUri = Uri.fromFile(createPicTempFiles());
            cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, mImageUri);
            //启动，系统自己选择
            startActivityForResult(cameraIntent,
ACTION_SELECT_CONTENT_FROM_CAMERA);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * 创建临时文件，保持文件。
     *
     * @return
     * @throws IOException
     */
    public File createPicTempFiles() throws IOException {
        String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss")
                .format(new Date());
        String imageFileName = "os_" + timeStamp + "_";
        File storageDir = Environment
                .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)
;
        File image = File.createTempFile(imageFileName,
                ".jpg",
                storageDir
```

```
        );
    image_uri = image;


        return image;
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {


        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case ACTION_SELECT_CONTENT_FROM_CAMERA:
                //成功返回
                //获取图片路径
                final File file = new File(image_uri.toString());
                Bitmap bm = null;
                if (file.exists()) {
                //根据图片路径加载手机中的图片
                bm = BitmapFactory.decodeFile(file.toString());
                //将当前的图片按一定比例创建一张新的图
                bm = Bitmap.createScaledBitmap(bm, INPUT_SIZE, INPUT_SIZE, false);
                //显示识别的结果
                List<Classifier.Recognition> results = null;
                take_photo_list =   new ArrayList<TakePhotoBean>();
                try {
                    results = classifier.recognizeImage(bm);
                    Log.i("results==",results.toString());
                //[[510] screen (52.9%), [869] monitor (29.1%)]
                String [] renameResult = null;
                renameResult                                                    =
results.toString().substring(1,results.toString().length()-1).split(",");
    Log.i("renameResult.length==",String.valueOf(renameResult.length));
```

```
                    if (renameResult.length>1){
                        for (int i = 0; i < renameResult.length; i++) {
                        //照片名字
                        String name =
renameResult[i].substring(renameResult[i].indexOf("]")+2,renameResult[i].indexOf("(")-1);
                        //照片出现的概率
                        String probability =
renameResult[i].substring(renameResult[i].indexOf("(")+1,renameResult[i].indexOf(")"));
                        if (i<3){
                        TakePhotoBean bean = new TakePhotoBean();
                        bean.setPhotoName(name);
                        bean.setNameProbability(probability);
                        take_photo_list.add(bean);
                                    }
                                }
                            }else {
                        TakePhotoBean bean = new TakePhotoBean();
                            bean.setPhotoName("暂无识别结果");
                            bean.setNameProbability("");
                            take_photo_list.add(bean);
                                }
                            }catch (Exception e){
                        TakePhotoBean bean = new TakePhotoBean();
                             bean.setPhotoName("暂无识别结果");
                             bean.setNameProbability("");
                             take_photo_list.add(bean);
                            }
    takePhoto_result_adapter = new TakePhoto_ResultAdapter(context,take_photo_list);
    //实例化 Dialog 内部类 Builder
    AlertDialog.Builder  builder = new AlertDialog.Builder(context);
    //加载布局
```

```
View view = LayoutInflater.from(context).inflate(R.layout.dialog_show_photo_msg, null);
//获取控件
final EditText edt_result = (EditText) view.findViewById(R.id.edt_Result);
ImageView img_result = (ImageView) view.findViewById(R.id.img_Result);
ListView lsv_item = (ListView) view.findViewById(R.id.lsv_item);
//赋值
img_result.setImageBitmap(bm);
lsv_item.setAdapter(takePhoto_result_adapter);
edt_result.setText(take_photo_list.get(0).getPhotoName());
lsv_item.setOnItemClickListener(new AdapterView.OnItemClickListener() {
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                    String name = take_photo_list.get(position).getPhotoName();
                    edt_result.setText(name);
                        }
                    });
view.findViewById(R.id.btnEnter).setOnClickListener(new View.OnClickListener() {
                    @Override
public void onClick(View v) {
                    //获取控件
                    EditText edt_result = (EditText)
createDirDialog.findViewById(R.id.edt_Result);
String result = panduan(edt_result.getText().toString());
                   if ("no".equals(result)){
                    String path = image_uri.toString();
                    StringBuffer buffer=new StringBuffer(path);
                    StringBuffer
newpath=buffer.replace(path.lastIndexOf("/")+1,path.lastIndexOf("."),edt_result.getText().toString
());
Log.e(buffer.toString(), newpath.toString());
                   try{
                       Boolean flag=file.renameTo(new File(newpath.toString()));
```

```
                    // 更新 ID 为重名命的记录
                    Uri uri = ContentUris.withAppendedId(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
(getAppState().getSelectedContainer().getObjectCount()));
                    ContentValues values = new ContentValues();
                    // 修改重命名后的值
    values.put(MediaStore.Images.Media.TITLE, edt_result.getText().toString());
    values.put(MediaStore.Images.Media.DATA, newpath.toString());
    // 获得 ContentResolver，并更新
    context.getContentResolver().update(uri, values, null, null);
    Toast.makeText(context, "命名成功!", Toast.LENGTH_SHORT).show();
    String new_str = "file://"+newpath.toString();
    Uri uri_new = Uri.parse((String) new_str);
    //上传图片
    UploadCameraObjectTask uploadObjectTask = new UploadCameraObjectTask(uri_new);
                                                uploadObjectTask.execute();
    //成功后关闭
    createDirDialog.dismiss();
                }catch(Exception e){
                Toast.makeText(context, "名字重复啦，请重命名!",
Toast.LENGTH_SHORT).show();
    e.printStackTrace();
                                            }
                                        }
    if("have".equals(result)){
    Toast.makeText(getContext(),"名字已存在，请重新命名",Toast.LENGTH_SHORT).show();
                                            }
                                    }
                                });


view.findViewById(R.id.btnCancel).setOnClickListener(new View.OnClickListener() {
```

```
                    @Override
                    public void onClick(View v) {
                    //取消关闭
                    createDirDialog.cancel();
                                }
                            });
builder.setTitle("拍照结果");
builder.setView(view);
createDirDialog = builder.create();
createDirDialog.show();
                    break;
                }
            default:
                break;
        }
}

/**
 * 上传拍照图片
 */
private class UploadCameraObjectTask extends
        AsyncTask<String, Object, TaskResult<ObjectForUpload>> {
    private Uri fileUri;

    /**
     * 上传拍照图片。
     *
     * @param fileUri
     */
    private UploadCameraObjectTask(Uri fileUri) {
        this.fileUri = fileUri;
```

```
        }

        protected TaskResult<ObjectForUpload> doInBackground(String... params) {
            try {
                //文件存储位置，如果拍照存本地在临时目录
                String filePath = DisplayUtils.getOriginalFilePath(
                        getActivity(), fileUri);
                //上传当前目录
                String directory = getAppState().getSelectedDirectory().getName();
                //本地临时目录只取文件名称
                String[] parts = filePath.split("/");
                String path = directory + parts[parts.length - 1];
                String contentType = "image/jpeg";
                String containerName = getAppState().getSelectedContainer().getName();
                InputStream fileInputStream = new FileInputStream(filePath);
                //调用上传服务
                ObjectForUpload objectForUpload = getService().upload(containerName,
fileInputStream, contentType, path);

                return new TaskResult<ObjectForUpload>(objectForUpload);
            } catch (Exception e) {
                return new TaskResult<ObjectForUpload>(e);
            }
        }

        protected void onPostExecute(TaskResult<ObjectForUpload> result) {
            super.onPostExecute(result);
            if (result.isValid()) {
                //刷新当前目录
                GetOSSObjectsTask getObjectsTask = new GetOSSObjectsTask();
                getObjectsTask.execute();
```

```
                } else {
                    //上传失败
                    PromptDialogUtil.showErrorDialog(
                            getActivity(),
                            R.string.alert_take_photo_fail,
                            result.getException(),
                            new Intent(getActivity(), LoginActivity.class));
                }
            }
        }
        /**
         * 拍照后自动识别结果 adapter
         *
         * @author Administrator
         *
         */
        public class TakePhoto_ResultAdapter extends BaseAdapter {

            //列表中的数据
            private List<TakePhotoBean> take_photo_listview;
            //对应的 context
            private Context context;
            /**
             * 构造器。
             *
             * @param context
             */
            public TakePhoto_ResultAdapter(Context context, List<TakePhotoBean>
take_photo_listview) {

                this.take_photo_listview = take_photo_listview;
```

```
        this.context = context;
        //默认按照名称进行排序
    }
    /**
     * 数量。
     *
     * @return
     */
    @Override
    public int getCount() {
        return take_photo_listview.size();
    }
    /**
     * 按照索引号查询。
     *
     * @param index
     * @return
     */
    @Override
    public java.lang.Object getItem(int index) {

        return take_photo_listview.get(index);
    }
    /**
     * 对象 ID 同位置一致。
     *
     * @param position
     * @return
     */
    @Override
    public long getItemId(int position) {
```

```
            return position;
        }
        /**
         * 获取展示当前 Item 的视图，可以根据布局创建，也可以自己创建。
         *
         * @param position      索引位置。
         * @param convertView  旧的展示视图，可能为 null，如果为 null，需要创建。
         * @param parent        父视图。
         */
        @Override
        public View getView(final int position, View convertView, ViewGroup parent) {
            //根据布局创建 main_list_item.xml
            if (convertView == null) {
                convertView = LayoutInflater.from(context).inflate(R.layout.adapter_takephoto_result_item, null);
            }
            TextView tev_name = (TextView) convertView.findViewById(R.id.tev_name);
            TextView tev_probability = (TextView) convertView.findViewById(R.id.tev_probability);
tev_name.setText(take_photo_listview.get(position).getPhotoName());
tev_probability.setText(take_photo_listview.get(position).getNameProbability());
            return convertView;
        }
    }
```

有正确的识别照片截图计 0.5 分，有正确的根目录截图计 0.5 分。若没有代码计 0 分。

有正确的实现代码 1 分。

注：拍照图片为动态图片，识别结果可以不一样，只需合理就行。

任务三、大数据案例开发（8 分）

开发环境：Eclipse + JDK + Tomcat + Mysql + MongoDB + HBase +xueqing-client.zip +

xueqing-server.zip + xueqing-web.zip。

1.搭建开发环境和导入项目（1 分）

检查已安装的 MongoDB，HBase，MySQL 和岗位网站服务，进行大数据学情应用开发准备：

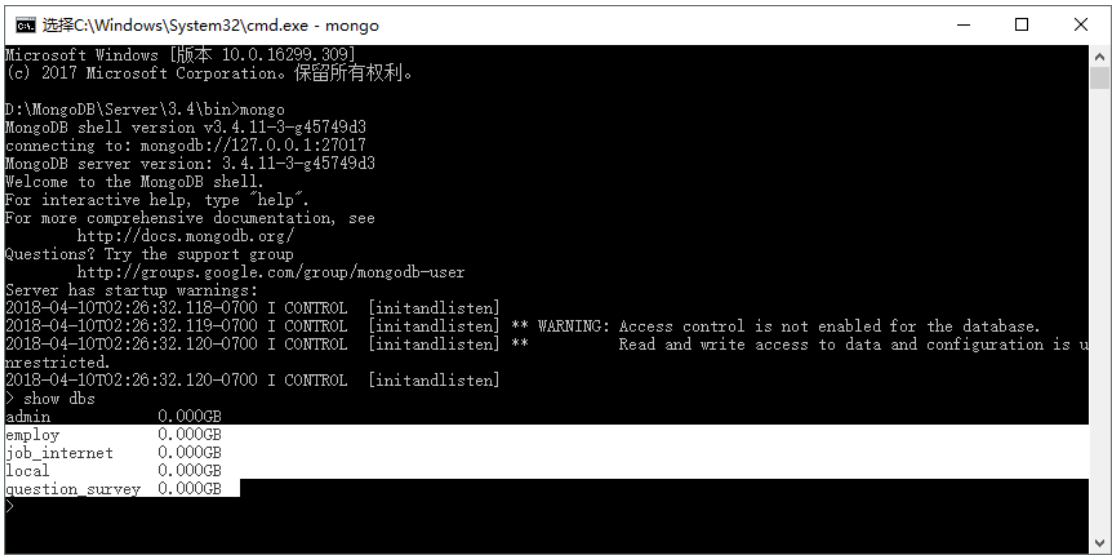
（1）导入三个 MongoDB 数据库目录 employ，job_internet，question_survey；

（2）导入 MySQL 的 xueqing-client 项目的 sql 文件 xueqing-client.sql；

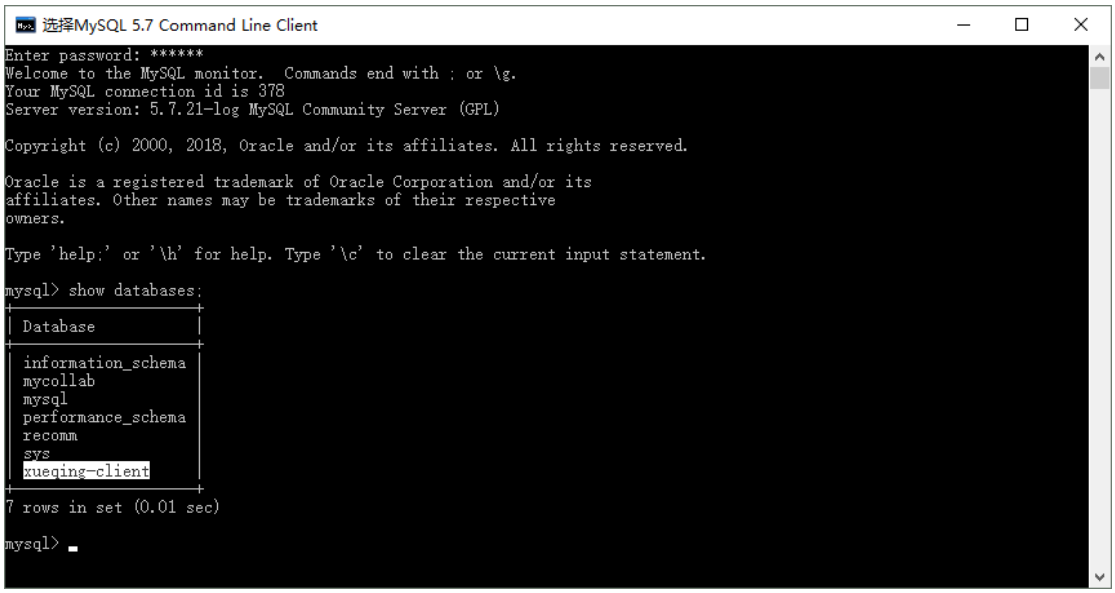
在 cmd 中登录 MongoDB 数据库，提交查询所有数据库的截图到答题框。

在 cmd 中登录 MySQL 数据库，提交查询所有数据库的截图到答题框。

1.效果截图



MongoDB 的 employ，job_internet，question_survey 三个数据库

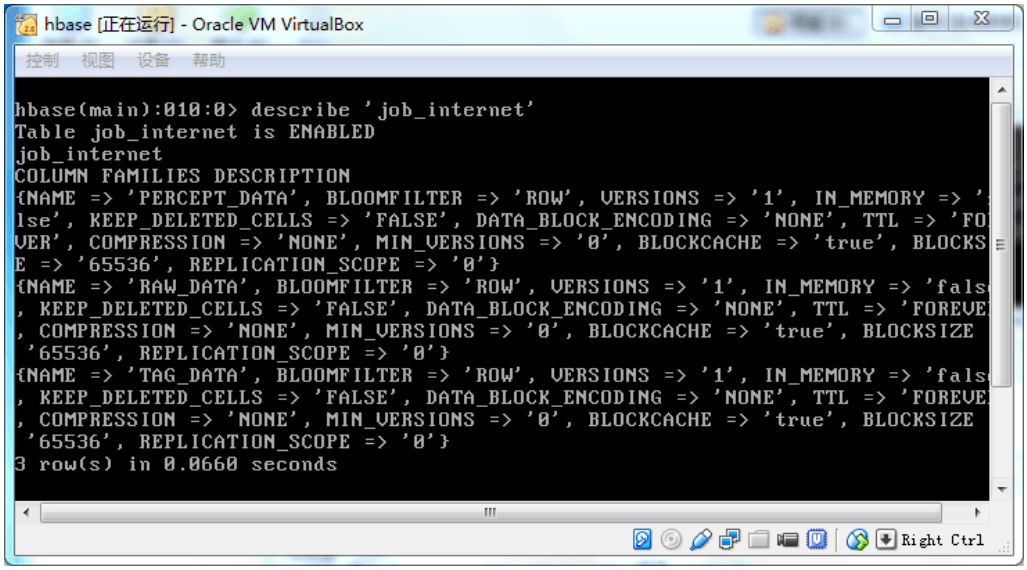


MySQL 中的 xueqing-client 数据库
有正确的 MongoDB 数据库截图计 0.5 分。 有正确的 MySQL 数据库截图计 0.5 分。

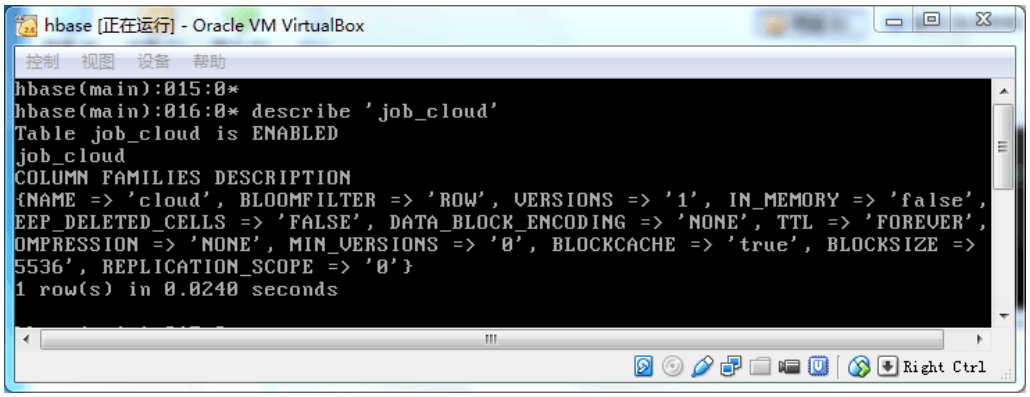
2.HBase 建表操作（1 分）

完成 xueqing-server 中 HBase 数据库建表，建立 job_internet 表（列簇为 PERCEPT_DATA、RAW_DATA、TAG_DATA）和 job_cloud 表（列簇为 cloud）。提交 HBase 两个表信息的查询结果截图和代码到答题框。

1.效果截图



HBase 的 job_internet 表查询



HBase 中的 job_cloud 表查询

2.修改代码

```
//**
        * 创建表
        */
```

```
public void createTable(String tablename,String families) throws IOException {
    // 新建一个数据表表名对象
    TableName tableName = TableName.valueOf(tablename);
    // 如果需要新建的表已经存在
    if (admin.tableExists(tableName)) {
        logger.info("表已经存在！");
    } else {
        logger.info("表创建 start");
        // 数据表描述对象
        HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);
        // 列族描述对象
        for (String fam:families.split(",")) {
        HColumnDescriptor family = new HColumnDescriptor(fam);
        // 在数据表中新建一个列族
        hTableDescriptor.addFamily(family);
        }
        // 新建数据表
        admin.createTable(hTableDescriptor);
        logger.info("表创建成功");
    }
}
```

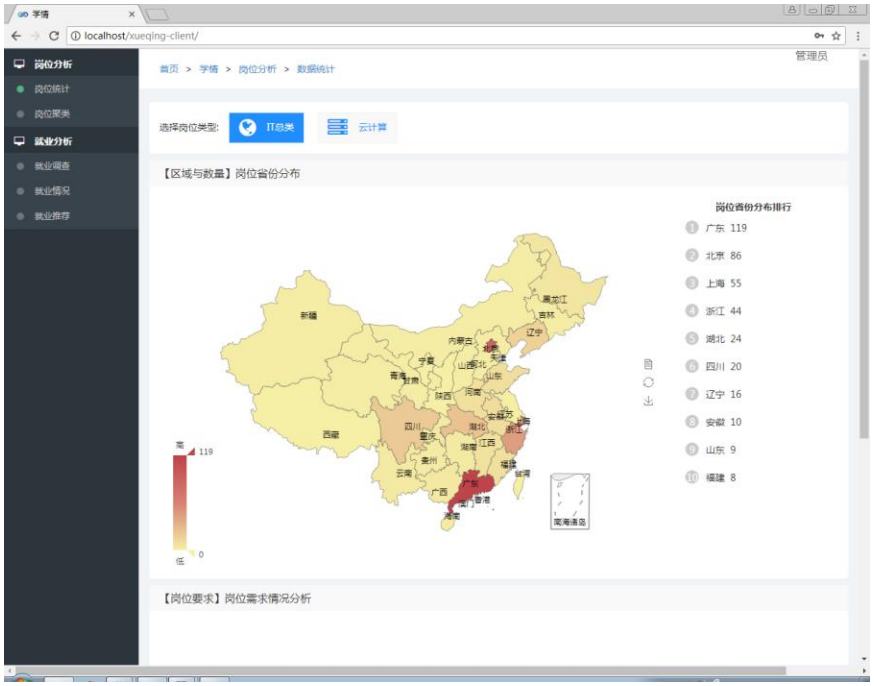
有正确的数据库查询截图计 0.5 分。若没代码计 0 分。

有正确的实现代码计 0.5 分。

3.岗位地区分布可视化（1 分）

爬取 xueqing-web 网站所有的岗位信息，对数据进行清洗，统计“云计算”岗位信息，将统计数据保存至 MongoDB 之中。通过 xueqing-client 展示云计算岗位地区分布图表。将统计图表的网页截图和相关代码提交到答题框。

1.效果截图



2.修改代码

```
/**
        * 岗位省份分布
        *
        * @param stmt
        * @param tableName
        * @throws SQLException
        * @throws IOException
        */
    public void countProvinceDistribution(String tableName, String mongoTable, String family) throws SQLException, IOException {
            MongoClient mongoClient = mongodbstorage.setUp();
            // String sql = "select LOCATION,count(1) as count,sum(AMOUNT) from " +
            // tableName
            // + " where ISPERCEPTED = 'no' group by LOCATION order by count desc";
            // ResultSet results = stmt.executeQuery(sql);

            // 建立表的连接
            Table table = connection.getTable(TableName.valueOf(tableName));
            // 创建一个空的 Scan 实例
```

```
                Scan scan1 = new Scan();
                // 可以指定具体的列族列
                scan1.addColumn(Bytes.toBytes(family),
Bytes.toBytes("LOCATION")).addColumn(Bytes.toBytes(family),
                        Bytes.toBytes("AMOUNT"));
                scan1.setCaching(60);
                scan1.setMaxResultSize(1 * 1024 * 1024); // 100k （MB1 * 1024 * 1024）
                scan1.setFilter(new PageFilter(1000));

                // 在行上获取遍历器
                ResultScanner scanner1 = table.getScanner(scan1);

                Map map = mongodbstorage.create(mongoTable, "job_province_distribution",
mongoClient);// mongodb 集合,key,value
                MongoCollection<Document> collection = (MongoCollection<Document>)
map.get("collection");
                Document document = (Document) map.get("document");
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                java.util.Date date = new java.util.Date();
                String da = sdf.format(date);
                mongodbstorage.appendString(document, "date", da);
                int[] cu = new int[34];
                int[] nu = new int[34];
                Vector<Document> vec = new Vector<Document>();
                for (Result res : scanner1) {
                    String loc = (new String(CellUtil.cloneValue(res.rawCells()[1]))).split("-")[0];
                    int total = Integer.parseInt(new
String(CellUtil.cloneValue(res.rawCells()[0])));
                    int num = 1;
                    if (!cities.containsKey(loc)) {
                    } else {
```

```
                    // String prov =
jobanalysisreposity.appendProvinceDistribution(document, loc,
                    // total, mongoClient);
                    String prov = cities.get(loc);
                    switch (prov) {
                    case "北京":
                        cu[0] += total;
                        nu[0] += num;
                        break;
                        ... ...
                    }
                    logger.info(loc + ":" + total);
                }
                Document[] doc = new Document[34];
                for (int i = 0; i <= 33; i++) {
                    doc[i] = new Document();
                }
                mongodbstorage.appendProvince(doc[0], "北京", cu[0], nu[0]);
                ... ...
                for (int i = 0; i <= 33; i++) {
                    vec.add(doc[i]);
                }
                mongodbstorage.appendArray(document, "category", vec);
                mongodbstorage.insertOne(collection, document);
            }
```

有正确的岗位统计图表截图计 0.5 分。若没代码计 0 分。

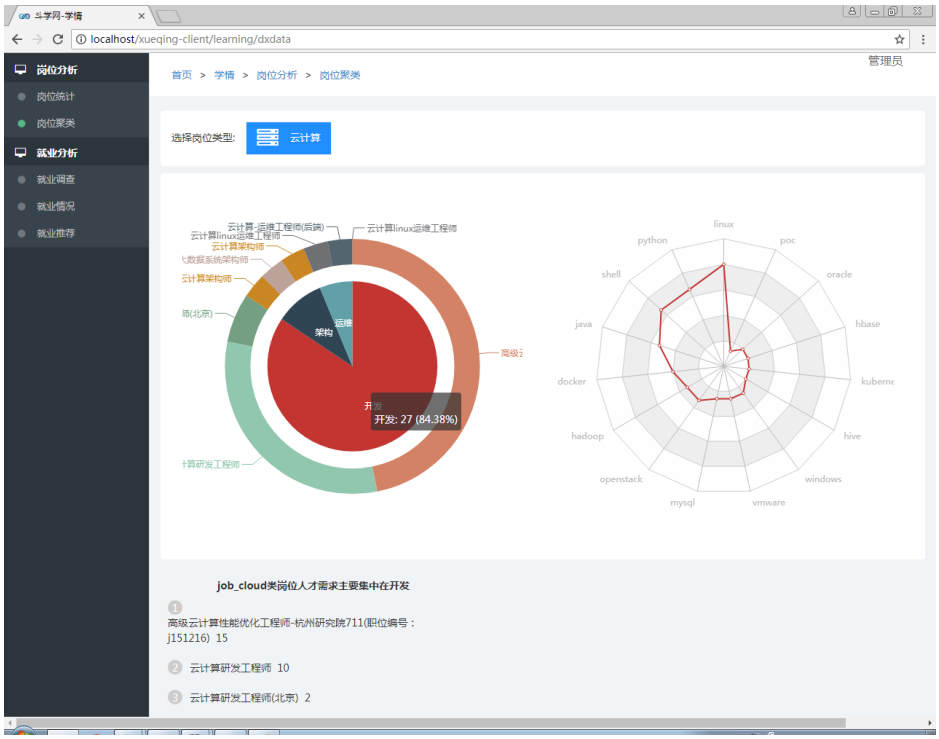
有正确的实现代码计 0.5 分。

4.岗位聚类（2 分）

对云计算岗位数据进行岗位聚类分析，将聚类分析结果数据保存至 MongoDB 之中，通过 xueqing-client 展示云计算岗位聚类结果的饼图，将聚类饼图的网页截图和代码提交到答

题框。

1.效果截图



2.修改代码

```
/**
        * 岗位循环聚类
        *
        * @param industry
        *                      岗位行业
        * @param category
        *                      岗位行业类别
        * @param dictionary
        *                      岗位技能点词典
        * @param clusterN
        *                      聚类数量
        * @param mongoClient
        */
       public void cluster(String industry, String category, List<String> dictionary, int clusterN,
                       MongoClient mongoClient) {
               List<String> id = new ArrayList<>();
```

```
            List<String> jobname = new ArrayList<>();


            try {
//                id = jobDataReposity.queryDataByColumn("job_" + industry, industry, "ID");
//                jobname = jobDataReposity.queryDataByColumn("job_" + industry, industry,
"JOB_NAME");
                id = jobDataReposity.queryDataByColumn("job_" + industry, industry, "ID");
                jobname = jobDataReposity.queryDataByColumn("job_" + industry, industry,
"JOB_NAME");
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }


            Map<String, Map<String, String>> allJob = allJobClassification(id, jobname,
industry);// 将取出的岗位群分类
            List<String> ids = new ArrayList<>();
            if (allJob != null) {
                Map<String, String> category1 = allJob.get(category);// 取小类
                for (Map.Entry<String, String> m : category1.entrySet())
                    ids.add(m.getKey()); // 取出所有开发类对应的 ID
                if (ids.size() > 1) {
                    List<String> jdes = new ArrayList<>();
                    List<String> jname = new ArrayList<>();
                    try {
                        jdes = jobDataReposity.queryTableByCondition(ids, "job_" +
industry, industry,
                                "ID", "DESCRIPTION");
                        // jdes = jobDataReposity.queryTableByCondition(ids,
"job_internet" ,
                        // "TAG_DATA",
```

```
                    // "ID", "DESCRIPTION");// 根据ID取出所有开发类的岗位描述
                    jname = jobDataReposity.queryTableByCondition(ids, "job_" +
industry, industry,
                            "ID", "JOB_NAME");
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                tfKmeans(jdes, jname, dictionary, industry, category, clusterN,
mongoClient);// 对开发类岗位描述进行聚类
            } else
                logger.info(industry + "-" + category + " not enough data.");
        }
    }
```

有正确的云计算岗位聚类饼图计 1 分。若没代码计 0 分。

有正确的实现代码计 1 分。

5.岗位推荐（3 分）

给定某个用户的技能数据，使用机器学习推荐算法，实现岗位推荐。为该用户推荐三个最佳的招聘岗位，并通过 EChart 图表展示推荐的岗位和技能对比，将展示的图表截图和代码提交到答题框。

说明：

（1）某用户技能数据：Java 技能 12 个月，MySQL 技能 6 个月，Python 技能 6 个月，HTML 技能 6 个月，Swift 技能 4 个月，MongoDB 技能 4 个月，Hbase 技能 4 个月，Javascript 技能 4 个月，Bootstrap 技能 2 个月，如图 3 所示。

（2）机器学习算法采用 Mahout 的 UncenteredCosineSimilarity 相似度算法和 NearestNUserNeighborhood 用户近邻算法。

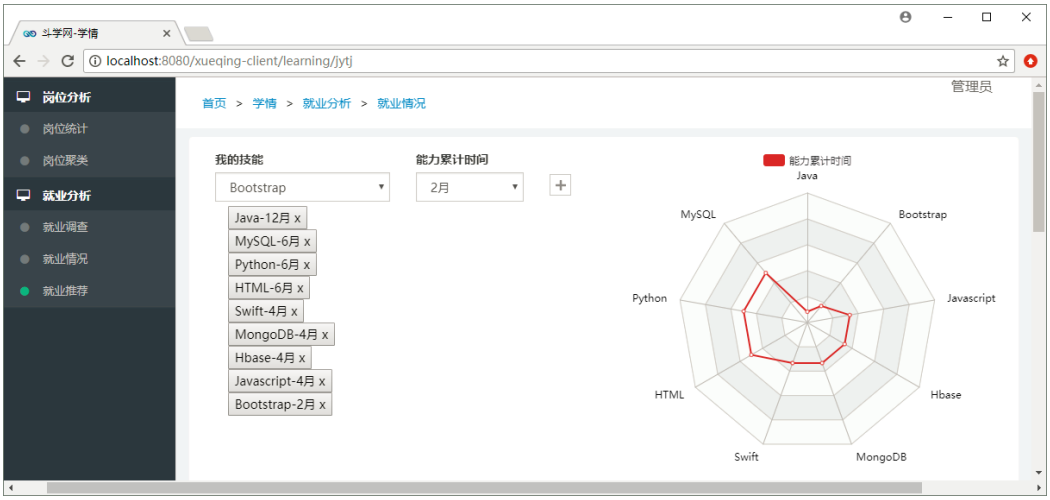
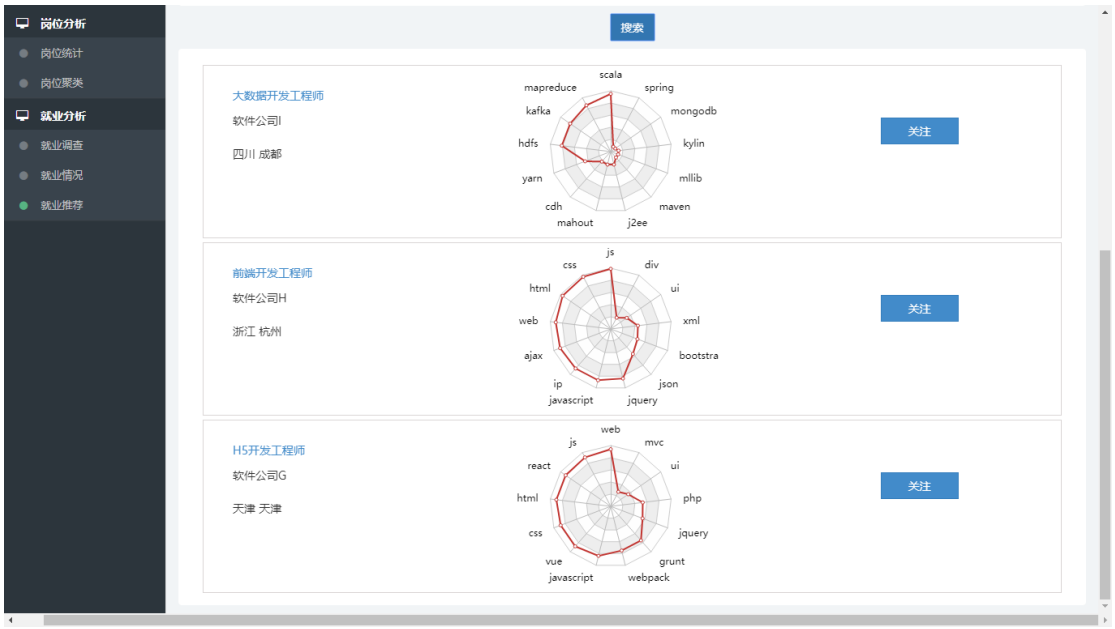


图 3 用户技能数据图

1.效果截图



2.修改代码

```
/**
        * 收集所有技能点。
        *
        * @param skillIdMaps
        * @param skills
        * @return
        */
       private static ArrayList<String> generateSkillIDs(ArrayList<String> skillIdList, String[]
skills) {
            for (String skill : skills) {
```

```
            String value=skill.trim().toLowerCase();
            int index = skillIdList.indexOf(value);
            if (index == -1) {
                skillIdList.add(value);
            }
        }
        return skillIdList;
    }

    /**
     * 产生当前 skill 的 IDs
     *
     * @param skillIdMaps
     * @param skills
     * @return
     */
    private static int[] getSkillIDs(ArrayList<String> skillIdList, String[] skills) {
        int[] ids = new int[skills.length];
        for (int i = 0; i < ids.length; i++) {
            String value=skills[i].trim().toLowerCase();
            ids[i] = skillIdList.indexOf(value);
        }
        return ids;
    }

    /**
     * 就业推荐数据信息从表 employ/job 查询，通过 Mahout 的相似算法进行推荐。
     *
     * @param mongoClient
     *                  存储在 employ/job 中。
     * @param skills
```

```
     *                  当前输入技能列表
     * @param weights
     *                  当前输入技能对应权重的列表
     * @param recommNum
     *                  推荐岗位个数
     * @return 推荐的岗位列表
     */
    public static Map<String, Object> getjytj(MongoClient mongoClient, String[] skills,
String[] weights, int number) {
        Map<String, Object> map = new HashMap<>();

        ArrayList<String> totalSkills = new ArrayList<>();// 所有技能点
        generateSkillIDs(totalSkills, skills);
        MongoDatabase mongoDatabase = mongoClient.getDatabase("employ");
        MongoCollection<Document> collection = mongoDatabase.getCollection("job");
        FindIterable<Document> findIterable = null;
        findIterable = collection.find();
        MongoCursor<Document> mongoCursor = findIterable.iterator();
        List<Map<String, Object>> maps = new ArrayList<>();// 存放所有的职位信息
        while (mongoCursor.hasNext()) {
            Document doc = mongoCursor.next();
            List<Document> jobs = (List<Document>) doc.get("jobs");
            for (Document s : jobs) {
                Map<String, Object> jobmes = new HashMap<>();
                String skill = s.getString("skills");
                jobmes.put("id", s.getString("id"));
                jobmes.put("provice", s.getString("provice"));
                jobmes.put("city", s.getString("city"));
                jobmes.put("job_name", s.getString("job_name"));
                jobmes.put("skill", skill);
                jobmes.put("weight", s.getString("weight"));
```

```
                    jobmes.put("company_name", s.getString("company_name"));
                    String[] jobskills = skill.split(",");
                    generateSkillIDs(totalSkills, jobskills);
                    maps.add(jobmes);
                }
            }
            // 产生所有岗位的数据模型(多维的向量数组）
            FastByIDMap<PreferenceArray> userData = new
FastByIDMap<PreferenceArray>();
            for (Map<String, Object> jobmes : maps) {
                int id = Integer.parseInt((String) jobmes.get("id"));
                String skill = (String) jobmes.get("skill");
                String[] skills_ = skill.split(",");
                // 技能转换成 ID
                int[] skillids = getSkillIDs(totalSkills, skills_);
                // 技能对应得权重
                double[] weightsparam = new double[skillids.length];
                String weight_ = (String) jobmes.get("weight");
                String[] weights_ = weight_.split(",");
                // mongodb 岗位库的一个岗位所有的技能
                GenericPreference[] genericPreferences = new
GenericPreference[skillids.length];
                for (int i = 0; i < skillids.length; i++) {
                    // id 岗位 ID skillids[i] 技能 ID Float.parseFloat(weights_[i]) 权重
                    genericPreferences[i] = new GenericPreference(id, skillids[i],
Float.parseFloat(weights_[i]));
                }
                userData.put(id, new
GenericUserPreferenceArray(Arrays.asList(genericPreferences)));
            }
            // 输入的用户的技能 默认 ID 为 0 库从 1 开始
```

```
            GenericPreference[] userGenericPreferences = new
GenericPreference[skills.length];
            for (int i = 0; i < userGenericPreferences.length; i++) {
                int[] userskillids = getSkillIDs(totalSkills, skills);
                userGenericPreferences[i] = new GenericPreference(0, userskillids[i],
Float.parseFloat(weights[i]) / 12);
            }
            userData.put(0, new
GenericUserPreferenceArray(Arrays.asList(userGenericPreferences)));
            // 偏好向量组生成数据模型
            DataModel model = new GenericDataModel(userData);
            try {
                // 相识度算法（使用该算法得 0.5 分）
                UserSimilarity userSimi = new UncenteredCosineSimilarity(model);
                // 近邻算法 number 推荐个数
                NearestNUserNeighborhood neighbor = new
NearestNUserNeighborhood(number, userSimi, model);
                //
                // 获得输入用户的相近岗位 0 代表用户输入 ID
                long[] ids = neighbor.getUserNeighborhood(0);
                List<List<RaderObj>> raderObjs = new ArrayList<>();
                List<List<Double>> values = new ArrayList<>();
                List<Map<String, Object>> job = new ArrayList<>();
                //遍历 3 个推荐的岗位（实现以下代码得 0.5 分）
                for (long id : ids) {
                    //在岗位库 寻找这 3 个岗位
                    for (Map<String, Object> jobmes : maps) {
                        int jobId = Integer.parseInt((String) jobmes.get("id"));
                        if (id == jobId) {
                            List<RaderObj> raderObj = new ArrayList<>();//雷达图展示
                            Map<String, Object> jobMap = new HashMap<>();//岗位的名
```

```
称 公司 地址
                    List<Double> jobWeights = new ArrayList<>();//技能点的权
重
                    jobMap.put("localtion",  jobmes.get("provice") + " " +
jobmes.get("city"));
                    jobMap.put("jobname",  jobmes.get("job_name"));
                    jobMap.put("companyname", jobmes.get("company_name"));
                    String skill = (String) jobmes.get("skill");
                    String weight= (String) jobmes.get("weight");
                    String[] jobSkills = skill.split(",");
                    String[] wes = weight.split(",");
                    //每个技能点生成一个雷达对象
                    for (String s : jobSkills) {
                        RaderObj obj = new RaderObj();
                        obj.setMax(1);
                        obj.setName(s);
                        raderObj.add(obj);
                    }
                    //每个雷达的技能权重
                    for (String w : wes) {
                        jobWeights.add(Double.parseDouble(w));
                    }
                    raderObjs.add(raderObj);
                    job.add(jobMap);
                    values.add(jobWeights);
                }
            }

        }
        map.put("job", job);//推荐的岗位信息
        map.put("radar", raderObjs);//雷达技能图
```

```
            map.put("values", values);//技能图对应得权重值
        } catch (TasteException e) {
            e.printStackTrace();
        }
        return map;
    }
```

有正确的推荐岗位截图计 1 分，公司信息、区域信息和技能点的雷达图正确计 1 分。若没代码计 0 分。

有正确的实现代码计 1 分。

任务四、微信小程序开发（2 分）

开发环境：Egret Wing+ O2OMall。

1.商店界面开发（2 分）

参照图 4 小程序页面的布局、元素和配色，实现微信 o2o 商城的“商店”界面开发。将“商店”界面截图和代码提交到答题框。

图 4 商店界面效果图

1.修改代码

首先需要在 pages/store/wxss 的 style.wxss 文件中添加代码

```
    view {
    /*    color:#81c7d1; */
    color:#232323;
    font-size:32rpx;
}
button[type="primary"][plain] {
    border: 1px solid #81c7d1;
    color: #81c7d1;
}
button[type="primary"] {
    color:#FFFFFF;
    background-color:#81c7d1;
}
.topbar{
    background: #f5f5f5;
    line-height:  64rpx;
    position:  fixed;
    width: 100%;
    z-index: 1000;
    border-bottom: 2rpx solid #e1e1e1;
}
.topbar-2nd {
    margin-top:90rpx;
    background: #f3f2f7 !important;
}
.topbar image {
    width: 48rpx;
    height: 48rpx;
    margin:20rpx;
    margin-right: 10rpx;
```

```
        margin-bottom:0;
    }
    .topbar .avatar {
        border-radius: 50%;
    }
    .topbar text{
        height: 48rpx;
        font-size:28rpx;
        line-height:  88rpx;
        margin-left: 10rpx;
        margin-top: 5rpx;
    }
    .font-bold {
        font-weight: 500;
    }
    .font-s16 {
        font-size:32rpx;
    }
    .font-s15 {
        font-size:30rpx;
    }
    .font-s14 {
        font-size:28rpx;
        line-height:  36rpx;
    }
    .text-primary {
        color:#FF6347;
    }
    .view-row{
        display:  flex;
        flex-direction:row;
```

```
    }
    .block { display: flex; flex-direction:column; }
    .block .row { flex-direction: row; }
    .block .column { flex-direction: column; }
    .block .block-content { padding: 20rpx;  width: 100%;  padding-left:0; padding-right:0; }
    .block .block-title { padding: 20rpx; width: 100%; padding-left:0; padding-right:0;
font-size:36rpx; font-weight: 500; }
    .block .form-group {
        margin-top:20rpx;
        margin-left:40rpx;
        margin-bottom:20rpx;
    }
    .block .form-group label {
        margin-top:10rpx;
    }
    .block .form-group input {
        margin-top:10rpx;
        border-bottom: 2rpx solid #f2f2f2;
    }
    .block .form-group picker {
        padding-top:10rpx;
        padding-bottom: 20rpx;
        border-bottom: 2rpx solid #f2f2f2;
    }
    .table-list {
        display: flex;
        flex-direction:column;
    }
    .table-list .row  {
        display: flex;
        flex-direction:row;
```

```
}
.table-list .column { flex-direction: column; }
.table-list .row image   {
    width:100%;
}
.table-list .row view{
    height: auto;
}
.view-column {
    display: flex;
    flex-direction:column;
}
.border-1 {
    border: 2rpx solid #f2f2f2;
}
.border-t-1 {
    border-top: 2rpx solid #f2f2f2;
}
.border-b-1 {
    border-bottom: 2rpx solid #f2f2f2;
}
.border-l-1 {
    border-left: 2rpx solid #f2f2f2;
}
.border-r-1 {
    border-right: 2rpx solid #f2f2f2;
}
.border-item-1 {
    border-top: 2rpx solid #ffffff;
    border-bottom: 2rpx solid #f2f2f2;
}
```

```
.pull-left {
    float: left;
}
.pull-right{
    float: right;
}
.push-t-5 {
    margin-top:10rpx;
}
.push-l-5 {
    margin-left:10rpx;
}
.push-t-10 {
    margin-top:20rpx!important;
}
.push-t-15 {
    margin-top:30rpx !important;
}
.push-b-10 {
    margin-bottom:20rpx;
}
.push-l-10 {
    margin-left:20rpx;
}
.push-l-20 {
    margin-left:40rpx;
}
.push-r-10 {
    margin-right:20rpx;
}
.push-r-20{
```

```
    margin-right:40rpx;
}
.pad-5-t {
    padding-top:10rpx;
}
.pad-5 {
    padding:10rpx;
}
.pad-10 {
    padding:20rpx;
}
.pad-20 {
    padding:40rpx;
}
.pad-40 {
    padding:80rpx;
}

.rm-padding {
    padding:0 !important;
}
.clear {
    clear: both;
}
.text-muted {
    color: #a2a2a2 !important;
}
.text-warn {
    color:#E64340 !important;
}
.text-right {
```

```
    text-align: right !important;
}
.bg-grey {
    border: 1px solid #f5f5f5;
    background: #f5f5f5;
}
.bg-primary {
    color: #ffffff;
    border: 1px solid #81c7d1;
    background: #81c7d1;
}
.overhidden {
    overflow: hidden;
    white-space:nowrap;
    text-overflow:ellipsis
}

.hidden  {
    display: none !important;
}
.numinput input {
    display:inline-block;
    font-size:28rpx;
    height:32rpx;
    min-height: 1em;
    width:32rpx;
    border:2rpx solid #81c7d1;
    padding:0px;
    text-align:center;
}
.phrow {
```

```
        line-height:  100rpx;
        height:100rpx;
    }
    .bottombar {
        bottom: 0px;
        position:  fixed;
        line-height:  100rpx;
        height:100rpx;
        width: 100%;
        z-index: 1000;
    }
    .bottombar text {
        font-weight: 400;
        font-size: 30rpx;
    }
    .bottombar image {
        width: 60rpx;
        height: 60rpx;
        margin:20rpx;
        margin-bottom:0;
    }
    //商店界面
    1.在 pages/store/products/list 文件的 list.js 文件的 data 模块中加入代码
        goods:[
    {   id:   "1",corver:"/res/manager/tabs/bg1.jpg",name:"VR  虚  拟  现  实  课  程
",show_price:"120",sale_price:"120"},
    ],
    imgUrls:  [
    {
    url: '/res/manager/tabs/bg1.jpg'
    }, {
```

```
url: '/res/manager/tabs/bg2.jpg'
}
],
indicatorDots: true,
autoplay: true,
interval: 5000,
duration: 1000,
menubar:[
{ img: "/res/icons/1.png", name: "账户余额", link: "accounts"},
{ img: "/res/icons/2.png", name: "充值记录", link: "payin"},
{ img: "/res/icons/3.png", name: "消费记录", link: "payin"},
{ img: "/res/icons/4.png", name: "个人资料", link: "user"}
],
    },
    // 链接到详情页
    detail:function(e){
        var data = e.target.dataset;
        wx.navigateTo({ url: '/pages/store/products/detail/detail?_id=' + data.id });
```

2.在 pages/store/products/list 文件的 list.wxml 文件中加入代码

```
<view class="top">
  <swiper indicator-dots="{{indicatorDots}}" autoplay="{{autoplay}}"
interval="{{interval}}" duration="{{duration}}">
<block wx:for="{{imgUrls}}" wx:key="_imgUrls">
<swiper-item>
<navigator hover-class="navigator-hover">
<image src="{{item.url}}" class="slide-image"/>
</navigator>
</swiper-item>
</block>
</swiper>
```

```
</view>
<view class="myrow">
<view class="img" wx:for="{{menubar}}" wx:key="_menubar">
<image src="{{item.img}}" bindtap="{{item.link}}" class="imgbar"></image>
<view class="name">{{item.name}} </view>
</view>
</view>
<view class="table-list">
    <view class="row border-item-1 pad-10" style="padding-top:26rpx;"
wx:for="{{goods}}" wx:key="_id" >
    <view style="width:200rpx;" class="push-r-10"> <image bindtap="detail"
style="height:70rpx;width:100rpx" data-id="{{item._id}}" mode="widthFix"
src="{{item.corver}}"></image> </view>
        <view style="width:50%" class="push-r-10">
<view class="hiddenover">
<text bindtap="detail" data-id="{{item._id}}" class="font-s15
item-name">{{item.name}}</text>
</view>
<view>
<text bindtap="detail" data-id="{{item._id}}" class="price">
￥{{item.show_price}}</text>
</view>
</view>
<view style="width:30%;min-width:160rpx;">
<button size="mini" bindtap="addtocard" data-id="{{item._id}}" type="primary"
data-price="{{item.sale_price}}" class="buybutton"
data-amount="1" bindtap="detail">购买</button>
</view>
</view>
    </view>
```

3.在 pages/store/products/list 文件的 list.wxss 文件中添加代码

```
@import "../../wxss/style.wxss"

.slide-image{
width:400px;
height:150px;
}
.imgbar{
    position: relative;
    left:10rpx;
    width: 35px;
    height: 35px;
    background-color: #ffffff;
}

.bottombar {
        bottom: 0px;
        border: 1px solid #ffffff;
    border-top: 1px solid #EEEEE0;
        background: #ffffff;
        color:#ffffff;
        position:  fixed;
        line-height:  100rpx;
        height:100rpx;
        width: 100%;
        z-index: 1000;
}

.bottombar text {
        color: #ffffff;
        font-weight: 400;
        font-size: 30rpx;
```

```
}
.bottombar image {
        width: 60rpx;
        height: 60rpx;
        margin:20rpx;
        margin-bottom:0;
}
.text-mimi{
        font-size: 25rpx;
        color: #81c7d1;
        margin-left: 10rpx;
}

.hiddenover{
      width: 400rpx;
      overflow: hidden;
      text-overflow: ellipsis;
      white-space: nowrap;
      margin-top: -5px;
}

.price{
    color: #ff2150;
    position: absolute;
    margin-top:10rpx;
    font-size:30rpx;
}

.item-name{
    font-size: 28rpx;
}
```

```
.myrow{
    height: 180rpx;
    margin-top: 70rpx;
    width: 780rpx;
}

.img{
    position: relative;
    display: inline;
    margin-left: 20rpx;
}

.name{
     position: relative;
    font-size: 23rpx;
    display: inline;
    top: 40rpx;
    left:-70rpx;
    z-index: 100;
}

.slide-image{
        width: 100%;
}
```

有合理的“商店”界面截图计 1 分，若没代码计 0 分。

有正确的实现代码计 1 分。

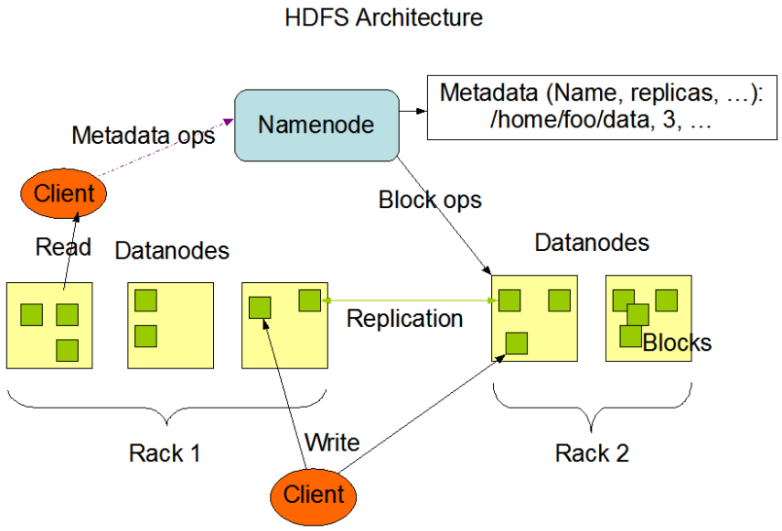
第六部分：文档及职业素养（共 10 分）

任务一、工作总结报告（5 分）

1.架构说明（3 分）

绘制 Hadoop 分布式存储 HDFS 的架构图，并对架构图进行简述。

1.架构图



2.解释说明

HDFS 使用主从架构。一个 HDFS 集群是由一个 NameNode 和多个 DataNode 组成，NameNode 是一个管理文件系统命名空间和调节客户端访问文件的主服务器，DataNode 是管理对应节点的存储，通常是一个服务器部署一个 DataNode。

一个文件被分割为一个或多个数据块，这些数据块被存储在不同的 DataNode。NameNode 用于操作文件系统的命名空间，例如打开、关闭和重命名文件和目录，同时决定了数据块和 DataNode 的映射关系。

DataNode 用于处理数据的读写请求，同时执行数据块的创建，删除和来自 NameNode 的复制指令。

架构图有 DataNode 和 NameNode 及关系计 1 分，有数据的读写操作计 1 分。

有正确的架构图解释计 1 分。

2.命令解释（2 分）

简述 Linux 中 yum 命令与 rpm 命令的区别？如何使用 rpm 命令强制安装有依赖的安装包？

1.解释说明 （1）yum 命令可以自动处理依赖性关系，一次安装所有依赖的软体包；而 rpm 命令不能自动处理依赖关系。 （2）使用 rpm 命令强制安装有依赖的包时需要加参数--nodeps 忽略依赖，否则无法安装。
说明了 yum 命令和 rpm 命令区别计 1 分。 使用--nodeps 参数计 1 分。

任务二：职业素养（5 分）

1.职业素养（5 分）

根据各参赛队工作作风、安全意识、团队协作和遵守考场纪律等表现情况，由裁判按照“‘云计算技术与应用’职业素养评分标准”现场判分。