2023 年全国职业院校技能大赛（高职组）

“云计算应用”

上午场：私有云、容器云 09:00~12:30

项目需求：某企业根据自身业务需求，实施数字化转型，规划和建设数字化平台建设，平台聚焦“DevOps 建运一体”和“数据驱动产品开发”，拟采用开源 OpenStack 搭建企业内部私有云平台，开源 Kubernetes 搭建云原生服务平台，选择国内主流公有云平台服务，基于数字化平台底座，面向业务开发云应用产品。

拟将该任务交给工程师 A 与 B，分工协助完成云平台服务部署、云应用开发、云系统运维等任务，系统架构如下：

系统架构如图 1 所示，IP 地址规划如表 1 所示。

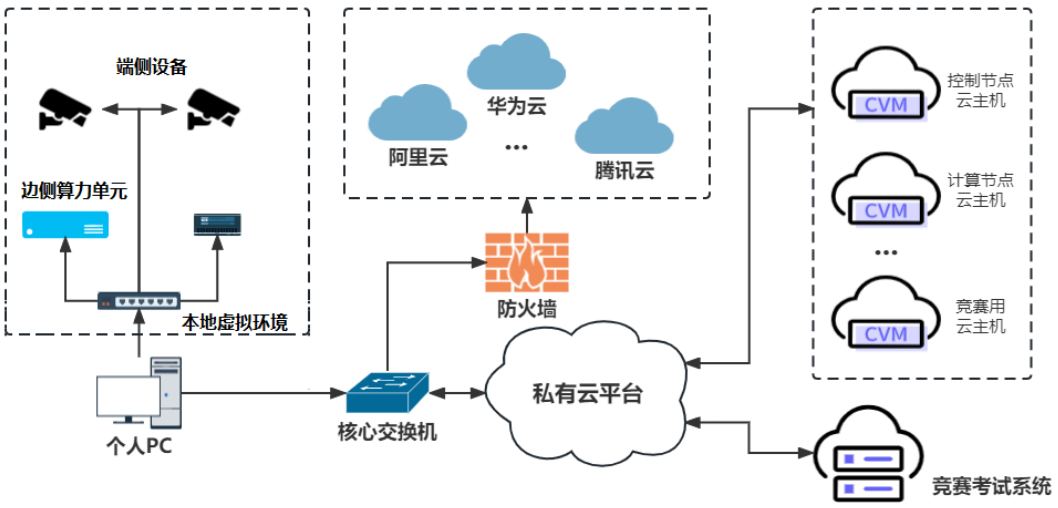

图 1 系统架构图

表 1 IP 地址规划

设备名称	主机名	接 口	IP 地址	说明
云服务器 1	controller	eth0	192.168.x.0/24	vlan x
		eth1	自定义	自行创建
云服务器 2	compute	eth0	192.168.x.0/24	vlan x
		eth1	自定义	自行创建

云服务器 3 ... 云服务器 n	自定义	eth0	192.168.x.0/24	
PC-1		本地连接	172.16.232.0/24	PC 使用

说明：

1.竞赛使用集群模式进行，比赛时给每个参赛队提供独立的租户与用户，各用户的资源配额相同，选手通过用户名与密码登录竞赛用私有云平台，创建云主机进行相应答题；

2.表中的 x 为工位号；在进行 OpenStack 搭建时的第二块网卡地址根据题意自行创建；

3.根据图表给出的信息，检查硬件连线及网络设备配置，确保网络连接正常；

4.考试所需要的账号资源、竞赛资源包与附件均会在考位信息表与设备确认单中给出；

5.竞赛过程中，为确保服务器的安全，请自行修改服务器密码；

模块一 私有云（30 分）

企业首先完成私有云平台搭建和运维，私有云平台提供云主机、云网络、云存储等基础架构云服务，并开发自动化运维程序。

任务 1 私有云服务搭建[5 分]

1.1.1 基础环境配置[0.2 分]

1.控制节点主机名为 controller，设置计算节点主机名为 compute；

2.hosts 文件将 IP 地址映射为主机名。

使用提供的用户名密码，登录提供的 OpenStack 私有云平台，在当前租户下，使用 CentOS7.9 镜像，创建两台云主机，云主机类型使用 4vCPU/12G/100G_50G 类型。当前租户下默认存在一张网卡，自行创建第二张网卡并连接至 controller 和 compute 节点（第二张网卡的网段为 10.10.X.0/24，X 为工位号，不需要创建路由）。自行检查安全组策略，以确保网络正常通信与 ssh 连接，然后按以下要求配置服务器：

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

（2）修改 hosts 文件将 IP 地址映射为主机名；

1.查看控制节点名字为 controller，查看 hosts 文件中有正确的主机名和 IP 映射计 0.1 分 2.控制节点正确使用两块网卡计 0.1 分

1.1.2 Yum 源配置[0.2 分]

使用提供的 http 服务地址，分别设置 controller 节点和 compute 节点的 yum 源文件 http.repo。

使用提供的 http 服务地址，在 http 服务下，存在 centos7.9 和 iaas 的网络 yum 源，使用该 http 源作为安装 iaas 平台的网络源。分别设置 controller 节点和 compute 节点的 yum 源文件 http.repo。

1.查看/etc/yum.repos.d/http.repo 文件，有正确的 baseurl 路径，计 0.2 分

1.1.3 配置无秘钥 ssh[0.2 分]

配置 controller 节点可以无秘钥访问 compute 节点。

配置 controller 节点可以无秘钥访问 compute 节点，配置完成后，尝试 ssh 连接 compute 节点的 hostname 进行测试。

1.查看控制节点允许计算节点无秘钥登录计 0.2 分

1.1.4 基础安装[0.2 分]

在控制节点和计算节点上分别安装 openstack-iaas 软件包。

在控制节点和计算节点上分别安装 openstack-iaas 软件包，根据表 2 配置两个节点脚本文件中的基本变量（配置脚本文件为/etc/openstack/openrc.sh）。

表 2 云平台配置信息

服务名称	变量	参数/密码
Mysql	root	000000
	Keystone	000000
	Glance	000000
	Nova	000000
	Neutron	000000
	Heat	000000
	Zun	000000
Keystone	DOMAIN_NAME	demo
	Admin	000000
	Rabbit	000000
	Glance	000000
	Nova	000000
	Neutron	000000
	Heat	000000
	Zun	000000
Neutron	Metadata	000000
	External Network	eth1（以实际为准）

1.检查环境变量文件配置正确计 0.2 分

1.1.5 数据库安装与调优[0.5 分]

在控制节点上使用安装 Mariadb、RabbitMQ 等服务。并进行相关操作。

在 controller 节点上使用 iaas-install-mysql.sh 脚本安装 Mariadb、Memcached、RabbitMQ 等服务。安装服务完毕后，修改/etc/my.cnf 文件，完成下列要求：

1.设置数据库支持大小写；

2.设置数据库缓存 innodb 表的索引，数据，插入数据时的缓冲为 4G；

3.设置数据库的 log buffer 为 64MB；

4.设置数据库的 redo log 大小为 256MB；

5.设置数据库的 redo log 文件组为 2。

6.修改 Memcached 的相关配置，将内存占用大小设置为 512MB，调整最大连接数参数为 2048；

7.调整 Memcached 的数据摘要算法（hash）为 md5；

1.检查数据库和 memcached 配置正确计 0.5 分

1.1.6 Keystone 服务安装与使用[0.5 分]

在控制节点上安装 Keystone 服务并创建用户。

在 controller 节点上使用 iaas-install-keystone.sh 脚本安装 Keystone 服务。

然后创建 OpenStack 域 210Demo，其中包含 Engineering 与 Production 项目，在域 210Demo 中创建组 Devops，其中需包含以下用户：

1.Robert 用户是 Engineering 项目的用户（member）与管理员（admin），email 地址为：Robert@lab.example.com。

2.George 用户是 Engineering 项目的用户（member），email 地址为：George@lab.example.com。

3.William 用户是 Production 项目的用户（member）与管理员（admin），email 地址为：William@lab.example.com。

4.John 用户是 Production 项目的用户（member），email 地址为：John@lab.example.com。

1.检查平台中的 210Demo 域中是否有题目所需的用户与项目，正确计 0.5 分

1.1.7 Glance 安装与使用[0.5 分]

在控制节点上安装 Glance 服务。上传镜像至平台，并设置镜像启动的要求参数。

在 controller 节点上使用 iaas-install-glance.sh 脚本安装 glance 服务。然后使用提供的 coreos_production_pxe.vmlinuz 镜像（该镜像为 Ironic Deploy 镜像，是一个 AWS 内核格式的镜像，在 OpenStack Ironic 裸金属服务时需要用到）上

传到 OpenStack 平台中，命名为 deploy-vmlinuz。

1.检查 glance 服务安装正确计 0.1 分
2.检查 deploy-vmlinuz 镜像内核格式正确计 0.4 分

1.1.8 Nova 安装与优化[0.5 分]

在控制节点和计算节点上分别安装 Nova 服务。安装完成后，完成 Nova 相关配置。

在 controller 节点和 compute 节点上分别使用 iaas-install-placement.sh 脚本、iaas-install-nova -controller.sh 脚本、iaas-install-nova-compute.sh 脚本安装 Nova 服务。在 OpenStack 中，修改相关配置文件，修改调度器规则采用缓存调度器，缓存主机信息，提升调度时间。

1.检查 nova 服务调度器配置正确计 0.5 分

1.1.9 Neutron 安装[0.2 分]

在控制和计算节点上正确安装 Neutron 服务。

使用提供的脚本 iaas-install-neutron-controller.sh 和 iaas-install-neutron-compute.sh，在 controller 和 compute 节点上安装 neutron 服务。

1.检查 neutron 服务安装正确计 0.1 分
2.检查 neutron 服务的 linuxbridge 网桥服务启动正确计 0.1 分

1.1.10 Dashboard 安装[0.5 分]

在控制节点上安装 Dashboard 服务。安装完成后，将 Dashboard 中的 Django 数据修改为存储在文件中。

在 controller 节点上使用 iaas-install-dashboad.sh 脚本安装 D**a**shboard 服务。安装完成后，修改相关配置文件，完成下列两个操作：

1.使得登录 Dashboard 平台的时候不需要输入域名；

2.将 Dashboard 中的 Django 数据修改为存储在文件中。

1.检查 Dashboard 服务中 Djingo 数据修改为存储在文件中配置正确计 0.2 分 2.检查 Dashboard 服务中登陆平台不输入域名配置正确计 0.3 分

1.1.11 Swift 安装[0.5 分]

在控制节点和计算节点上分别安装 Swift 服务。安装完成后，将 cirros 镜像进行分片存储。

在控制节点和计算节点上分别使用 iaas-install-swift-controller.sh 和 iaas-install-swift-compute.sh 脚本安装 Swift 服务。安装完成后，使用命令创建一个名叫 examcontainer 的容器，将 cirros-0.3.4-x86_64-disk.img 镜像上传到 examcontainer 容器中，并设置分段存放，每一段大小为 10M。

1.检查 swift 服务安装正确计 0.3 分 2.分段上传 cirros 镜像正确计 0.2 分

1.1.12 Cinder 创建硬盘[0.5 分]

在控制节点和计算节点分别安装 Cinder 服务，请在计算节点，对块存储进行扩容操作。

在控制节点和计算节点分别使用 iaas-install-cinder-controller.sh、iaas-install-cinder-compute.sh 脚本安装 Cinder 服务，请在计算节点，对块存储进行扩容操作，即在计算节点再分出一个 5G 的分区，加入到 cinder 块存储的后端存储中去。

1.检查 cinder 后端存储扩容成功计 0.5 分

1.1.13 配置主机禁 ping [0.5 分]

修改 controller 节点的相关配置文件，配置 controller 节点禁止其他节点可以 ping 它。

1.检查系统配置文件正确计 0.5 分

任务 2 私有云服务运维[15 分]

任务 2 私有云服务运维（15 分）

1.2.1 Heat 编排-创建用户[1 分]

编写 Heat 模板 create_user.yaml，创建名为 heat-user 的用户。

使用自己搭建的 OpenStack 私有云平台，使用 heat 编写摸板(heat_template_version: 2016-04-08)创建名为”chinaskills”的 domain，在此 domain 下创建名为 beijing_group 的租户，在此租户下创建名为 cloud 的用户，将此文件命名及保存在/root/user_create.yml。（竞赛系统会执行 yaml 文件，请确保执行的环境）

1.执行 heat 模板成功创建出域和租户以及用户正确计 1 分

1.2.2 KVM 优化[1 分]

在 OpenStack 平台上修改相关配置文件，启用-device virtio-net-pci in kvm。

在自行搭建的 OpenStack 私有云平台或赛项提供的 all-in-one 平台上，修改相关配置文件，启用-device virtio-net-pci in kvm。

1.检查 nova 服务 KVM 配置正确计 1 分

1.2.3 NFS 对接 Glance 后端存储[1 分]

使用 OpenStack 私有云平台，创建一台云主机，安装 NFS 服务，然后对接 Glance 后端存储。

使用赛项提供的 OpenStack 私有云平台，创建一台云主机（镜像使用 CentOS7.9，flavor 使用带临时磁盘 50G 的），配置该主机为 nfs 的 server 端，将该云主机中的 50G 磁盘通过/mnt/test 目录进行共享（目录不存在可自行创

建）。然后配置 controller 节点为 nfs 的 client 端，要求将/mnt/test 目录作为 glance 后端存储的挂载目录。

1.检查挂载共享目录正确计 0.5 分 2.检查共享目录用户合组正确计 0.5 分

1.2.4 Redis 主从[1 分]

使用赛项提供的 OpenStack 私有云平台，创建两台云主机，配置为 redis 的主从架构。

使用赛项提供的 OpenStack 私有云平台，申请两台 CentOS7.9 系统的云主机，使用提供的 http 源，在两个节点安装 redis 服务并启动，配置 redis 的访问需要密码，密码设置为 123456。然后将这两个 redis 节点配置为 redis 的主从架构。

1.检查 redis 主从集群部署正确计 1 分

1.2.6 Glance 调优[1 分]

在 OpenStack 平台中，修改相关配置文件，将子进程数量相应的配置修改成 2。

在 OpenStack 平台中，glance-api 处理请求的子进程数量默认是 0，只有一个主进程，请修改相关配置文件，将子进程数量相应的配置修改成 2，这样的话有一个主进程加 2 个子进程来并发处理请求。

1.检查 glance 服务进程配置正确计 1 分

1.2.7 Ceph 部署[1 分]

使用提供的 ceph.tar.gz 软件包，安装 ceph 服务并完成初始化操作。

使用提供的 ceph-14.2.22.tar.gz 软件包，在 OpenStack 平台上创建三台 CentOS7.9 系统的云主机，使用这三个节点安装 ceph 服务并完成初始化操作，第一个节点为 mon/osd 节点，第二、三个节点为 osd 节点，部署完 ceph 后，创建 vms、images、volumes 三个 pool。

1.检查 ceph 集群状态为 OK 正确计 0.5 分 2.检查 ceph 集群的 osd 正确计 0.5 分

1.2.8 Glance 对接 Ceph 存储[1 分]

修改 OpenStack 平台中 Glance 服务的配置文件，将 Glance 后端存储改为 Ceph 存储。

在自己搭建的 OpenStack 平台中修改 glance 服务的相关配置文件，将 glance 后端存储改为 ceph 存储。也就是所以的镜像会上传至 ceph 的 images pool 中。通过命令使用 cirros-0.3.4-x86_64-disk.img 镜像文件上传至云平台中，镜像命名为 cirros。

1.检查 glance 服务存储配置正确计 1 分

1.2.9 Cinder 对接 Ceph 存储[1 分]

修改 OpenStack 平台中 cinder 服务的配置文件，将 cinder 后端存储改为 Ceph 存储。

修改 OpenStack 平台中 cinder 服务的相关配置文件，将 cinder 后端存储改为 ceph 存储。使创建的 cinder 块存储卷在 ceph 的 volumes pool 中。配置完成后，在 OpenStack 平台中使用创建卷 cinder-volume1，大小为 1G。

1.检查 cinder 服务存储配置正确计 1 分

1.2.10 Nova 对接 Ceph 存储[1 分]

修改 OpenStack 平台中 Nova 服务的配置文件，将 Nova 后端存储改为 Ceph 存储。

修改 OpenStack 平台中 nova 服务的相关配置文件，将 nova 后端存储改为 ceph 存储。使 nova 所创建的虚拟机都存储在 ceph 的 vms pool 中。配置完成后在 OpenStack 平台中使用命令创建云主机 server1。

1.检查 nova 服务存储配置正确计 0.5 分 2.检查云主机在 vms pool 中正确计 0.5 分

1.2.11 私有云平台优化：系统网络优化[1 分]

使用自行创建的 OpenStack 云计算平台，配置控制节点的网络缓存，使得每个 UDP 连接(发送、接受)保证能有最小 358 KiB 最大 460 KiB 的 buffer，重启后仍然生效。

1.检查网络缓存的系统配置正确计 1 分

1.2.12 私有云平台排错：Neutron 网络组件排错[2 分]

使用赛项提供的 chinaskill-error01 镜像创建一台云主机（云主机的登录用户名为 root，密码为 Abc@1234）,该云主机中存在错误的网络服务，错误现象为 dashboard 不能查看网络相关的页面，请修复 neutron 服务。

1.检查数据库中网络服务不存在冲突计 2 分

1.2.13 私有云平台排错：Cinder 卷组件报错[2 分]

使用赛项提供的 chinaskill-error01 镜像创建一台云主机（云主机的登录用户名为 root，密码为 Abc@1234）,该云主机中存在错误的卷服务，错误现象为无法正常使用 cinder 命令，请修复卷服务，然后使用命令创建名为 chinaskill-

cinder 的卷，大小为 2G。

1.检查卷组件命令可以正常使用计 2 分

任务 3 私有云运维开发[10 分]

1.3.1 编写 Shell 一键部署脚本[1.0 分]

编写一键部署 nfs 云网盘应用系统。

在 OpenStack 私有云平台，创建一台云主机，使用提供的软件包，在 root 目录下编写一键部署脚本 install_owncloud.sh，要求可以一键部署 owncloud 云网盘应用系统。

1.检查 owncloud 服务正常启动计 1 分

1.3.2 Ansible 服务部署：部署 ELK 集群服务[2 分]

编写 Playbook，部署的 ELK。

使用赛项提供的 OpenStack 私有云平台，创建三台 CentOS7.9 系统的云主机分别命名为 elk-1、elk-2 和 elk-3，Ansible 主机可以使用上一题的环境。要求 Ansible 节点编写剧本，执行 Ansible 剧本可以在这三个节点部署 ELK 集群服务（在/root 目录下创建 install_elk 目录作为 ansible 工作目录，部署的入口文件命名为 install_elk.yaml）。具体要求为三个节点均安装 Elasticserach 服务并配置为 Elasticserach 集群；kibana 安装在第一个节点；Logstash 安装在第二个节点。

1.执行 yaml 文件正确计 0.5 分 2.检查 ELK 服务部署正确计 1.5 分

1.3.3 Ansible 部署 Kafka 服务[2.0 分]

编写 Playbook，部署的 ZooKeeper 和 Kafka。

使用提供的 OpenStack 私有云平台，创建 4 台系统为 centos7.5 的云主机，其中一台作为 Ansible 的母机并命名为 ansible，另外三台云主机命名为 node1、node2、node3，通过附件中的/ansible/ansible.tar.gz 软件包在 ansible 节点安装 Ansible 服务；使用这一台母机，编写 Ansible 脚本（在/root 目录下创建 example 目录作为 Ansible 工作目录，部署的入口文件命名为 cscc_install.yaml），编写 Ansible 脚本使用 roles 的方式对其他三台云主机进行安装 kafka 集群的操作（zookeeper 和 kafka 的安装压缩包在 gpmall-single.tar.gz 压缩包中，将 zookeeper 和 kafka 的压缩包解压到 node 节点的/opt 目录下进行安装）。

1.执行 yaml 文件正确计 0.5 分
2.检查 Kafka 和 Zookeeper 服务部署正确计 1.5 分

1.3.4 OpenStack 资源管理的客户端程序开发 [2 分]

使用已建好的 OpenStack Python 运维开发环境，在 root 目录下创建 resource_manager.py 脚本，基于 OpenStack 资源配额管理服务封装客户端工具，resource_manager.py 程序支持命令行带参数执行，命令参数要求说明如下：

（1）位置参数“command”，表示操作类型。操作类型包括“list”：标识查看所有的对象；“get”：查询指定的对象。

（2）位置参数“resource”，表示资源信息类型：类型包括 “provider”：资源提供者；“inventory”：资源库存；“usage”：资源使用情况。

（3）参数“-p 或-- provider”，标识资源提供者的名称。

功能要求如下：

（1）程序查询所有资源提供者，以 json 格式控制台输出。

执行实例如下： python3 resource_manager.py list provider

（2）查询指定资源提供者的资源信息，以 json 格式控制台输出。

执行实例如下： python3 resource_manager.py get provider -p “providername”

（3）查询指定资源提供者的资源库存信息，以 json 格式控制台输出。

执行实例如下： python3 resource_manager.py get inventory -p

“providername”

（4）查询指定资源提供者的资源使用信息，以 json 格式控制台输出。

执行实例如下： python3 resource_manager.py get usage -p “providername”

1. 实现查询所有资源提供者的功能开发，计 0.5 分；
2. 实现查询指定资源提供者的资源信息的功能开发，计 0.5 分；
3. 实现查询指定资源提供者的资源库存信息的功能开发，计 0.5 分；
4. 实现查询指定资源提供者的资源使用信息的功能开发，计 0.5 分。

1.3.5 OpenStack 数据库操作开发[3 分]

使用已建好的 OpenStack Python 运维开发环境，在/root 目录下创建 mysql_db_manager.py 脚本，基于 python 的 sqlalchemy 库进行数据库的管理，实现对 OpenStack 内指定服务的 MariaDB 数据库管理功能，实现该数据库下表创建、删除、数据插入、更新等操作（为确保检测成功，OpenStack MariaDB 数据库的账户 root、密码 000000 不要修改）：

该表(table)用于描述 OpenStack 成员信息，表结构如下：

表名称"member",该表有 4 个字段：

id INTEGER 主键，自动递增；

name VARCHAR(256);

level INTEGER；

place varchar(256);

（1）mysql_db_manager 内部实现 DM_Manager 类，类初始化方法 def __init__(self, db_name)，参数 db_name 是 OpenStack 内部指定的某一服务数据库名称。

（2）DM_Manager 类实现 create_table(self)方法。实现创建表功能，表指 member 表。

（3）DM_Manager 类实现 insert_table_data(self,**kwargs)方法。实现表中表数据插入功能，参数 kwargs 参数为要插入数据记录信息，key 与表字段名称一致。

（4）DM_Manager 类实现 update_table_data(self,**kwargs)方法。实现表中数据记录更新功能，参数 kwargs 参数为要更新数据记录信息，key 与表字段名

称一致。

（5）DM_Manager 类实现 run_table_raw_sql(self, raw_sql:str): 方法，无参数。实现表中直接执行 sql 语句功能，raw_sql 参数为 sql 语句。

（6）DM_Manager 类实现 delete_table(self)方法。实现删除表功能，表指 member 表。

1. 完成 OpenStack 内部指定数据库链接功能，计 0.5 分；
2. 实现 OpenStack 指定数据库中创建表功能，计 0.5 分；
3. 实现指定数据库表插入数据记录功能，计 0.5 分；
4. 实现指定数据库表数据更新功能，计 0.5 分。
5. 实现通过 sql 直接操作执行数据库功能，计 0.5 分。
6. 实现删除指定数据库表功能，计 0.5 分。

模块二 容器云（30 分）

企业构建 Kubernetes 容器云集群，引入 KubeVirt 实现 OpenStack 到 Kubernetes 的全面转型，用 Kubernetes 来管一切虚拟化运行时，包含裸金属、VM、容器。同时研发团队决定搭建基于 Kubernetes 的 CI/CD 环境，基于这个平台来实现 DevOps 流程。引入服务网格 Istio，实现业务系统的灰度发布，治理和优化公司各种微服务，并开发自动化运维程序。

任务 1 容器云服务搭建[5 分]

2.1.1 部署容器云平台[5 分]

使用 OpenStack 私有云平台创建两台云主机，云主机类型使用 4vCPU/12G/100G 类型，分别作为 Kubernetes 集群的 Master 节点和 node 节点，然后完成 Kubernetes 集群的部署，并完成 Istio 服务网格、KubeVirt 虚拟化和 Harbor 镜像仓库的部署。

1.Kubernetes 集群部署成功得 5 分；

任务 2 容器云服务运维[15 分]

2.2.1 容器化部署 MariaDB [0.5 分]

编写 Dockerfile 文件构建 mysql 镜像，要求基于 centos 完成 MariaDB 数据库的安装和配置，并设置服务开机自启。

编写 Dockerfile 构建镜像 erp-mysql:v1.0，要求使用 centos7.9.2009 镜像作为基础镜像，完成 MariaDB 数据库的安装，设置 root 用户的密码为 tshoperp，新建数据库 jsh_erp 并导入数据库文件 jsh_erp.sql，并设置 MariaDB 数据库开机自启。

1.镜像构建成功得 0.1 分； 2.数据库安装且导入数据成功得 0.4 分。

2.2.2 容器化部署 Redis [0.5 分]

编写 Dockerfile 文件构建 redis 镜像，要求基于 centos 完成 Redis 服务的安装和配置，并设置服务开机自启。

编写 Dockerfile 构建镜像 erp-redis:v1.0，要求使用 centos7.9.2009 镜像作为基础镜像，完成 Redis 服务的安装，修改其配置文件关闭保护模式，并设置 Redis 服务开机自启。

1.镜像构建成功的 0.1 分； 2.Redis 服务安装成功且配置正确得 0.4 分。

2.2.3 容器化部署 Nginx [0.5 分]

编写 Dockerfile 文件构建 nginx 镜像，要求基于 centos 完成 Nginx 服务的安装和配置，并设置服务开机自启。

编写 Dockerfile 构建镜像 erp-nginx:v1.0，要求使用 centos7.9.2009 镜像作为基础镜像，完成 Nginx 服务的安装，使用提供的 app.tar.gz 和 nginx.conf 启动 Nginx 服务，并设置开机自启。

1.镜像构建成功得 0.3 分； 2.Nginx 安装成功且配置正确得 0.2 分。

2.2.4 容器化部署 ERP[0.5 分]

编写 Dockerfile 文件构建 erp 镜像，要求基于 centos 完成 JDK 环境和 ERP 服务的安装与配置，并设置服务开机自启。

编写 Dockerfile 构建镜像 erp-server:v1.0，要求使用 centos7.9.2009 镜像作为基础镜像，完成 JDK 环境的安装，启动提供的 jar 包，并设置服务开机自启。

1.镜像构建成功得 0.3 分； 2.erp 环境安装正确得 0.2 分。

2.2.5 编排部署 ERP 管理系统[1 分]

编写 docker-compose.yaml 文件，要求使用镜像 mysql、redis、nginx 和 erp 完成 ERP 管理系统的编排部署。

编写 docker-compose.yaml 完成 ERP 管理系统的部署，要求定义 mysql、redis、nginx 和 erp 共四个 Service，分别使用镜像 erp-redis:v1.0、erp-mysql:v1.0、erp-nginx:v1.0 和 erp-server:v1.0，并将 nginx 服务的 80 端口映射到宿主机的 8888 端口。

1.8888 端口可以访问到 ERP 系统得 1 分

2.2.6 部署 GitLab [1 分]

将 GitLab 部署到 Kubernetes 集群中，设置 GitLab 服务 root 用户的密码，使用 Service 暴露服务，并将提供的项目包导入到 GitLab 中。

在 Kubernetes 集群中新建命名空间 gitlab-ci，将 GitLab 部署到该命名空间下，Deployment 和 Service 名称均为 gitlab，以 NodePort 方式将 80 端口对外暴露为 30880，设置 GitLab 服务 root 用户的密码为 admin@123，将项目包 demo-2048.tar.gz 导入到 GitLab 中并命名为 demo-2048。

1.GitLab 部署正确且能正常访问得 0.5 分； 2.项目导入成功得 0.5 分。

2.2.7 部署 GitLab Runner [1 分]

将 GitLab Runner 部署到 Kubernetes 集群中，为 GitLab Runner 创建持久化构建缓存目录以加速构建速度，并将其注册到 GitLab 中。

将 GitLab Runner 部署到 gitlab-ci 命名空间下，Release 名称为 gitlab-runner，为 GitLab Runner 创建持久化构建缓存目录/home/gitlab-runner/ci-build-cache 以加速构建速度，并将其注册到 GitLab 中。

1.GitLab Runner 部署成功得 0.2 分； 2.GitLab Runner 注册成功得 0.3 分； 3.GitLab Runner 持久化配置成功得 0.5 分。

2.2.9 构建 CI/CD [2 分]

编写流水线脚本触发自动构建，要求基于 GitLab 项目完成代码的编译、镜像的构建与推送，并自动发布应用到 Kubernetes 集群中。

编写流水线脚本.gitlab-ci.yml 触发自动构建，具体要求如下：

（1）基于镜像 maven:3.6-jdk-8 构建项目的 drone 分支；

（2）构建镜像的名称：demo:latest；

（3）将镜像推送到 Harbor 仓库 demo 项目中；

（4）将 demo-2048 应用自动发布到 Kubernetes 集群 gitlab-ci 命名空间下。

1.项目编译成功得 0.5 分； 2.镜像构建成功得 0.5 分； 3.服务发布成功得 0.5 分； 4.服务能正常访问得 0.5 分。

2.2.10 **服务网格：创建 VirtualService [1 分]**

将 Bookinfo 应用部署到 default 命名空间下，为 Bookinfo 应用创建一个名为 reviews 的 VirtualService，要求来自名为 Jason 的用户的所有流量将被路由到 reviews 服务的 v2 版本。

1.流量正确走向 Jason 用户得 1 分。

2.2.11 **KubeVirt 运维：创建 VMI [1 分]**

使用提供的镜像在 default 命名空间下创建一台 VMI，名称为 exam，指定 VMI 的内存、CPU、网卡和磁盘等配置，并开启 Sidecar 注入。

将提供的镜像 cirros-0.5.2-x86_64-disk.img 转换为 docker 镜像 cirros:v1.0，然后使用镜像 cirros:v1.0 镜像在 default 命名空间下创建一台 vmi，名称为 vmi-

cirros，内存为 1024M。

1.注入成功得 0.5 分。 2.vmi 创建成功并且可正常访问的 0.5 分。

2.2.12 容器云平台优化：使用审计日志[2 分]

启用审计日志可以帮助集群管理员快速的定位问题。请启用 Kubernetes 集群的审计日志，要求以 JSON 格式保存审计日志并输出到/etc/kubernetes/audit-logs/audit.log 文件中，日志最多保留 10 天，单个日志文件不超过 500M。然后编写审计策略文件/etc/kubernetes/audit-policy/policy.yaml，要求在日志中用 RequestResponse 级别记录 Pod 变化，并记录 kube-system 中 ConfigMap 变更的请求消息体。

1.审计日志开启成功得 1 分。 2.日志成功输出得 1 分。

2.2.13 容器云平台排错：容器引擎报错[1.5 分]

使用赛项提供的 chinaskill-k8s-error01 镜像创建一台云主机（云主机的登录用户名为 root，密码为 Abc@1234）,该云主机中存在错误的容器服务，错误现象为无法正常使用 crictl ps -l 命令，请修复容器服务。

1.containerd 服务正常启动得 1.5 分。

2.2.14 容器云平台排错：DNS 服务报错[1.5 分]

使用赛项提供的 chinaskill-k8s-error01 镜像创建一台云主机（云主机的登录用户名为 root，密码为 Abc@1234）,该云主机中存在错误的 DNS 服务，错误现象为 coredns 一直处于 CrashLoopBackOff 状态，请修复 CoreDNS 服务，并将并

发上限更改为 5000。

1.CoreDNS 服务为 Running 得 1 分。 2.并发上限配置正确得 0.5 分。

任务 3 容器云运维开发[10 分]

2.3.1 管理 service 资源[2 分]

Kubernetes Python 运维脚本开发，使用 Restful APIs 方式管理 service 服务。

使用已经部署完成的 Kubernetes 两节点云平台，在 Master 节点安装 Python 3.7.3 的运行环境与依赖库。

使用 python request 库和 Kubernetes Restful APIs，在/root 目录下，创建 api_manager_service.py 文件，要求编写 python 代码，代码实现以下任务：

（1）首先查询查询服务 service，如果 service 名称“nginx-svc”已经存在，先删除。

（2）如果不存在“nginx-svc”，则使用 service.yaml 文件创建服务。

（3）创建完成后，查询该服务的信息，查询的 body 部分以 json 格式的文件输出到当前目录下的 service_api_dev.json 文件中。

（4）然后使用 service_update.yaml 更新服务端口。

（5）完成更新后，查询该服务的信息，信息通过控制台输出，并通过 json 格式追加到 service_api_dev.json 文件后。

1.执行 api_manager_service.py 文件，成功创建 Service 资源得 1 分 2.查询 Service 资源，配置正确得 1 分。

2.3.2 管理 Pod 服务[3 分]

Kubernetes Python 运维脚本开发-使用 SDK 方式，通过 Deployment 管理 Pod

服务。

使用已经部署完成的 Kubernetes 两节点云平台，在 Master 节点安装 Python 3.7.3 的运行环境与依赖库。

使用 Kubernetes python SDK 的“kubernetes”Python 库，在/root 目录下，创建 sdk_manager_deployment.py 文件，要求编写 python 代码，代码实现以下任务：

（1）首先使用 nginx-deployment.yaml 文件创建 deployment 资源。

（2）创建完成后，查询该服务的信息，查询的 body 部分通过控制台输出，并以 json 格式的文件输出到当前目录下的 deployment_sdk_dev.json 文件中。

1.执行 sdk_manager_deployment.py 脚本,成功创建 deployment 资源计 1.5 分
2.检查创建的 deployment 资源，配置信息无误计 1.5 分

2.3.3 Kubernetes CRD 自定义资源的管理封装[3 分]

Kubernetes 容器云平台通过 CRD 机制进行自定义 APIs 资源拓展，将 chinaskill-cloud-*.yaml 共 5 个文件复制到 root 目录下。参考 chinaskill-cloud-11.yaml 文件，编写 CRD 文件“chinaskill-cloud-crd.yaml”，放在 root 目录下。

说明：Competition CRD 命名要求如下：

Kind 为 Competition

Plural 为 competitions

singular 为 competition

shortNames 为 cpt

session 含义是赛程

content 含义为竞赛内容。

使用已建好的 Kubernetes Python 运维开发环境，在/root 目录下创建 crd _manager.py 脚本。crd_manager.py 编写基于 Kubernetes SDK 实现 Competition CRD 的创建、删除与事件变化监听。

crd_manager.py 内部实现 3 个方法：

crd_manager.py 内部实现 3 个方法：

（1）实现方法 create_crd(),实现对 Competition CRD 的创建。

（2）实现方法 delete_crd()，实现对 Competition CRD 的删除。

（3）实现方法 watch_crd_object(),实现对 CRD 资源的变化事件监听，将监听到 Competition CRD 被删除，将 event 信息输出到控制台，并停止监听。

1. 基于 Kubernetes Python SDK，实现 CRD 自定义资源创建功能，计 1 分； 2. 基于 Kubernetes Python SDK，实现 CRD 自定义资源删除功能，计 1 分； 3. 基于 Kubernetes Python SDK，实现 CRD 自定义资源 watch 与 event 监听功能，遍历 CRD 删除 event 并输出，计 1 分；

2.3.4 Kubernetes CRD 的自定义对象管理封装[2 分]

基于前一题目的 Competition CRD，使用已建好的 Kubernetes Python 运维开发环境，在/root 目录下创建 crd_object_manager.py 脚本。

crd_object_manager.py 编写基于 Kubernetes SDK 实现 Competition CRD 的自定义对象的创建、删除与事件变化监听。

crd_manager.py 内部实现 3 个方法：

（1）实现方法 create_crd_object(ymlfile:str),实现 CRD 的自定义对象的创建，yamlfile 为 CRD 的自定义对象 yaml 文件。

（2）实现方法 delete_crd_object(crd_object_name:str),实现创建 CRD 功能，crd_object_name 为 CRD 的自定义对象名称。

（3）实现方法 watch_crd_object(crd_object_name:str),实现 CRD 的自定义对象变化事件监听，当监听到 CRD 的自定义对象的名称为 crd_object_name 被删除时，将 event 信息输出到控制台，并停止监听。

1. 基于 Kubernetes Python SDK，实现 CRD 的自定义对象创建功能，计 0.5 分； 2. 基于 Kubernetes Python SDK，实现 CRD 的自定义对象删除功能，计 0.5 分； 3. 基于 Kubernetes Python SDK，实现 CRD 的自定义对象 watch 与 event 监听功能，遍历 CRD 的自定义对象删除 event 并输出，计 1 分；

下午场：公有云 13:30~16:00

模块三 公有云

企业选择国内公有云提供商，选择云主机、云网络、云硬盘、云防火墙、负载均衡等服务，可创建 Web 服务，共享文件存储服务，数据库服务，数据库集群等服务。搭建基于云原生的 DevOps 相关服务，构建云、边、端一体化的边缘计算系统，并开发云应用程序。

根据上述公有云平台的特性，完成公有云中的各项运维工作。

任务 1 公有云服务搭建[5 分]

3.1.1 私有网络管理[0.3 分]

在公有云中完成虚拟私有云的创建。

具体要求如下：

（1）在上海一区域进行创建操作；

（2）创建一个名为 intnetX 的内部网络：IP 地址为 172.16.0.0/16；

（3）创建子网名称为 intnetX-server：IP 地址为 172.16.1.0/24;

（4）创建子网名称为 intnetX-mysql：IP 地址为 172.16.2.0/24;

1.查看 vpc 名字为 intnetX 正确计 0.1 分
2.查看子网名为 intnetX-server 和子网网段正确计 0.1 分
3.查看子网名为 intnetX-mysql 和子网网段正确计 0.1 分

3.1.2 云实例管理[0.5 分]

登录华为云平台，创建两台云实例。

具体要求如下：

（1）计费模式：按需计费；

（2）地域：上海一；

（3）CPU 架构：x86 计算；

（4）规格：c7.xlarge.2；

（5）镜像：CentOS 7.5 64 位；

（6）系统盘：高 IO 50G 硬盘；

（7）公网带宽：按带宽计费，5Mbps；

（8）实例名称：ChinaSkill-node-1、ChinaSkill-node-2；

（9）登录方式：使用密码登录，密码自定义。

1.查看云实例使用的类型是否正确计 0.2 分 2.查看云实例使用的云硬盘是否正确计 0.3 分

3.1.3 数据库管理[0.2 分]

使用 intnetX-mysql 网络创建三台云服务器 chinaskill-sql-1、chinaskill-sql-2 和 chinaskill-sql-3（系统使用 CentOS7.9），使用提供的压缩文件 mongodb-repo.tar.gz 中的软件包源，在这三台云服务器上部署 MongoDB 数据库服务。

1.查看数据库安装正确计 0.2 分

3.1.4 主从数据库[0.5 分]

在 chinaskill-sql-1、chinaskill-sql-2 和 chinaskill-sql-3 云服务器中配置 MongoDB 一主二从数据库+副本集操作，chinaskill-sql-1 节点为主节点，其余节点为从节点，配置 MongoDB 集群名称为 cloud。

1.查看 mongo 数据库 rs 集群状态正确计 0.5 分

3.1.5 node 环境管理[0.5 分]

使用提供的压缩文件，安装 Node.js 环境。

使用提供的压缩文件 rocketchat-cloud.tar.gz 中软件包源，在 ChinaSkill-node-1 部署 nodejs，根据所提供的 rocket.chat 所需要求安装 nodejs 对应版本。

1.查看 node 版本为 v12.16.1 正确计 0.5 分

3.1.6 安全组管理[0.5 分]

根据要求，创建一个安全组。

具体要求如下：

（1）名称：intnetX-security；

（2）允许策略：只放行源地址为 172.16.1.0/24 访问 27017 端口；

（3）允许策略：只放行源地址为 172.16.1.0/24 使用 ping 命令访问；

（4）关联实例：将 intnetX-security 安全组关联至所创建的数据库中；

1.查看安全组是否被创建计 0.1 分
2.查看安全组 tcp 策略是否按要求创建计 0.2 分
3.查看安全组 icmp 策略是否按要求创建计 0.2 分

3.1.7 RocketChat 上云[0.5 分]

使用 http 服务器提供文件，将 Rocket.Chat 应用部署上云。

使用 http 服务器提供的压缩文件 rocketchat-cloud.tar.gz 中的 RocketChat 软件包，在 ChinaSkill-node-1 中部署 RocketChat 的 Web 服务。使用 chinaskill-sql-1 的 MongoDB 为后端数据库，设置 RocketChat 服务访问端口为 3000。

1.查看 RocketChat 服务正常计 0.2 分
2.查看前端页面是否正常返回计 0.3 分

3.1.8 NAT 网关[0.5 分]

根据要求创建一个公网 NAT 网关。

具体配置如下：

（1）名称为：kcloud-nat；

（2）虚拟私有云：intnetX；

（3）子网：intnetX-server；

（4）规则：内部子网地址访问外网；

（5）设置 Chinaskill-node-1 云服务器 3000 端口转换为外部 3000 服务端口。

1.查看虚拟私有云中是否存在访问外网 NAT 策略计 0.5 分

3.1.9 云备份[0.5 分]

创建一个云服务器备份存储库名为 server_backup，容量为 100G。将 ChinaSkill-node-1 云服务器制作镜像文件 chinaskill-image。

1.查看云服务器备份是否正确计 0.5 分

3.1.10 负载均衡器[0.5 分]

根据要求创建一个负载均衡器 chinaskill-elb。

将 ChinaSkill-node-1 和 ChinaSkill-node-2 加入负载均衡的后端。设置一个可用的公网服务 IP，服务端口为 3000。配置监听器，监听 3000 端口。对浮动公共 IP 进行 Web 访问测试。

1.查看负载均衡器参数是否正确计 0.5 分

3.1.11 弹性伸缩管理[0.5 分]

根据要求新建一个弹性伸缩启动配置。

具体要求如下：

（1）启动配置名称：template-exam；

（2）计费模式：按量计费；

（3）地域：上海一；

（4）镜像：chinaskill-image；

（5）登录方式：使用密码登录，密码自定义。

创建一个伸缩组，具体要求如下：

（1）名称：as-exam；

（2）最小伸缩数：1；起始实例数：1；最大伸缩数：5；

（3）启动配置：template-exam；

（4）使用负载均衡：chinaskill-elb；

（5）移出策略：移出最旧的实例。

为伸缩组 as-exam 新建告警触发策略，具体要求如下：

（1）如果实例的内存利用率在 5 分钟内的最大值小于 40%，且连续发生 3 次。则实例数减少 1 台。冷却 60 秒；

（2）如果实例的内存利用率在 5 分钟内的最大值大于 80%，且连续发生 3 次。则实例数增加 1 台。冷却 60 秒；

（3）如果实例的 CPU 利用率在 5 分钟内的最大值小于 40%，且连续发生 3 次。则实例数减少 1 台。冷却 60 秒；

（4）如果实例的 CPU 利用率在 5 分钟内的平均值大于等于 80%，且连续发生 3 次。则实例数增加 1 台。冷却 60 秒。

1.查看弹性伸缩参数是否正确计 0.5 分

任务 2 公有云服务运维[10 分]

3.2.1 云容器引擎[1 分]

在公有云上，按照要求创建一个 x86 架构的容器云集群。

具体要求如下：

（1）集群名称：kcloud；

（2）集群版本：v1.25；

（3）地域：上海一；

（4）集群管理规模：50 节点；

（5）节点使用子网：intnetX-server；

（6）节点预留容器 IP 上限：64；

（7）容器网段：10.10.0.0/16。

创建一个集群节点，节点配置信息要求如下：

（1）节点名称：kcloud-server；

（2）节点规格：c6s.xlarge.2

（3）节点：EulerOS 2.9

（4）使用 docker 容器引擎

1.查看云容器是否被正确创建 cce.0.kcloud.v1.25.10.10.0.0/16 计 1 分

3.2.2 云容器管理[1 分]

使用插件管理在 kcloud 容器集群中安装 dashboard 可视化监控界面。

1.查看 dashboard 首页，有 Kubernetes Dashboard 返回计 1 分

3.2.3 使用 kubectl 操作集群[2 分]

在 kcloud 集群中安装 kubectl 命令，使用 kubectl 命令管理 kcloud 集群。

1.使用 kubectl 命令查询集群是否正确计 2 分

3.2.4 安装 helm[2 分]

在 k8s 集群中创建 chartmuseum 命名空间，编写 yaml 文件部署 ChartMuseum 服务。

使用提供的 helm 软件包（软件包为 helm-v3.3.0-linux-amd64.tar.gz 在 http 服务下），在 kcloud 集群中安装 helm 服务。

1.查看 helm 是否被正确安装，查看 helm 版本是否为 v3.3.0 计 2 分

3.2.5 chartmuseum 仓库部署[2 分]

在 k8s 集群中创建 chartmuseum 命名空间，编写 yaml 文件部署 ChartMuseum 服务。

在 k8s 集群中创建 chartmuseum 命名空间，编写 yaml 文件在 chartmuseum 命名空间中使用 chartmuseum:latest 镜像创建本地私有 chart 仓库，设置其仓库存储目录为宿主机的/data/charts 目录。编写 service.yaml 文件，为 chart 私有仓库创建 Service 访问策略，定义其为 ClusterIP 访问模式。编写完成后启动 chartmuseum 服务。

1.检测 chartmuseum 服务反馈是否正确计 2 分

3.2.6 WordPress 应用部署[2 分]

根据提供的 chart 包 wordpress.tgz 部署 WordPress 服务。

根据提供的 chart 包 wordpress-13.0.23.tgz 部署 wordpress 服务，根据 chart 包中内容创建 wordpress 所需要的 pv，并修改其访问模式为 NodePort。使用修改后的 chart 包创建 wordpress 服务。

1.查看 wordpress 前端页面是否正确计 2 分

任务 3 公有云运维开发[10 分]

3.3.1 开发环境搭建[1 分]

创建一台云主机，并登录此云服务器，基于云服务器 Python3.6.8 运行环境，安装 SDK 依赖库。

通过华为云控制台，选择北京四区域，创建一台 x86 架构，“按需计费”的 2 核，4G，系统盘 50G 的云实例，实例名为 chinaskill，选择镜像为 CentOS 7.9 64bit(40GB)，分配独立的公网 IP，带宽选择按使用流量计费 5M。登录此云服务器，在云服务器 Python3.6.8 环境上，安装华为云最新版本 SDK 的依赖库，包括弹性云服务、虚拟私有云、镜像服务、容器云引擎、云数据库的 python 库。

1.检查云主机开发环境，正确计 1 分

3.3.2 密钥对管理[2 分]

编写 Python 代码，实现密钥对的创建。

在云服务器的/root/huawei/目录下编写 create_keypair.py 文件，使用 SDK 编写 Python 代码，创建华为云的秘钥对，具体要求为：

（1）秘钥对名称：chinaskills_keypair;

（2）如果秘钥对已经存在，代码中需要先删除；

（3）输出此秘钥对详细信息。

1.执行 Python 文件，成功创建出 keypair 并输出详细信息得 2 分。

3.3.3 云硬盘管理[2 分]

调用 SDK 云硬盘管理的方法，实现云主机的的增删查改。

在/root/huawei 目录下编写 create_block_store.py 文件，使用 SDK 编写

Python 代码，调用创建华为云的云硬盘，具体要求如下：

（1）云硬盘可用区域：cn-north-4a

（2）云硬盘名称：chinaskills_volume

（3）云硬盘规格和大小：超高 IO，100G

（4）设置云硬盘共享

（5）设置云硬盘加密，加密秘钥为默认的 KMS 密钥

（6）如果云硬盘已经存在，代码中需要先删除

（7）输出此云硬盘的详细信息（状态要求为 available）

1.执行 Python 文件，成功创建出云硬盘并输出详细信息得 2 分。

3.3.4 云主机管理[3 分]

调用 SDK 云主机管理的方法，实现云主机的的增删查改。

使用已建好的运维开发环境，在/root/huawei 目录下创建 ecs_manager.py 脚本，完成 ECS 云主机管理，ecs_manager.py 程序支持命令行参数执行。

提示说明：华为云主机支持安装所需 Python 库。提交前答案前，需安装所开发程序所依赖的 Python 库。

（1）程序支持根据命令行参数，创建 1 个云主机。

位置参数“create”，表示创建；

参数“-i 或--input”，格式为 json 格式文本的云主机的名称、镜像名称 2 个信息。其他参数同上述开发环境云主机一致。

创建待成功，再返回查询该云主机的信息，结果以 json 格式输出到控制台。

参考执行实例如下：

python3 /root/huawei/ecs_manager.py create --input '{ "name": " chinaskill001", " imagename": "imageid"} '

（2）支持查询给定具体名称的 ECS 云主机查询。

位置参数“get”，表示查询 ECS；

参数“-n 或 --name”支持指定名称 ECS 查询，类型为 string。

参数“-o 或 --output”支持查询该 ECS 信息输出到文件，格式为 json 格式。

（3）程序支持查询目前区域账号下所有的 ECS 云主机。

位置参数“getall”，表示查询所有 ECS 云主机；

参数“-o 或--output”支持输出到文件，格式为 yaml 格式。

（4）支持删除指定名称的云主机。

位置参数“delete”，表示删除一个 ECS 云主机；返回 response，通过控制台输出。

参数“-n 或--name”支持指定名称查询，类型为 string。

1.执行 ecs_manager.py 脚本，指定 create 和配置参数，成功创建 1 台云主机，计 1 分 2.执行 ecs_manager.py 脚本，指定 get 和配置参数，成功查询具体名称云主机，计 1 分 3.执行 ecs_manager.py 脚本，指定 getall 和配置参数，成功查询目前区域账号下所有的云主机，计 1 分 4.执行 ecs_manager.py 脚本，指定 delete 和配置参数，成功删除指定的云主机，计 1 分

3.3.5 基于 FastAPI 框架封装华为云 VPC 服务 Restful APIs 接口[2 分]

使用已建好的运维开发环境，可用区为华北-北京 4，在 /root/huaweicloud_fastapi 目录下创建 main.py 文件，基于 FastAPI 框架，完成虚拟私有云 VPC 管理服务的封装，端口为 7045。

提示说明：华为云主机支持安装所需 Python 库。提交前答案前，需安装所开发程序所依赖的 Python 库。

（1）封装创建接口，路径为/cloud_vpc/create_vpc，参数格式为{"name": "","cidr": ""},name 为 VPC 名称，cidr 为虚拟私有云下可用子网的范围；方法为 POST，返回数据类型为 JSON 类型。

（2）封装单个 VPC 查询接口，路径为/cloud_vpc/vpc/{vpc_name}，路径中{vpc_name}为待查询 VPC；方法为 GET，返回数据类型为 JSON 类型。

（3）封装服务查询接口，要求可以查询到所有 VPC，路径为 /cloud_vpc/vpc；方法为 GET，返回的数据类型 JSON 类型的列表类型。

（4）封装更新 VPN 名称接口，路径为/cloud_vpc/update_vpc，参数格式为{"new_name": "","old_name": ""}，new_name 为 VPC 新名称，old_name 为待删

除的 VPN 名称；方法为 PUT，返回 JSON 类型的更新后数据。

（5）封装删除接口，接口名为/cloud_vpc/delete_vpc，参数格式为{"vpc_name": ""}，vpc_name 为 VPC 新名称；方法为 DELETE，返回 JSON 类型的删除后的数据。

1.创建接口执行成功得 0.5 分。
2.查询接口执行成功得 0.5 分。
3.更新接口执行成功得 0.5 分。
4.删除接口执行成功得 0.5 分。

任务 4 边缘计算系统运维[10 分]

【适用平台】私有云

4.1.1 KubeEdge cloudcore 云端模块部署与配置[3 分]

在本地 http 文件服务器中下载 k8s-allinone-v1.22.1.qcow2 镜像（该镜像启动后内置一个完整的 kubernetes 服务）并完成下列操作：

（1）上传该镜像至公有云 OBS 存储桶中；

（2）使用该镜像文件创建为私有镜像；

（3）使用该镜像创建一台云主机，该云主机作为 KubeEdge 的云端节点。

云主机申请完成之后，在提供的 OBS 访问域名中，下载软件包 kubernetes_kubeedge.tar 到云端节点并部署 KubeEdge1.11 边缘计算系统，在云端节点部署 KubeEdge cloudcore 云端模块、启动 cloudcore 服务并配置该服务可被 systemd 管理。

1.cloudcore 服务正常启动得 1.5 分
2.cloudStream 端口正常启动得 1.5 分

4.1.2 KubeEdge edgecore 边端模块部署与配置[4 分]

在本地 OpenStack 环境中申请两台 CentOS7.9 虚拟机做为 KubeEdge 的边缘

端节点。主机名分别为 edge-node1、edge-node2。在 http 文件服务器中下载 kubernetes_kubeedge.tar 软件包，使用该软件包在该虚拟机部署 KubeEdge edgecore 边端模块，并启动 edgecore 服务。加入成功之后，启用 metrics 监控服务。

1.edgecore 服务正常启动得 2 分。 2.边缘节点加入到云端得 2 分。

4.1.3 KubeEdge 应用部署-部署计数器应用[3 分]

利用搭建好的边缘计算平台，在云端节点上使用提供的相关镜像（k8simage 目录下）和 yaml 文件（kubeedge-counter-demo 目录下），自行编写 DeviceModel、Device 资源，要求如下：

DeviceModel 名字为 counter-model；

属性为 status，类型为字符串，存储模式为读写模式，默认值为空；

Device 名字为 counter；

引用 counter-model 设备模型；

将此设备通过节点选择器与 edge-node1 节点进行绑定；

在 status 字段中的 twins 字段定义属性名称为 status、理想值为“OFF”、真实值为“0”；

编写完成后，按照顺序分别部署好 DeviceModel、Device 以及云端计数应用控制器和计数应用。部署完成后，将计数器打开。

1.Device 与 DeviceModel 配置正确得 1.5 分。 2.计数器服务部署成功得 1.5 分。

任务 5 边缘计算云应用开发[5.0 分]

【适用平台】私有云

5.1.1 开发环境构建与用户注册服务开发[1 分]

使用华为云已搭建好的 KubeEdge 环境，安装 Python 3.7.3 运行环境，并安装开发所需的依赖包，使用提供的 mongoRepo.tar.gz 软件包在边端安装 MongoDB 数据库，设置数据库为任何机器都可以连接登录。

开发环境搭建好之后，将提供的代码包导入到 Linux 服务器 root 目录下，并打开 fastapi_user 工程目录进行用户管理服务开发，微服务端口为 8046，运行服务后，注册名为 chinaskill 的用户，密码为 Abc@1234，手动启动代码运行服务。

1.环境搭建成功得 0.5 分。 2.服务启动后用户注册成功得 0.5 分。

5.1.2 AI 检测服务与网关服务开发[1 分]

使用现有的开发环境，将提供的代码包导入到 Linux 服务器 root 目录下,打开 fastapi_ai_pcb 工程目录，基于 YOLO5 与已训练的模型实现 PCB 缺陷 AI 识别服务，fastapi 框架来定义 PCB 图片的缺陷检测接口，调用接口时，需要指定 device 实例名称和上传缺陷图片，当识别到缺陷图片后，返回图片 base64 编码后的数据、模型的名字、当前的上传时间、以及识别坐标的位置信息和实例名称。

然后打开 fastapi_gateway 工程目录，基于 fastapi-gateway 开源框架实现网关服务，AI PCB 微服务端口为 8055，微服务网关端口 30850 通过这个网关来访问用户管理服务和 AI PCB 识别服务，手动启动 AI PCB 和网关服务。

AI 接口接口路径为：/detect/pcb_detect，参数传递到路径接口上，参数名为 device_id。

识别图片在工程目录中的 runs 目录里。

1.识别代码编写正确得 0.5 分。 2.服务启动后功能正常使用得 0.5 分。

5.1.3 设备与设备模型管理服务开发[1 分]

使用现有的开发环境，将提供的代码包导入到 Linux 服务器 root 目录下，打开 fastapi_cloud_edge 工程目录，基于 FastAPI 框架和 Kubernetes SDK 实现了一组 API，主要针对 KubeEdge 拓展的 CRD 的资源进行访问封装，微服务端口为 8045。

1、完善设备模型管理服务，要求如下：

（1）定义获取 DeviceModel 资源接口，接口名为/devicemodel/devicemodel，方法为 GET；

（2）定义获取单个 DeviceModel 接口，接口名为/devicemodel/{name}，方法为 GET；

（3）定义创建 DeviceModel 接口，接口名为/devicemodel/devicemodel，方法为 POST， 参数格式为{"name": "", "properties": ""}，name 为 DeviceModel 的名字，properties 为 DeviceModel 更新的属性内容，类型为列表类型；

（4）定义设备模型同步接口，接口名为/devicemodel/devicemodel，方法为 PUT，参数格式为{"name": "", "properties": ""}，name 为 DeviceModel 的名字，properties 为 DeviceModel 创建的属性内容，类型为列表类型；

（5）定义删除 DeviceModel 资源接口，接口名为/device/devicemodel，可以删除指定 DeviceModel，参数格式为{"name": ""}，name 为 DeviceModel 的名字，方法为 DELETE。

2、完善设备管理服务，要求如下：

（1）定义获取所有的 device 资源，接口名为/device/device，方法为 GET；

（2）定义获取单个 device，接口名为/device/{name}，方法为 GET；

（3）定义设备同步接口，接口名为/device/device，参数格式为{"name": "","desired": {}}，name 为更新设备的名字，desired 为更新设备的内容，内容格式为 JSON 格式，方法为 PUT；

（4）定义创建 device 接口，接口名为/device/device，参数格式为{"name":"","dmName": "","nodeName": ""}，name 为 device 的名字，dmName 为 DeviceModel 的名字，nodeName 为边缘节点名字，方法为 POST；

（5）定义删除 device 接口，可以删除指定的 device 资源，接口名为/device/device，参数格式为{"name": ""}，名字为要删除的 device 名字，方法为

DELETE。

手动启动设备与设备模型管理服务。

1.设备代码编写正确得 0.5 分。 2.服务启动后接口可以正常使用得 0.5 分。

5.1.4 云边数据协同开发[1 分]

使用现有的开发环境，将提供的代码包导入到 Linux 服务器 root 目录下，打开 fastapi_cloud_edge 工程目录,在前面几道题的基础之上，对 pcb_data_manager.py 文件完善代码，要求如下：

将数据模型上传到数据库中，并返回该模型在数据库中的 ID，若有相同模型则不上传；

定义接收边缘消息程序；

云端接收边端 MQTT 主题里所发送的数据，将数据进行解析，存储到 MongoDB 数据库中，并对云端 pcb-device 设备信息进行更新。

设备信息字段为 pcb_model（模型名字）、pcb_status（设备状态）、pcb_result（模型在数据中的 ID）、upload_time（识别图片的时间）

1.代码编写正确得 0.5 分。 2.可以正常接收数据并更新设备信息得 0.5 分。

5.1.5 云端节点管理开发[1 分]

使用现有的开发环境，将提供的代码包导入到 Linux 服务器 root 目录下，打开 fastapi_cloud 工程目录，基于 FastAPI 框架和 Kubernetes SDK 实现了一组 API，来进行管理云端节点和边缘节点信息的后端服务，微服务端口为 8047，编码要求功能如下：

定义获取云端节点信息的路由，接口为 /cloud_edge_service/node/cloudnode，请求为 GET，返回的数据类型为列表类型。

定义获取单个节点信息的路由，接口为/cloud_edge_service/node/{node_name}，请求为 GET，返回数据类型为字典类型。

定义获取边缘节点信息的路由，接口为/cloud_edge_service/node/edgenode，请求为 GET，返回数据类型为列表类型。

定义节点资源使用情况路由，接口为/cloud_edge_service/metrics/node/{node_name}，请求为 GET，返回获取到的资源使用情况,将任意一台边缘节点的主机名存储到/root/name.txt 文件里。

手动启动云端节点管理服务。

1.节点代码编写正确得 0.5 分。 2.服务启动后接口可以正常使用得 0.5 分。