**四川农业大学**

**容器迁移用户指南**

版 本： v1.1

编 辑： 侯 双

审 核： 王继涛

发布日期： 2024年3月13日

版本：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 修订时间 | 修订人 | 审核人 | 修订章节 | 修订内容 |
| 1.0 | 2023/06/20 | 侯双 | 王继涛 | 全部 | 初版编写 |
| 1.1 | 2023/03/13 | 侯双 | 王继涛 | 2.2 | VPN用户指南 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1. 前言 3](#_Toc16477)

[2. 平台使用 3](#_Toc22378)

[2.1 平台简介 3](#_Toc178)

[2.2 OpenVPN用户指南（校外用户必看） 3](#_Toc7678)

[2.2.1 OpenVPN客户端安装 3](#_Toc3420)

[2.2.2 客户端登录 4](#_Toc29040)

[2.3 堡垒机用户指南 5](#_Toc22237)

[2.4 容器平台用户指南 6](#_Toc14598)

[2.4.1 平台简介 6](#_Toc17647)

[2.4.2 应用部署 12](#_Toc3320)

[3. 技术要求 24](#_Toc10434)

[3.1 docker容器技术 24](#_Toc5160)

[3.2 K8s技术快速入门 24](#_Toc1625)

[3.3 Docker镜像制作 25](#_Toc141)

[3.4 常见应用Dockerfile 26](#_Toc5576)

[3.5 常见应用yaml文件 28](#_Toc10237)

[4. 架构规划 28](#_Toc9052)

[5. 常见问题 28](#_Toc25656)

# **前言**

本文为四川农业大学容器迁移用户指南。指导各校内用户、软件服务商、运维实施人员，完成贵单位校内业务由VMware平台向容器平台的迁移。为了您更好的理解并使用文档，我们希望您具有一定的Linux操作经验、具备docker容器技术、熟悉常见的kubernetes（以下简称k8s）操作命令、具备一定的容器项目实施经验。

您在使用此文档时，如果发现无法解决您的现有问题，请您及时与文档管理者取得联系，帮助解决您的现有问题；如果您对文档或产品有改进的意见或建议时，也请您与文档管理者取得联系，我们会结合您建议进行改进；感谢您的配合。

# **平台使用**

## 平台简介

四川农业大学容器平台是基于docker和kubernetes本地化部署，属于私有云。本集群相关软硬件产品均部署在学校数据中心机房，由学校老师专职管理维护。为校内各业务单位、应用系统提供容器化运行环境。

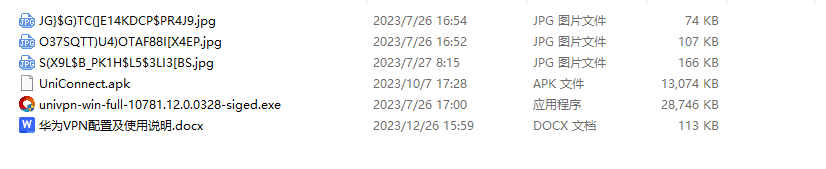
容器平台由三台Master节点、四台Node节点、一台混合式集中存储、四台分布式存储组成，可提供高达cpu400核、2TB内存、1.3PB存储的运行环境，集群暂不提供GPU算力。集群环境部署版本Docker v19.03.15，kubernetes v1.20.11，KubeSphere v3.2.1。平台采用多租户管理模式，为每个业务系统分配独立的namespace和管理账号，各业务系统运行环境相互隔离，确保各项业务安全、可靠、稳定运行。平台提供web管理界面，支持主流浏览器，可通过web端实现应用的部署、调试、扩容、资源监测等操作。

## UniVPN用户指南（校外用户必看）

UniVPN提供专用虚拟通道，用户可在公网环境下访问内网应用。非校内网络用户，必须先连接UniVPN才能访问分配的容器资源，校内网络用户可忽略本项操作内容。

### UniVPN客户端安装

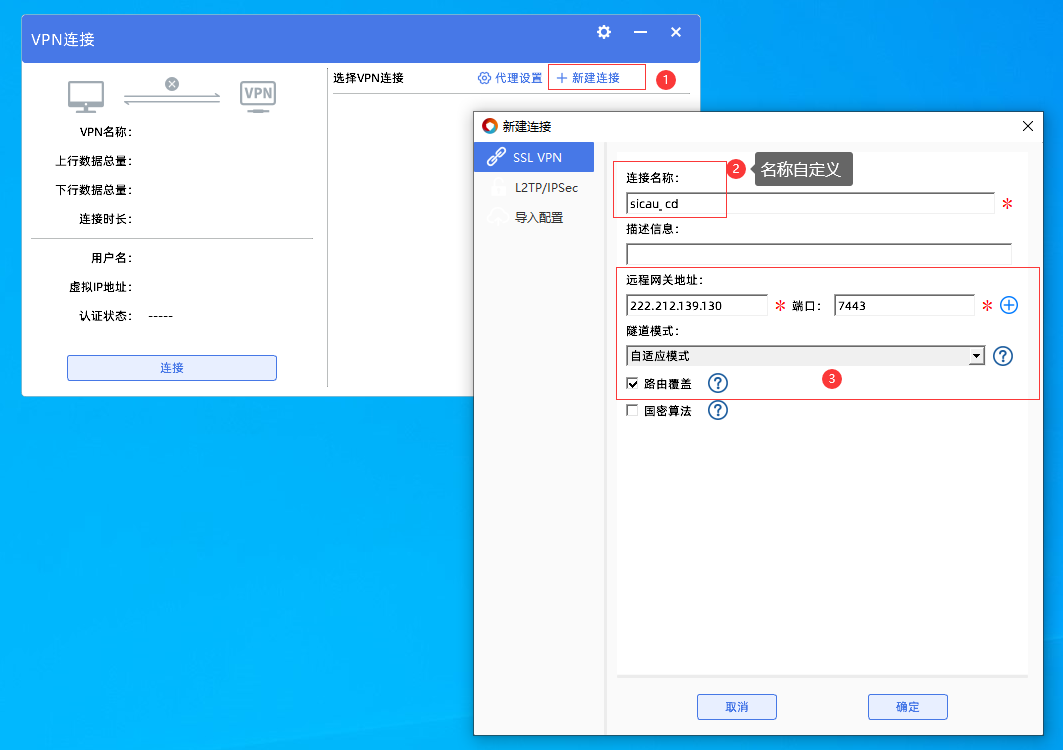
1）学校提供UniVPN客户端软件安装包，用户根据系统版本选择并安装。



1. 安装完成后，配置VPN连接地址；

**VPN连接地址1：**222.212.139.130，端口7443

**VPN连接地址2：**125.67.197.2，端口7443



### 客户端登录

用户名称和密码：统一身份认证账号及对应密码



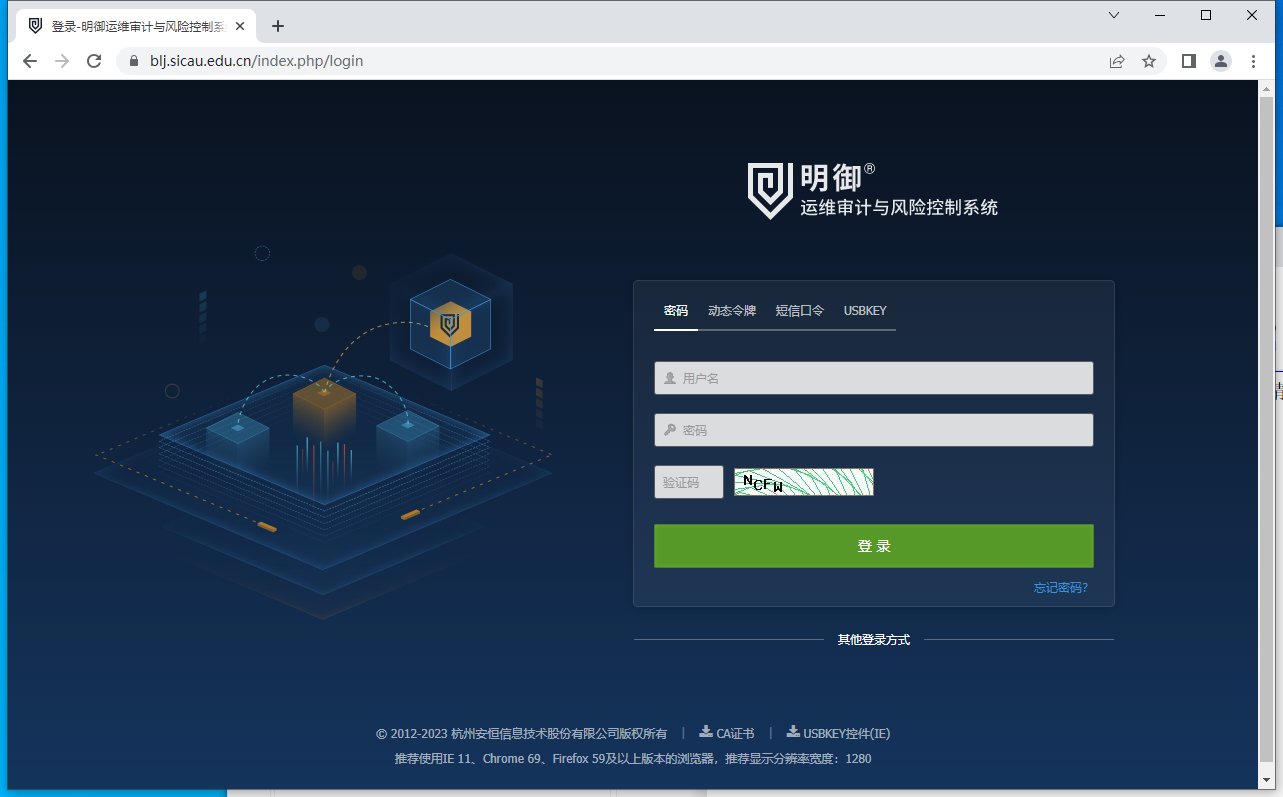


## 堡垒机用户指南

学校分配的容器资源无法直接访问，需通过堡垒机进行访问。请使用分配的堡垒机账号、密码登录运维审计系统。

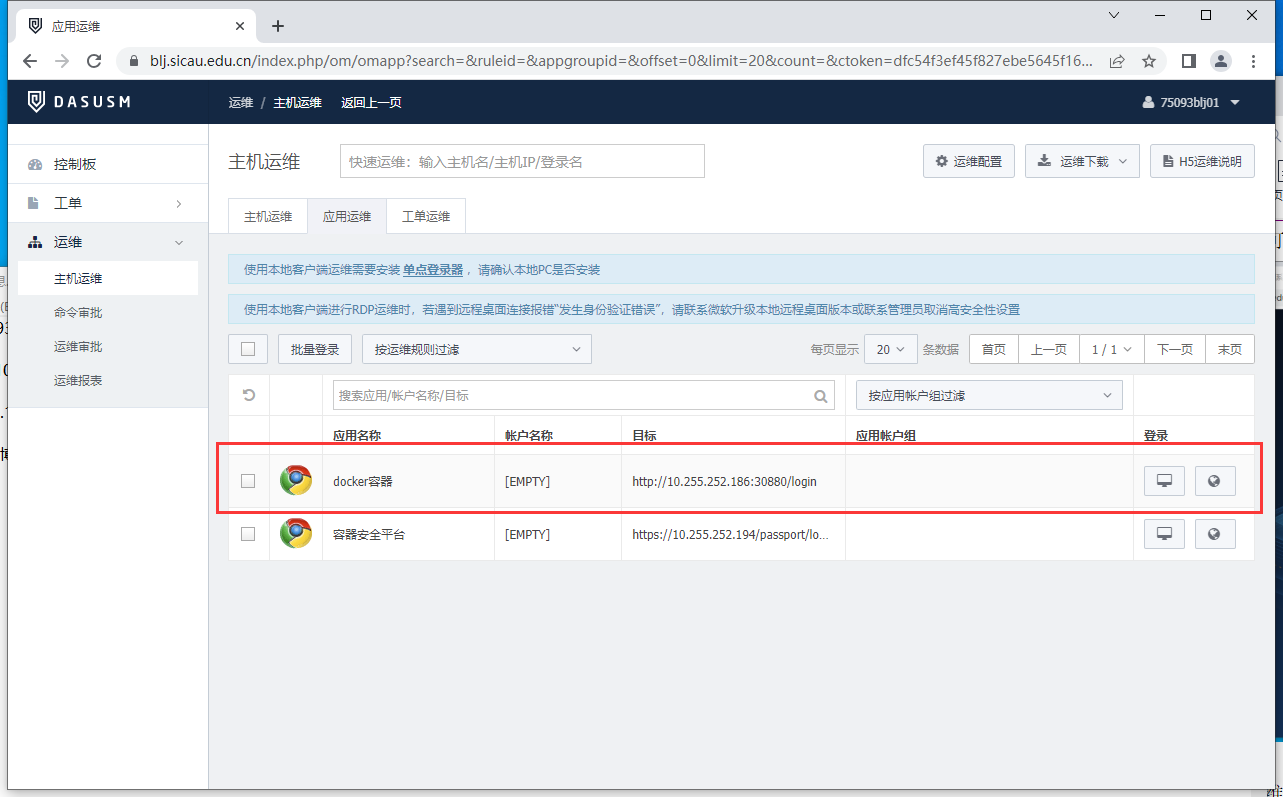
堡垒机地址：[https://blj.sicau.edu.cn](https://blj.sicau.edu.cn/index.php/login)

校内网络用户可直接访问，校外网络用户请先[连接UniVPN](#_OpenVPN用户指南（校外用户必看）)。



运维—主机运维—应用运维

Docker容器运维，可通过本地客户端和H5客户端两种方式登录。使用本地客户端登录，请先安装 **单点登录器** 。





## 容器平台用户指南

学校分配的容器资源不允许本地浏览器直接访问，需通过堡垒机进行访问。操作本系统前，我们希望你具备一定的linux、Docker、kubernetes基础知识。

### **平台简介**

#### 系统登录

请使用学校为你分配的用户名和密码登录系统。初次登录系统时，弹窗提示需要修改初始密码。密码强度要求：至少包含大小写字母、数字和符合，长度不低于8位。

登录成功后，项目中即可看到学校为你分配的容器资源。此处项目名称ietc-manager 对应 k8s 中的命名空间namespace。

点击项目名称，即可进行项目空间内部。

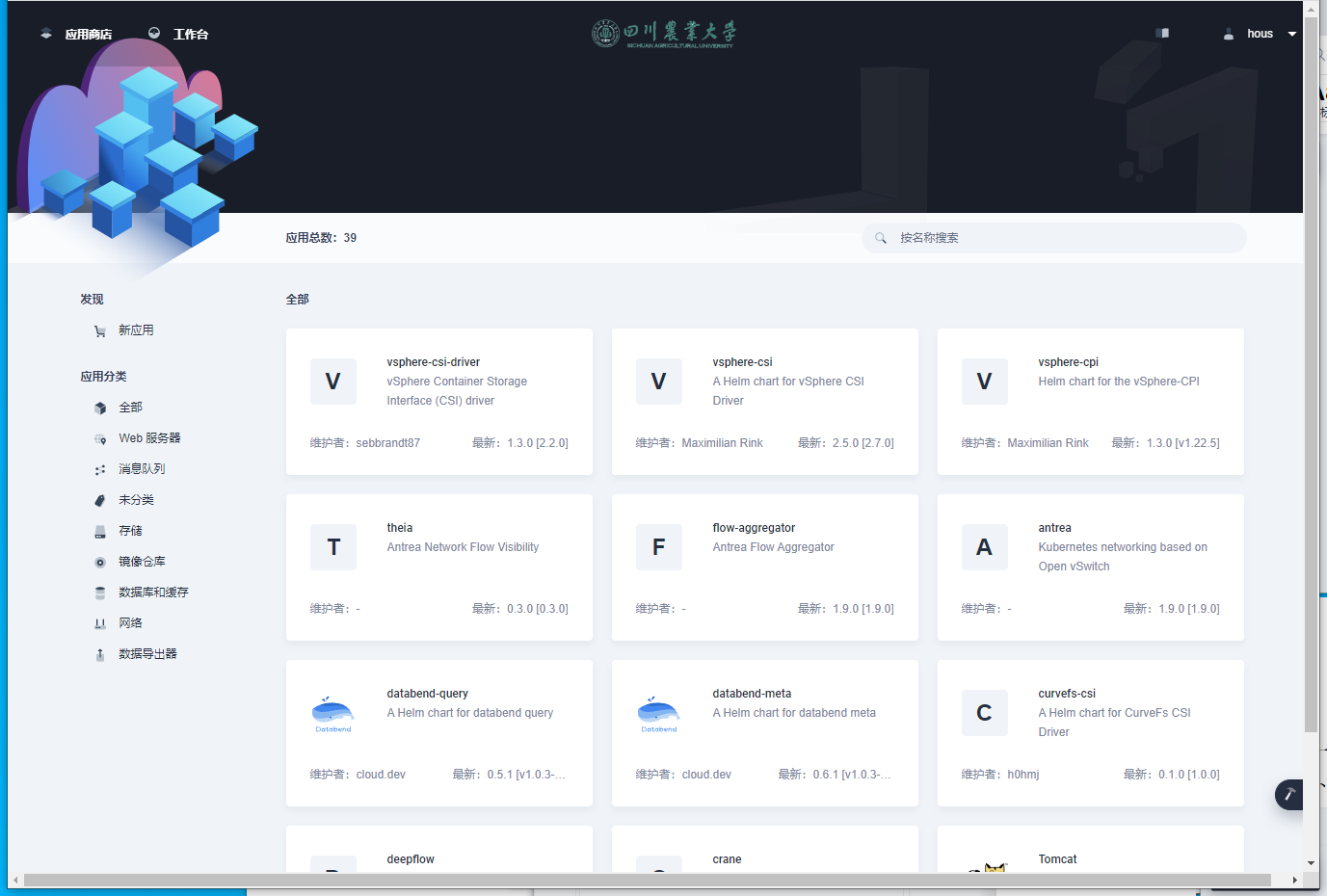




#### 应用商店

K8s基于helm 包管理器创建的应用。Helm类似centos系统的yum包管理器、ubuntu系统的apt-get包管理器。Helm部署应用简单快捷，通过几项简单的配置即可完成应用部署。平台目前发布的helm应用包，可能存在更新不及时，可定制化程度低，用户可根据自身情况决定是否使用helm部署应用。

Helm相关操作命令，使用指南请自行查阅资料。



#### 概览

项目基础信息一览，包含所属空间、容器数量、cpu、内存等系统资源监测。支持按资源类型（cpu、内存、出入站流量）对部署的容器进行排序。



#### 应用负载

应用：基于Helm包管理器部署的应用；

服务：对应k8s中的service服务；

工作负载：对应k8s中各类pod控制器。**部署**对应k8s中的无状态应用Deployment，**有状态副本集**对应k8s中的有状态应用StatefulSet，**守护进程集**对应k8s中的DaemonSet；

任务：**任务**对应k8s中的一次性任务job，定时**任务**对应k8s中的定时任务cronjob；

应用路由：对应k8s中的ingress；

容器组：查看项目空间下部署的容器及状态；



#### 存储

存储卷：对应k8s中的PersistentVolumeClaim，简称pvc；

存储卷快照：对应k8s中的VolumeSnapshotContent；



#### 配置

保密字典：支持创建TLS证书信息、私有厂库信息、用户名和密码信息；

配置字典：对应k8s中的confMap；



#### 灰度发布

灰度发布是在生产环境进行应用迭代的一种重要方式。您可以选择不同的发布方法，在应用升级至新版本的过程中实现平滑过渡。



#### 镜像构建器

可通过连接代码厂库或上传打包后的程序文件构建docker镜像，无需手写Dockerfile文件，即可构建新镜像。



#### 监控告警

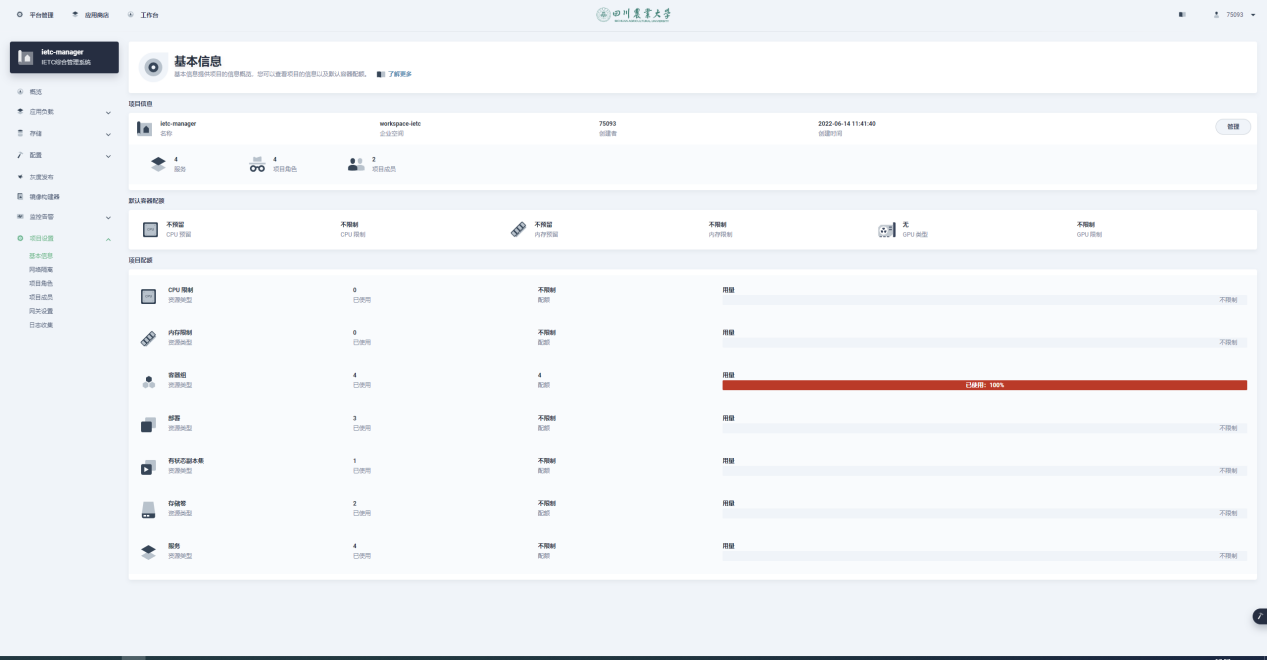
用户可针对本项目的各类应用配置告警策略，当系统资源（cpu、内存、网络流量、副本数量）达到设定的告警规则时，系统发出资源告警；

自定义告警监控：用户可自定义创建监控大屏；



#### 项目设置

项目基本信息、项目成员信息、项目角色信息、日志信息等；

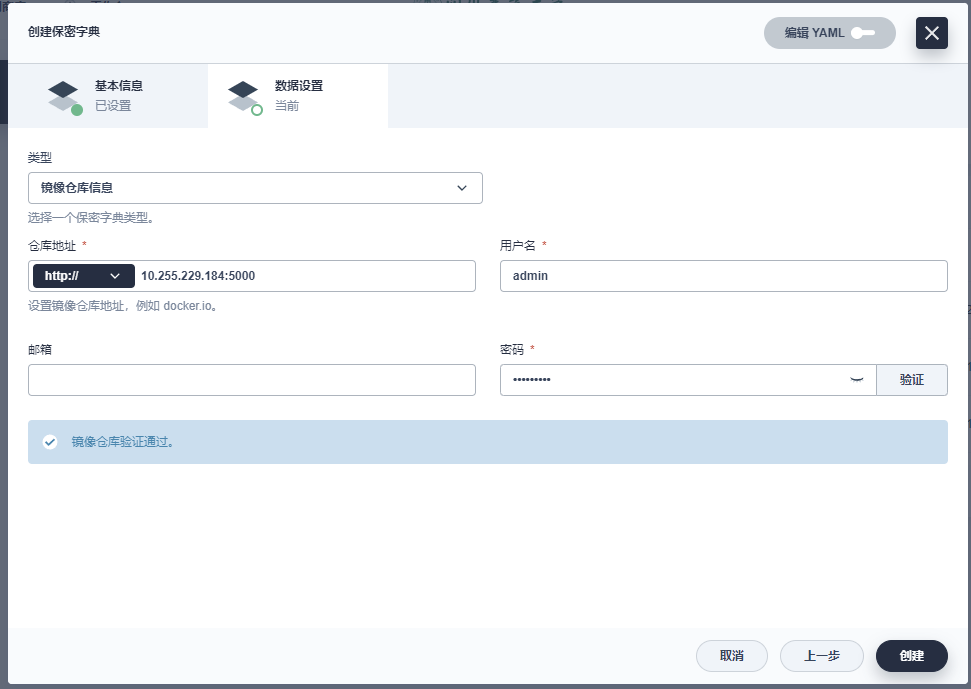


### **应用部署**

#### 配置私有仓库

**配置**—**保密字典**，点击**创建**按钮，设置仓库名称、别名、描述信息，选择保密字典类型为镜像仓库信息，输入仓库地址，用户名及密码，并验证仓库信息是否正确。



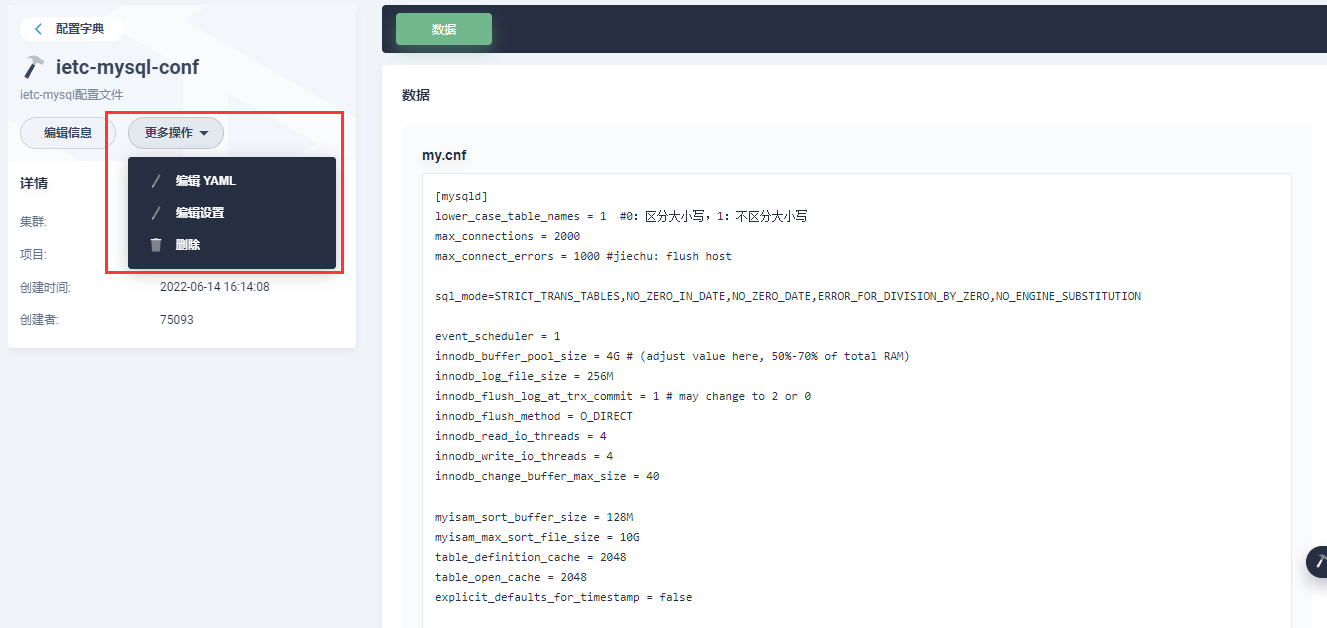


#### 创建配置信息confMap

创建confMap配置；可通过编辑yaml文件方式修改；







#### 创建存储卷（PVC）

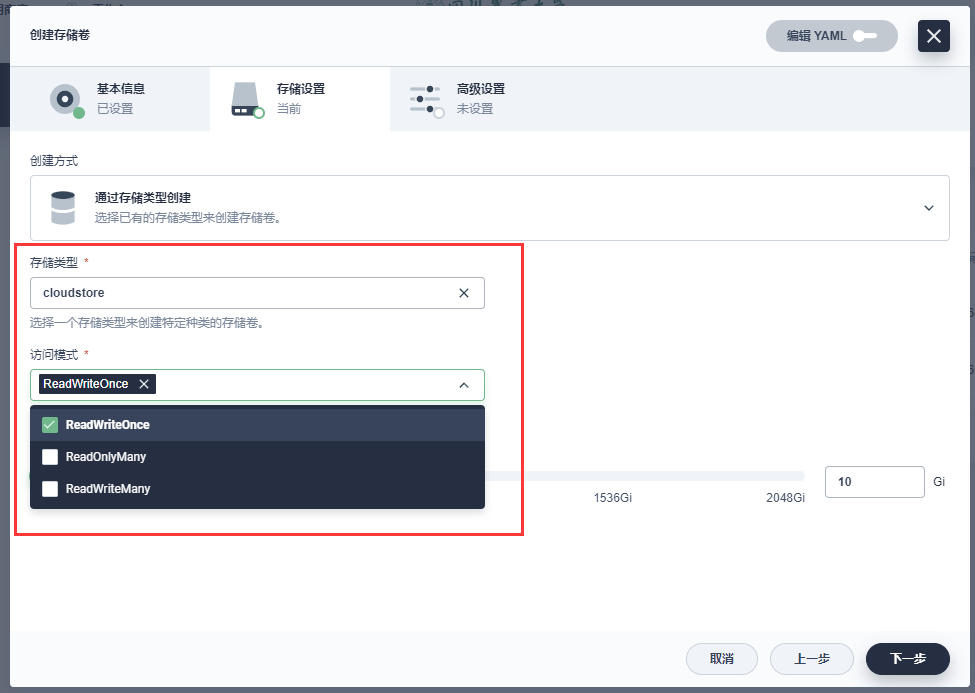
创建存储卷，指定存储卷读写类型、存储容量。若你的账号无创建存储卷权限时，请联系平台管理员为你分配。

存储卷三种读写模式：

ReadWriteOnce：支持单个node节点上的多个pod副本读写文件，若多个pod副本调度到不同node节点时，仅有一个node节点上的pod可正常启动并读写文件；

ReadWriteMany：支持多个node节点上的多个pod副本读写文件；

ReadWriteOnce：只读卷，只能读取文件，无法写入；



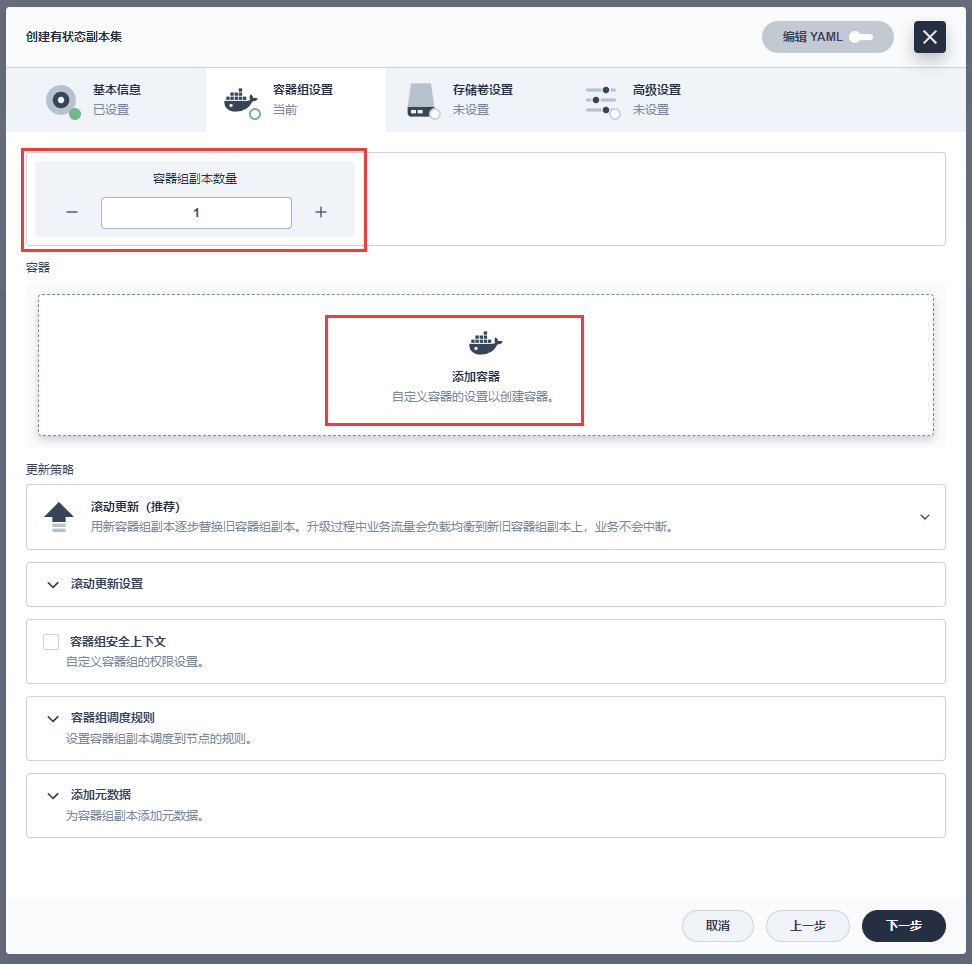
#### 应用部署（Deployment、StatefulSet）

此处以部署有状态应用StatefulSet **mysql数据库**为例，无状态应用Deployment部署类似。

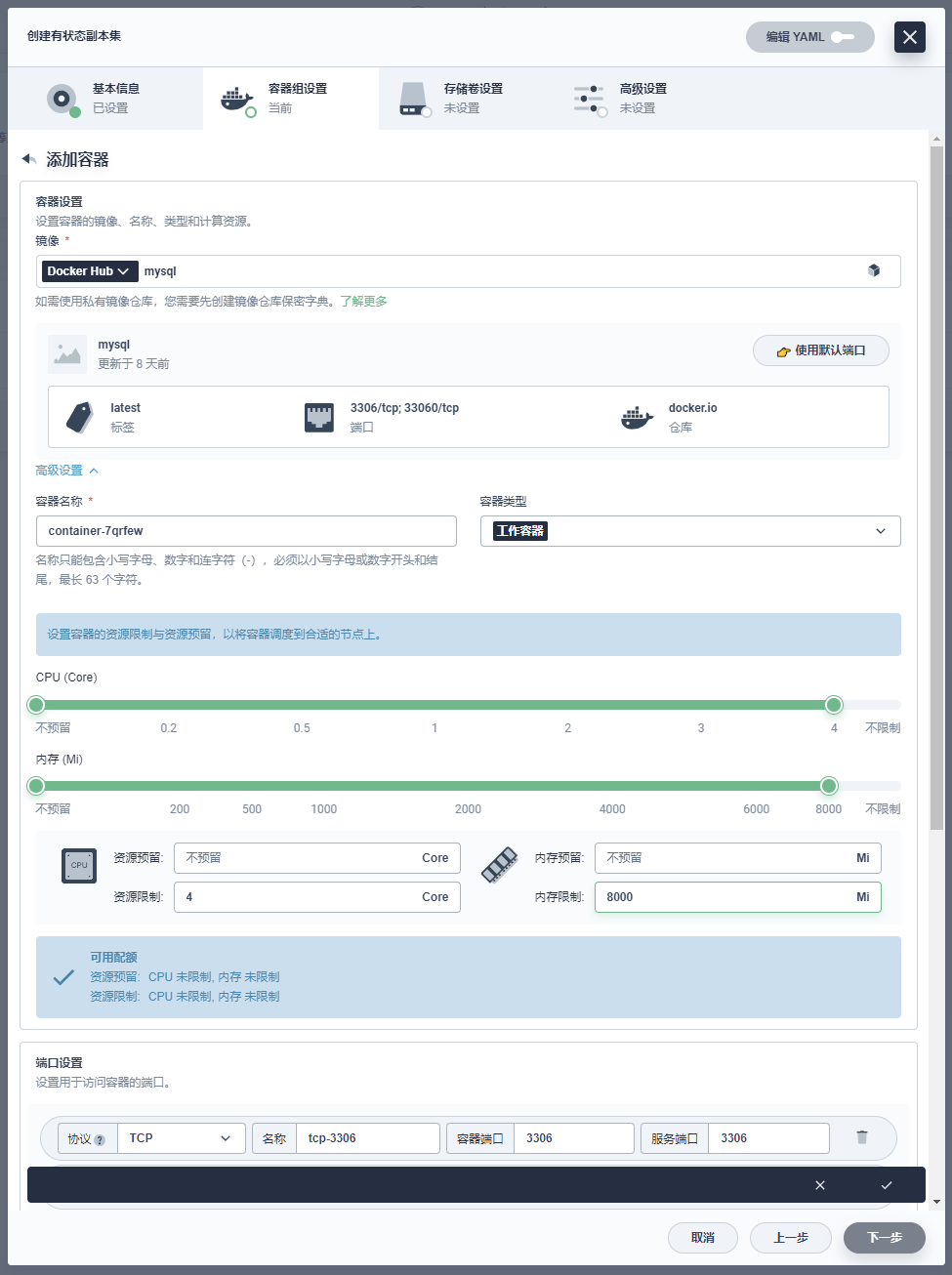
工作负载—有状态副本级—创建—输入“名称、别名、描述信息”



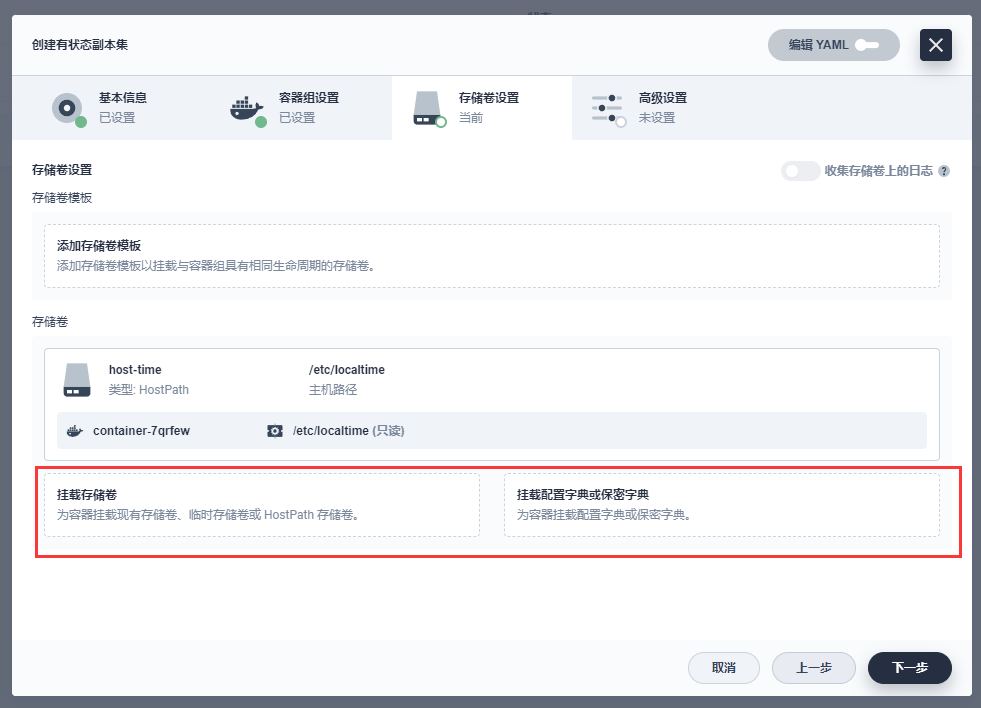
设定容器组副本数量、添加容器镜像、配置更新策略；对yaml文件熟悉可直接编辑yaml文件部署应用；



选择对应的厂库信息，搜索镜像、配置容器类型、cpun内存资源限制、端口、环境变量等信息；



挂载存储卷（pvc）和配置文件（confMap）



可配置应用调度到固定节点；若pod应用无特殊要求，建议不配置本项，pod可在集群内部的灵活调度，单节点故障不影响业务系统正常运行。





#### 创建服务（service）

服务—创建，选择服务类型，此处使用**指定工作负载**为mysql创建服务；



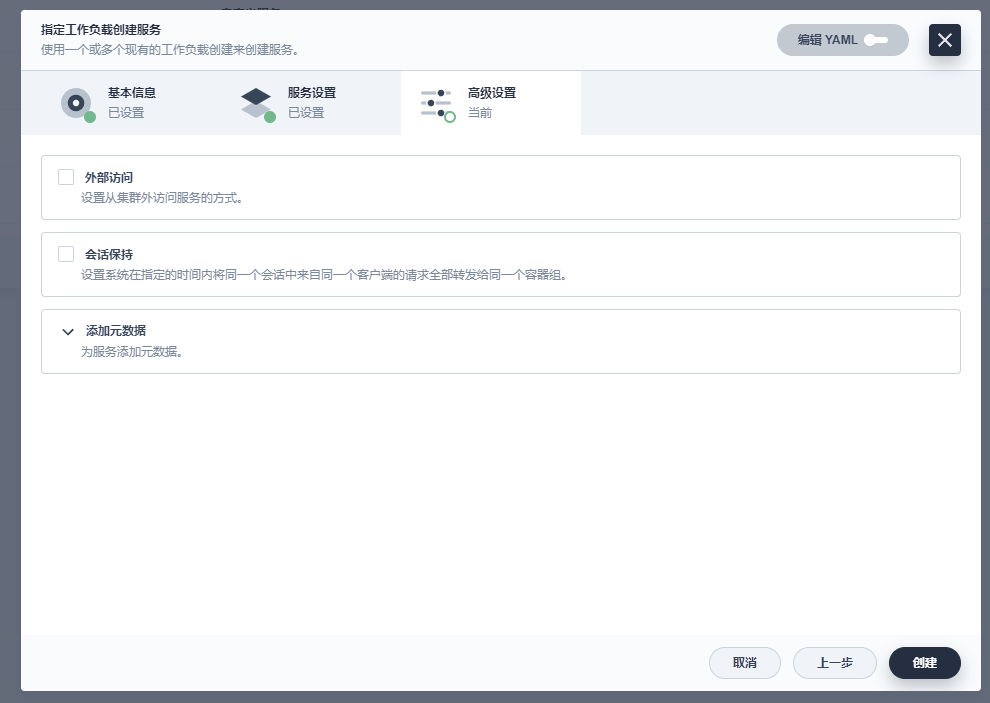


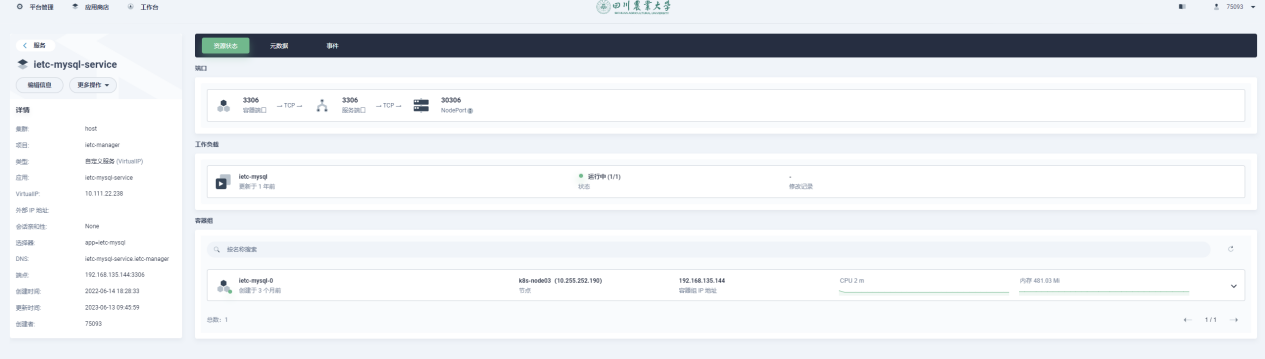
工作负载选择器处 指定 需要关联的应用，配置端口信息；



可配置外部访问模式、会话保持等信息；

外部访问模式分为NodePort和LoadBalancer两种；





说明：可参照有状态应用mysql的部署流程，完成其他应用的部署；在应用部署时，针对不同类型的应用选择Deployment或StatefulSet，请自行查阅资料。

mysql、Elasticsearch等数据存储类应用建议使用StatefulSet部署；

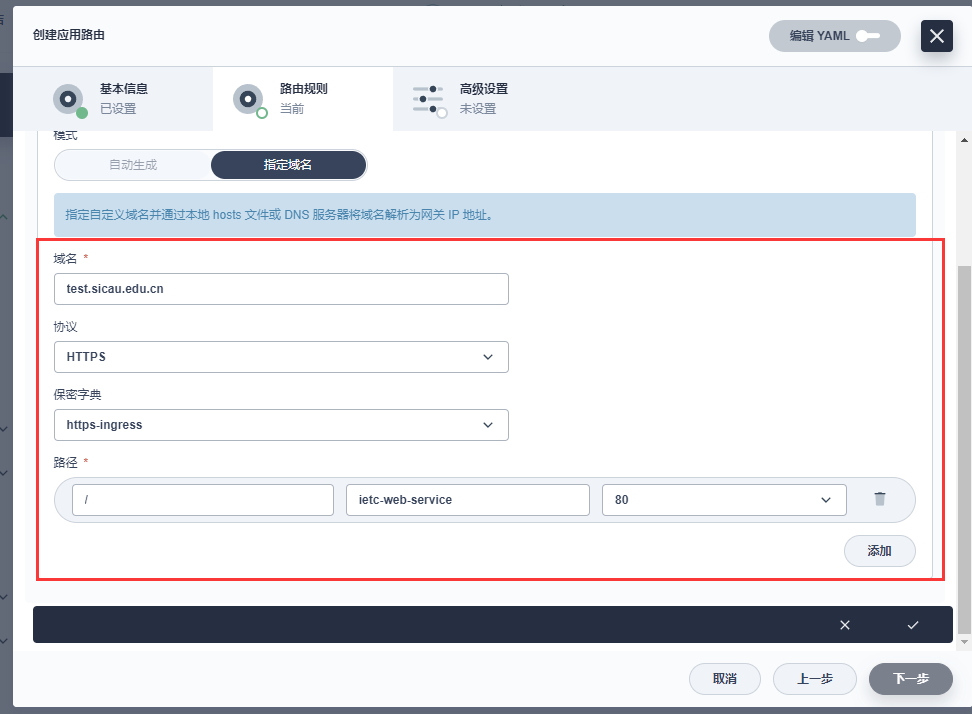
#### 配置https证书

按网络安全管理要求，上线的应用必须走https加密访问，禁止使用http访问。在k8s集群中部署应用时，可不在应用程序中配置https证书，将https证书配置到ingress侧即可实现https加密访问；



#### 创建路由（ingress）

应用负载-应用路由，创建ingress域名映射



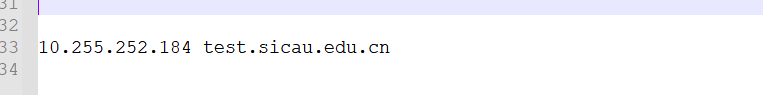


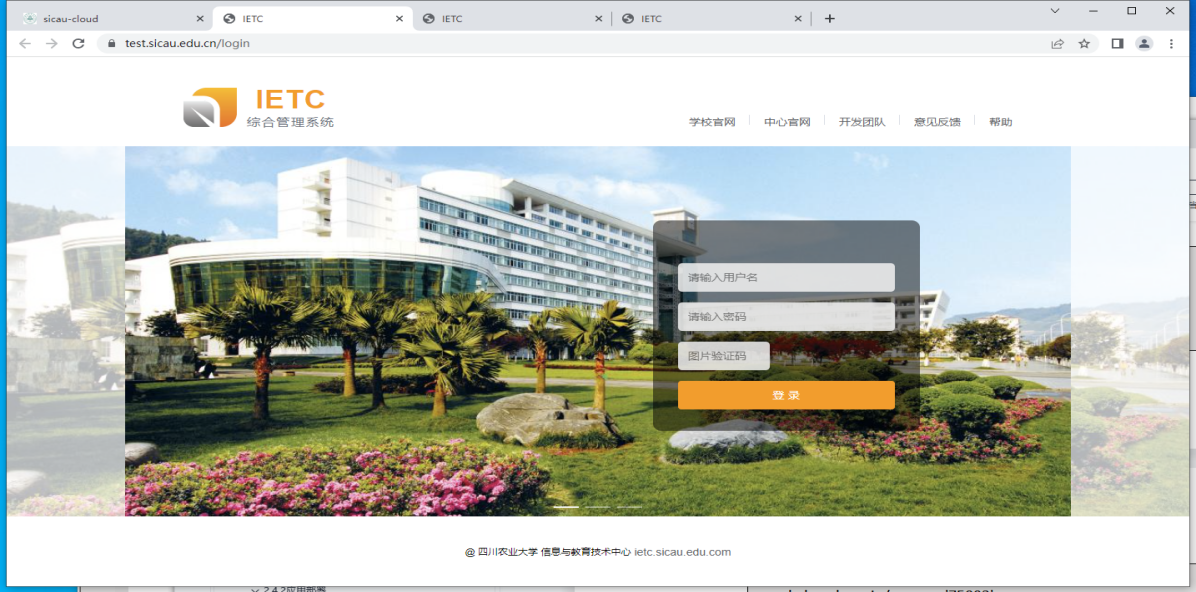
配置完成后，需手动编辑yaml文件，添加ingress 选择器类型，详细文件如下：

|  |
| --- |
| kind: Ingress  apiVersion: networking.k8s.io/v1  metadata:  name: ietc-ingress  namespace: ietc-manager  annotations:  kubesphere.io/creator: '75093'  spec:  # 配置完成后，需在yaml文件中配置ingress选择器类型  ingressClassName: nginx  tls:  - hosts:  - test.sicau.edu.cn  secretName: https-ingress  rules:  - host: test.sicau.edu.cn  http:  paths:  - path: /  pathType: ImplementationSpecific  backend:  service:  name: ietc-web-service  port:  number: 80 |

#### 配置本地hosts文件，验证ingress配置

手动添加一条域名解析指向10.255.252.184





# **技术要求**

我们真诚的希望您具有一定的Linux操作经验、具备docker容器技术、熟悉常见的kubernetes操作命令、具备一定的容器项目实施经验。关于docker容器技术和k8s容器编排技术，我们提供了一些博客文档供你参阅。

## docker容器技术

Docker教程：<https://www.runoob.com/docker/docker-tutorial.html>

Docker快速入门博客：<https://blog.csdn.net/qq_39578545/article/details/107741565>

Docker常用命令：<https://zhuanlan.zhihu.com/p/583421480?utm_id=0>

Docker官网：<https://docs.docker.com/desktop/>

Docker官方镜像厂库：<https://registry.hub.docker.com/>

## K8s技术快速入门

K8s基础概念：

<https://blog.csdn.net/qq_44239779/article/details/127633338?spm=1001.2014.3001.5501>

K8s基础架构：

<https://blog.csdn.net/aquester/article/details/104340076?spm=1018.2226.3001.4187>

K8s部署nginx实战案例：

<https://blog.csdn.net/weixin_45710811/article/details/126256165?spm=1018.2226.3001.4187>

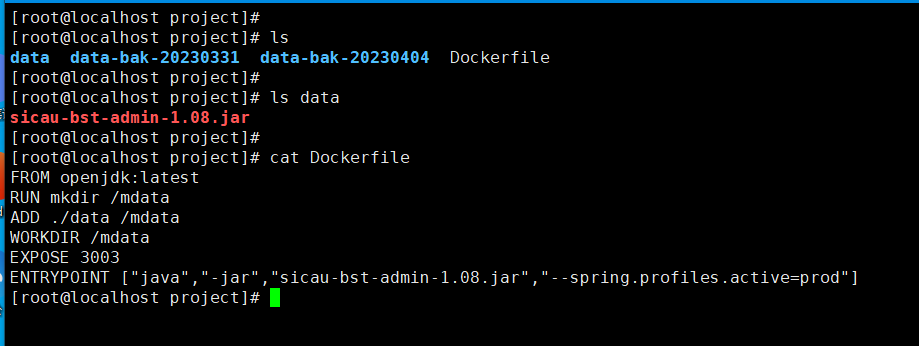
## Docker镜像制作

此处以基于Java语言编写的程序后端，打包成jar包为例，构建docker镜像；

环境要求：

一台安装有docker程序的liunx服务器，服务器可访问互联网，需要从公网拉取Jdk基础镜像。

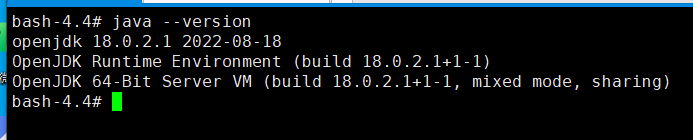
|  |
| --- |
| docker pull openjdk:latest #拉取openjdk镜像  mkdir /mdata/project #新建项目目录  cd /mdata/project #进入项目目录  mkdir ./data #项目目录下创建data目录，将打包后的\*.jar包上传至data目录中  vim Dockerfile #项目目录中创建Dockerfile文件  docker build -t jar-test:v1.01 ./ #使用build命令构建镜像  jar-test 镜像名称，v1.01 镜像tag版本，./ 表示当前路径下的Dockerfile 文件（此处请确保data目录和Dockerfile文件在同一目录下）  docker imager | grep jar-test #查看镜像 |
| # 设置基础镜像openjdk:latest  FROM openjdk:latest  # 在镜像中创建/mdata 目录  RUN mkdir /mdata  # 将data目录中的文件 拷贝到 镜像中mdata 目录下  ADD ./data /mdata  # 设定工作目录  WORKDIR /mdata  # 暴露镜像端口 3003（不同镜像暴露端口不同，具体端口请根据程序配置确定）  EXPOSE 3003  # 镜像中 应用启动命令  ENTRYPOINT ["java","-jar","sicau-bst-admin-1.08.jar","--spring.profiles.active=prod"] |



## 常见应用Dockerfile

#### 基于java语言开发的jar包

在软件项目部署实施过程中，jar包出现的频率较高，部署实施也相对简单，一般通过命令java -jar \*.jar 的形式即可启动项目。项目实施过程中请格外主要jdk版本是否满足当前程序包的运行要求，可通过java -version 查看jdk版本。



|  |
| --- |
| FROM openjdk:latest  RUN mkdir /mdata  ADD ./data /mdata  WORKDIR /mdata  EXPOSE 3003  ENTRYPOINT ["java","-jar","sicau-bst-admin-1.08.jar","--spring.profiles.active=prod"] |

#### 基于java语言开发的war包

在软件项目部署实施过程中，war包出现的频率同样高，常使用Apache-tomcat-\*运行war包程序；项目镜像制作过程中注意Apache-tomcat基础镜像版本。

|  |
| --- |
| FROM tomcat:latest  RUN rm -rf /usr/local/tomcat/webapps/\*  COPY ./sicau-bst.war /usr/local/tomcat/webapps |

#### 基于PHP语言的项目包

本项目基于php5进行开发，对应版本选择php:5.6.40，打包时需安装GD扩展和mysql扩展，若执行apt-get update时出现NO\_PUBKEY错误，请根据错误提示修改Dockerfile中响应的keys值。apt-get源可根据现场网络情况修改。

实际生产环境中，请根据开发版本选择对应版本的php基础镜像。

|  |
| --- |
| FROM php:5.6.40-apache  #配置国内apt-get源  COPY ./sources.list /etc/apt/sources.list  #安装依赖  RUN apt-get install -y gnupg  #若更新软件包时，出现 NO\_PUBKEY 871920D1991BC93C错误，请根据错误提示修改key值  RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 3B4FE6ACC0B21F32  RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 871920D1991BC93C  #更新已安装的软件包  RUN apt-get update  RUN apt-get install -y libfreetype6-dev libmcrypt-dev libpng-dev  #安装并开启GD扩展  RUN docker-php-ext-install gd  RUN docker-php-ext-enable gd  #安装并开启mysql扩展  RUN docker-php-ext-install mysql  RUN docker-php-ext-enable mysql  #复制项目文件  COPY ./yljn.sicau.edu.cn /var/www |
| #sources.list 文件内容  deb http://archive.ubuntu.com/ubuntu/ focal main restricted  deb http://archive.ubuntu.com/ubuntu/ focal-updates main restricted  deb http://archive.ubuntu.com/ubuntu/ focal universe  deb http://archive.ubuntu.com/ubuntu/ focal-updates universe  deb http://archive.ubuntu.com/ubuntu/ focal multiverse  deb http://archive.ubuntu.com/ubuntu/ focal-updates multiverse  deb http://archive.ubuntu.com/ubuntu/ focal-backports main restricted universe multiverse  deb http://security.ubuntu.com/ubuntu/ focal-security main restricted  deb http://security.ubuntu.com/ubuntu/ focal-security universe  deb http://security.ubuntu.com/ubuntu/ focal-security multiverse |

#### 基于node.js框架开发的项目包

|  |
| --- |
| FROM node:latest  COPY ./ietc /usr/local/ietc  WORKDIR /usr/local/ietc  EXPOSE 8001  CMD ["npm","run","start\_single"] |

#### 基于.net框架开发的项目包

基于dotnet构建的docker镜像差异较大，请根据实际版本选择对应的基础镜像，此处Dockerfile文件仅供参考；

|  |
| --- |
| FROM microsoft/dotnet:2.1-aspnetcore-runtime  RUN apt-get update  RUN apt-get install -y --no-install-recommends libgdiplus libc6-dev  #容器中系统的工作空间  WORKDIR /app  #拷贝当前文件夹下的文件到容器中系统的工作空间  COPY ./app /app  #设置Docker容器对外暴露的端口  EXPOSE 5000  #容器中使用 ["dotnet","系统启动的dll"] 来运行应用程序  ENTRYPOINT ["dotnet", "NBI.DashBoardService.Core.dll"] |

#### 基于nginx代理部署的前端包

本项目为vue3开发的前端项目，使用nginx代理静态文件，打包构建前端镜像；

|  |
| --- |
| FROM nginx:latest  COPY ./dist /usr/share/nginx/html  COPY ./ssl /etc/nginx/ssl  COPY ./nginx.conf /etc/nginx  EXPOSE 80,443 |

## 常见应用yaml文件

### **Mysql**

本yaml为部署单机版mysql数据库，仅对数据目前进行持久化存储。不同版本数据库配置文件my.cnf有所差异，请根据具体版本编写my.cnf文件。

|  |
| --- |
| apiVersion: apps/v1  kind: StatefulSet  metadata:  name: ietc-mysql  namespace: ietc-manager  labels:  app: ietc-mysql  annotations:  kubesphere.io/creator: '75093'  kubesphere.io/description: ietc-mysql数据库  spec:  replicas: 1  selector:  matchLabels:  app: ietc-mysql  template:  metadata:  creationTimestamp: null  labels:  app: ietc-mysql  annotations:  kubesphere.io/restartedAt: '2022-07-07T10:39:57.928Z'  logging.kubesphere.io/logsidecar-config: '{}'  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  - name: volume-r10pqv  persistentVolumeClaim:  claimName: ietc-mysql-pvc  - name: volume-j7513f  ##### mysql配置文件my.conf；可根据实际情况自行确定是否需要单独配置mysql配置；  configMap:  name: ietc-mysql-conf  defaultMode: 420  containers:  - name: container-snfh89  image: 'mysql:8.0.29'  ports:  - name: tcp-3306  containerPort: 3306  protocol: TCP  - name: tcp-33060  containerPort: 33060  protocol: TCP  env:  - name: MYSQL\_ROOT\_PASSWORD  value: Password@2021  - name: TZ  value: Asia/Shanghai  resources:  limits:  cpu: '1'  memory: 2000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  - name: volume-r10pqv  mountPath: /var/lib/mysql  ####### mysql 配置文件，my.conf ######  - name: volume-j7513f  readOnly: true  mountPath: /etc/my.cnf  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  schedulerName: default-scheduler  serviceName: ietc-mysql-aj1v  podManagementPolicy: OrderedReady  updateStrategy:  type: RollingUpdate  rollingUpdate:  partition: 0  revisionHistoryLimit: 10 |
| ### mysql配置文件my.cnf 仅对当前版本（mysql:8.0.29）确保正确性，其它版本配置文件请自行查阅文档；  [mysqld]  lower\_case\_table\_names = 1 #0：区分大小写，1：不区分大小写  max\_connections = 2000  max\_connect\_errors = 1000 #jiechu: flush host  sql\_mode=STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION  event\_scheduler = 1  innodb\_buffer\_pool\_size = 4G # (adjust value here, 50%-70% of total RAM)  innodb\_log\_file\_size = 256M  innodb\_flush\_log\_at\_trx\_commit = 1 # may change to 2 or 0  innodb\_flush\_method = O\_DIRECT  innodb\_read\_io\_threads = 4  innodb\_write\_io\_threads = 4  innodb\_change\_buffer\_max\_size = 40  myisam\_sort\_buffer\_size = 128M  myisam\_max\_sort\_file\_size = 10G  table\_definition\_cache = 2048  table\_open\_cache = 2048  explicit\_defaults\_for\_timestamp = false  tmp\_table\_size = 256M 　　 #64-256最佳，默认128M  max\_heap\_table\_size = 2G  server-id = 1 # 确保在整个Mysql集群中唯一  log-bin=mysql-bin # 开启二进制日志  expire\_logs\_days=7 #日志保留7天  binlog-do-db=ietc\_qcloud ##记录库日志  character-set-server = utf8mb4 #服务器字符集  collation-server = utf8mb4\_general\_ci #服务器默认的字符集  binlog-ignore-db=information\_schema ##不记录日志的库  binlog-ignore-db=mysql  binlog-ignore-db=performance\_schema  binlog-ignore-db=sys |

### **Redis**

本yaml文件部署为单机redis，数据未做持久化存储，数据仅存储在内存中，容器重启后数据将丢失。若需对数据进行持久化存储，请自行查阅相关资料配置redis数据持久化。

|  |
| --- |
| kind: StatefulSet  apiVersion: apps/v1  metadata:  name: redis  namespace: ietc-test  labels:  app: redis  annotations:  kubesphere.io/creator: '75093'  spec:  replicas: 1  selector:  matchLabels:  app: redis  template:  metadata:  creationTimestamp: null  labels:  app: redis  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  containers:  - name: container-hj9vss  image: redis  ports:  - name: tcp-6379  containerPort: 6379  protocol: TCP  resources:  limits:  cpu: '1'  memory: 2000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  schedulerName: default-scheduler  serviceName: redis-gpfu  podManagementPolicy: OrderedReady  updateStrategy:  type: RollingUpdate  rollingUpdate:  partition: 0  revisionHistoryLimit: 10 |

### **Mongodb**

本yaml为部署单机版mongodb，使用默认配置文件和端口，仅对数据存储目录/data/db 进行持久化存储。若需集群部署，请自行查阅相关资料。

|  |
| --- |
| kind: StatefulSet  apiVersion: apps/v1  metadata:  name: mongodb  namespace: ietc-test  labels:  app: mongodb  annotations:  kubesphere.io/creator: '75093'  spec:  replicas: 1  selector:  matchLabels:  app: mongodb  template:  metadata:  creationTimestamp: null  labels:  app: mongodb  annotations:  logging.kubesphere.io/logsidecar-config: '{}'  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  - name: volume-s58o7g  persistentVolumeClaim:  claimName: mongodb  containers:  - name: container-mayk2g  image: mongo  ports:  - name: tcp-27017  containerPort: 27017  protocol: TCP  resources:  limits:  cpu: '1'  memory: 2000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  - name: volume-s58o7g  mountPath: /data/db  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  schedulerName: default-scheduler  serviceName: mongodb-aath  podManagementPolicy: OrderedReady  updateStrategy:  type: RollingUpdate  rollingUpdate:  partition: 0  revisionHistoryLimit: 10 |

### **Nginx**

本yaml文件为nginx代理静态文件实现前端应用部署，nginx应用部署相同，配置文件参照如下 nginx.conf。

|  |
| --- |
| kind: Deployment  apiVersion: apps/v1  metadata:  name: hyb-web  namespace: huiyinbi  labels:  app: hyb-web  annotations:  deployment.kubernetes.io/revision: '15'  kubesphere.io/creator: '75093'  spec:  replicas: 0  selector:  matchLabels:  app: hyb-web  template:  metadata:  creationTimestamp: null  labels:  app: hyb-web  annotations:  kubesphere.io/restartedAt: '2023-04-06T07:23:32.947Z'  logging.kubesphere.io/logsidecar-config: '{}'  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  - name: volume-a24yiu  configMap:  name: nginx-conf  defaultMode: 420  containers:  - name: container-woqeg1  image: '192.168.1.1:5000/huiyinbi/hyb-web-k8s:v20230428'  ports:  - name: tcp-80  containerPort: 80  protocol: TCP  env:  - name: TZ  value: Asia/Shanghai  resources:  limits:  cpu: '4'  memory: 8000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  - name: volume-a24yiu  readOnly: true  mountPath: /etc/nginx/nginx.conf  subPath: nginx.conf  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  imagePullSecrets:  - name: harbor  schedulerName: default-scheduler  strategy:  type: RollingUpdate  rollingUpdate:  maxUnavailable: 25%  maxSurge: 25%  revisionHistoryLimit: 10  progressDeadlineSeconds: 600 |
| ### nginx.conf 配置文件  user nginx;  worker\_processes auto;  error\_log /var/log/nginx/error.log warn;  pid /var/run/nginx.pid;  events {  worker\_connections 1024;  }  http {  include /etc/nginx/mime.types;  default\_type application/octet-stream;  log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  '$status $body\_bytes\_sent "$http\_referer" '  '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  access\_log /var/log/nginx/access.log main;  sendfile on;  keepalive\_timeout 65;  server {  listen 443 ssl;  server\_name bst.sicau.edu.cn;  ssl\_certificate /etc/nginx/ssl/\_.sicau.edu.cn\_cert\_chain.pem;  ssl\_certificate\_key /etc/nginx/ssl/\_.sicau.edu.cn\_key.key;  ssl\_session\_cache shared:SSL:1m;  ssl\_session\_timeout 5m;  ssl\_protocols TLSv1.2 TLSv1.3;  ssl\_ciphers EECDH+CHACHA20:EECDH+CHACHA20-draft:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3DES:!MD5;  ssl\_prefer\_server\_ciphers on;  location / {  root /usr/share/nginx/html;  index index.html;  try\_files $uri $uri/ /index.html;  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }  location /Api/ {  add\_header Access-Control-Allow-Origin \*;  add\_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';  add\_header Access-Control-Allow-Headers 'DNT,X-Mx-ReqToken,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Authorization';  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_pass http://10.104.85.172:3003/;  }  }  } |

### **基于jar包的java项目**

|  |
| --- |
| kind: Deployment  apiVersion: apps/v1  metadata:  name: hyb-server  namespace: huiyinbi  labels:  app: hyb-server  annotations:  deployment.kubernetes.io/revision: '8'  kubesphere.io/creator: '75093'  spec:  replicas: 0  selector:  matchLabels:  app: hyb-server  template:  metadata:  creationTimestamp: null  labels:  app: hyb-server  annotations:  logging.kubesphere.io/logsidecar-config: '{}'  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  - name: volume-asewcl  persistentVolumeClaim:  claimName: huiyinbi-server  containers:  - name: container-5415bq  image: '10.255.229.184:5000/huiyinbi/hyb-server-k8s:v20230504'  ports:  - name: tcp-3003  containerPort: 3003  protocol: TCP  env:  - name: TZ  value: Asia/Shanghai  - name: BSTBASEURL  value: 'https://10.255.252.184:30032/Api/'  - name: uploadFile  value: /uploadFile/  - name: MYSQLHOST  value: 10.111.155.180  - name: MYSQLPORT  value: '3306'  - name: MYSQLDBNAME  value: huiyinbi-qcloud  - name: MYSQLUSER  value: root  - name: MYSQLPW  value: Sicau@xjzx2023  resources:  limits:  cpu: '4'  memory: 8000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  - name: volume-asewcl  mountPath: /uploadFile  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  imagePullSecrets:  - name: harbor  schedulerName: default-scheduler  strategy:  type: RollingUpdate  rollingUpdate:  maxUnavailable: 25%  maxSurge: 25%  revisionHistoryLimit: 10  progressDeadlineSeconds: 600 |

### **基于Apache-tomcat运行的war包**

|  |
| --- |
| kind: Deployment  apiVersion: apps/v1  metadata:  name: apache-tomcat  namespace: ietc-test  labels:  app: apache-tomcat  annotations:  deployment.kubernetes.io/revision: '1'  kubesphere.io/creator: '75093'  spec:  replicas: 0  selector:  matchLabels:  app: apache-tomcat  template:  metadata:  creationTimestamp: null  labels:  app: apache-tomcat  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  containers:  - name: container-guvslg  image: tomcat  ports:  - name: tcp-8080  containerPort: 8080  protocol: TCP  resources:  limits:  cpu: '2'  memory: 4000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  schedulerName: default-scheduler  strategy:  type: RollingUpdate  rollingUpdate:  maxUnavailable: 25%  maxSurge: 25%  revisionHistoryLimit: 10  progressDeadlineSeconds: 600 |

### **PHP项目包**

|  |
| --- |
| kind: Deployment  apiVersion: apps/v1  metadata:  name: yljn-v1  namespace: dag-yljn  labels:  app: yljn  version: v1  annotations:  deployment.kubernetes.io/revision: '13'  kubesphere.io/creator: '75093'  spec:  replicas: 1  selector:  matchLabels:  app: yljn  version: v1  template:  metadata:  creationTimestamp: null  labels:  app: yljn  version: v1  annotations:  kubesphere.io/restartedAt: '2023-05-23T05:04:30.535Z'  logging.kubesphere.io/logsidecar-config: '{}'  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  - name: volume-53yrxr  persistentVolumeClaim:  claimName: yljn-data  containers:  - name: container-capsfj  image: '192.168.1.1:5000/dag\_yljn/dag-yljn:5.6.34-k8s'  ports:  - name: tcp-80  containerPort: 80  protocol: TCP  resources:  limits:  cpu: '4'  memory: 8000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  - name: volume-53yrxr  mountPath: /var/www/html  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  nodeSelector:  kubernetes.io/hostname: k8s-node01  serviceAccountName: default  serviceAccount: default  securityContext: {}  imagePullSecrets:  - name: dockerhub  schedulerName: default-scheduler  strategy:  type: RollingUpdate  rollingUpdate:  maxUnavailable: 25%  maxSurge: 25%  revisionHistoryLimit: 10  progressDeadlineSeconds: 600 |

### **Node.js 项目包**

|  |
| --- |
| kind: Deployment  apiVersion: apps/v1  metadata:  name: ietc-server  namespace: ietc-manager  labels:  app: ietc-server  annotations:  deployment.kubernetes.io/revision: '37'  kubesphere.io/creator: '75093'  kubesphere.io/description: ietc综合管理系统—后端  spec:  replicas: 1  selector:  matchLabels:  app: ietc-server  template:  metadata:  creationTimestamp: null  labels:  app: ietc-server  annotations:  kubesphere.io/restartedAt: '2023-05-25T08:10:05.100Z'  logging.kubesphere.io/logsidecar-config: '{}'  spec:  volumes:  - name: host-time  hostPath:  path: /etc/localtime  type: ''  - name: volume-9h2iqs  persistentVolumeClaim:  claimName: ietc-server-pvc  - name: volume-9lytkx  configMap:  name: ietc-server-def  defaultMode: 420  - name: volume-ilf22j  configMap:  name: ietc-server-por  defaultMode: 420  containers:  - name: ietc-server  image: '192.168.1.1:5000/ietc/ietc-server-k8s:v20221008'  ports:  - name: tcp-8001  containerPort: 8001  protocol: TCP  env:  - name: TZ  value: Asia/Shanghai  resources:  limits:  cpu: '1'  memory: 16000Mi  volumeMounts:  - name: host-time  readOnly: true  mountPath: /etc/localtime  - name: volume-9h2iqs  mountPath: /usr/local/ietc/public/uploads  - name: volume-9lytkx  readOnly: true  mountPath: /usr/local/ietc/dist/config/config.default.js  subPath: config.default.js  - name: volume-ilf22j  readOnly: true  mountPath: /usr/local/ietc/dist/config/config.prod.js  subPath: config.prod.js  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  imagePullPolicy: IfNotPresent  restartPolicy: Always  terminationGracePeriodSeconds: 30  dnsPolicy: ClusterFirst  serviceAccountName: default  serviceAccount: default  securityContext: {}  imagePullSecrets:  - name: ietc-registry-key  schedulerName: default-scheduler  strategy:  type: RollingUpdate  rollingUpdate:  maxUnavailable: 25%  maxSurge: 25%  revisionHistoryLimit: 10  progressDeadlineSeconds: 600 |

# **架构规划**

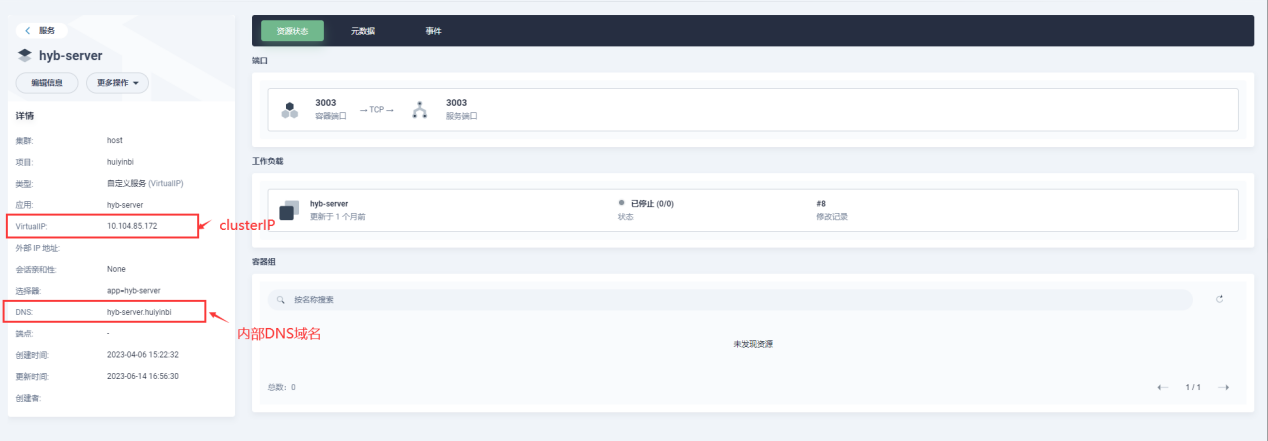
对于刚接触docker和k8s不久的用户，常常充满疑惑，不清楚docker或者k8s环境下各应用间是如何通信，如何交换数据的，各应用组件之间是如何连通的，用户又是以何种方式去访问应用服务。带着这样的疑问，我们先了解docker和k8s的一些基础网络知识。

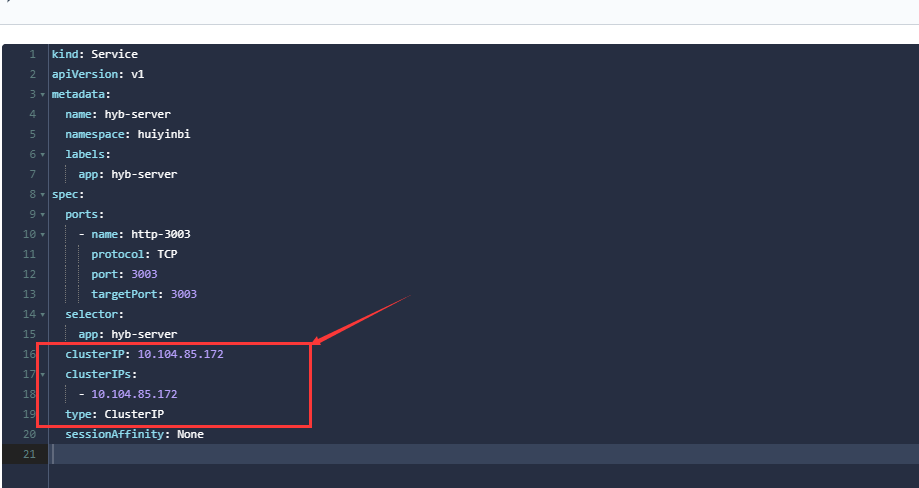
docker容器共有5种网络类型，默认的类型是桥接模式，通过一张名为docker0的网卡连接容器内部网络和服务器网络。docker部署应用时，我们通常称为单机容器模式，在docker run 或docker-compose 中配置端口映射，将容器端口映射到宿主机，通访问宿主机的IP+Port即可访问对应的应用。

k8s中容器网络分为集群内部网络和集群外部网络。集群内部网络供pod、service之间互连互访使用。 Pod配置地址池192.168.0.0/16 可为pod应用分配65536个地址。 service-cluster-ip-range配置地址池10.96.0.0/12 可为service服务分配83万个Cluster IP。更多k8s网络相关知识，请自行查阅资料。

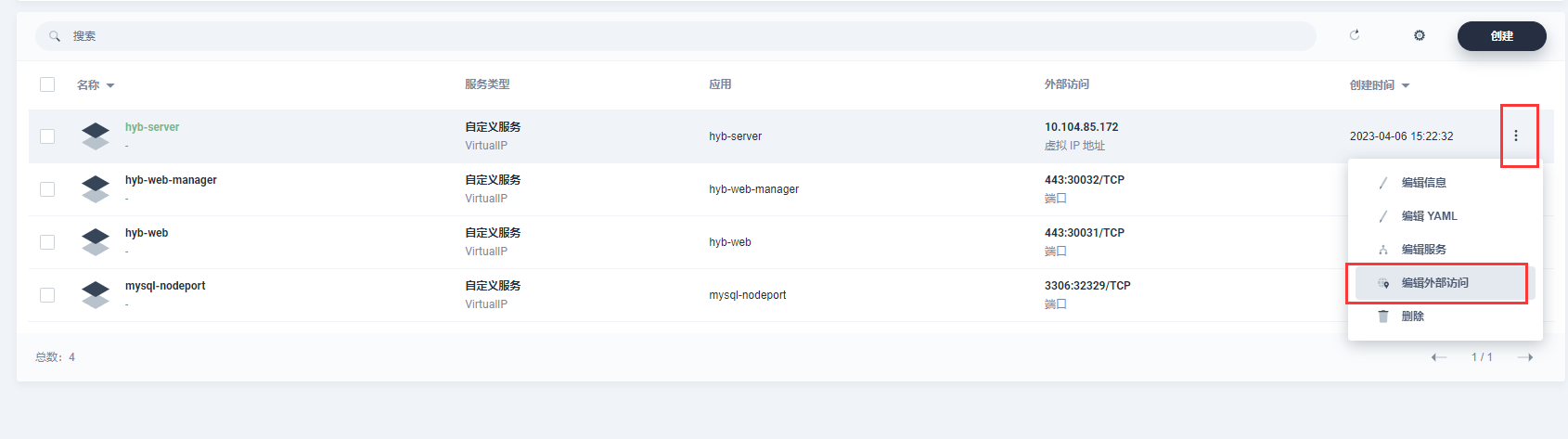
## k8s应用部署架构规划要求

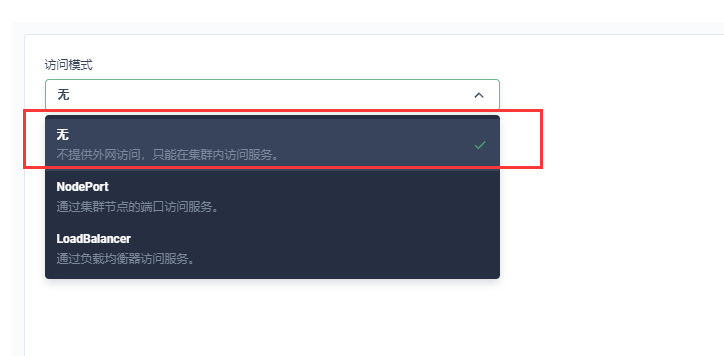
1. 各系统组件内部访问，数据交互统一使用**Cluster IP**或**内部DNS**域名访问；



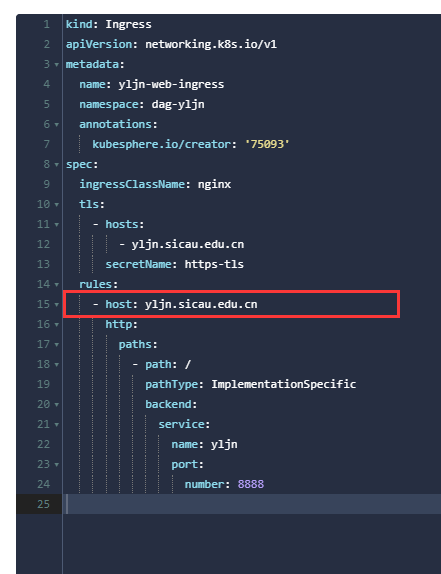


1. 系统部署调试阶段，数据库这类应用可配置NodePort端口，使用集群服务器IP+NodePort端口连接，调试完毕请关闭NodePort端口。**系统各组件之间强烈建议不使用NodePort端口**进行数据交互，NodePort端口默认范围**30000-32767**。

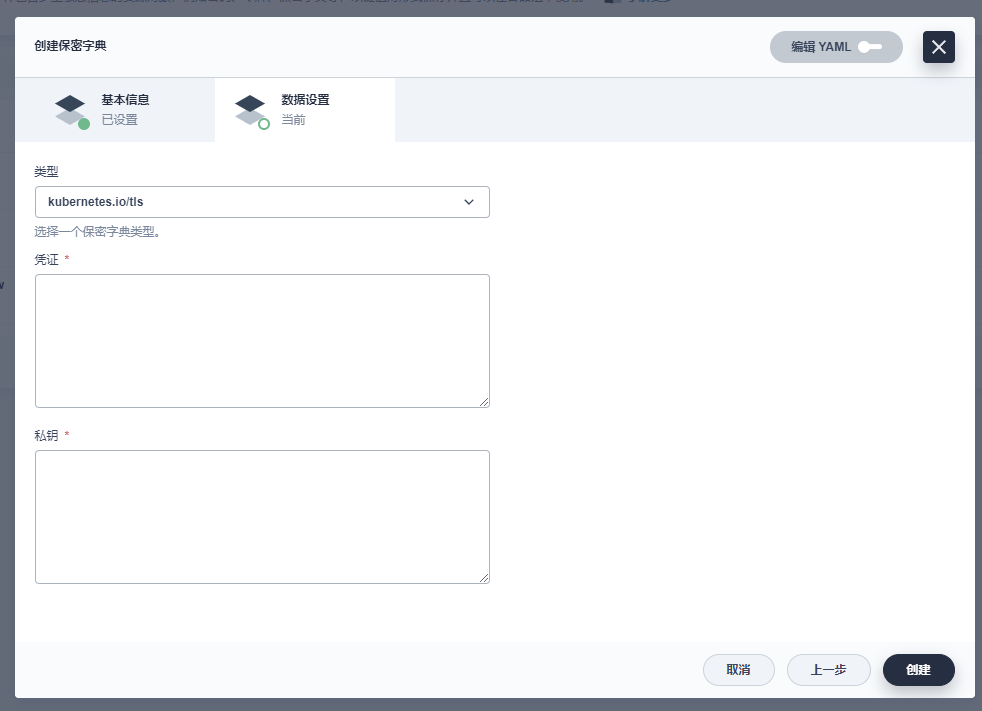




1. B/S架构应用系统，统一使用 ingress 配置域名作为系统统一入口。Ingress 配置请参照2.4.2节 应用部署-[创建路由（ingress）](#_创建路由（ingress）)部署。



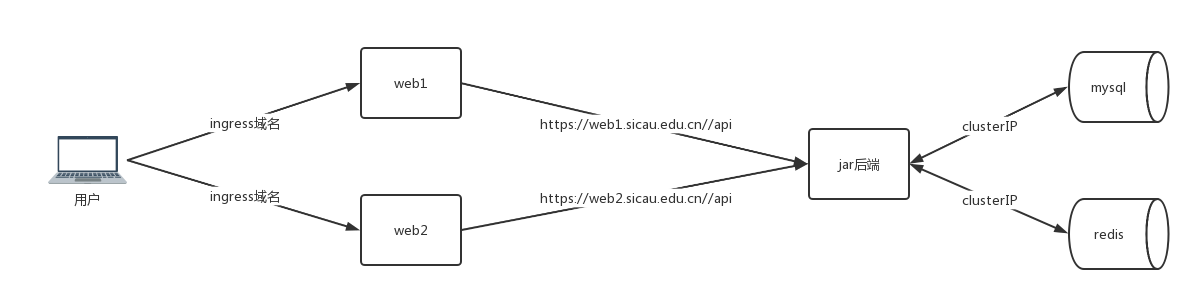
1. 必须实现全站https访问，禁止使用http协议访问。若需https证书，请联系管理老师获取。https证书可在ingress 配置中实现，无需在程序内部单独配置。https证书配置请参照2.4.2节 应用部署-[配置https证书](#_配置https证书)。



## 网络架构示例

本示例系统包含一个基于java开发的jar后端包，两个vue框架开发使用nginx代理的前端包，mysql数据库，redis库。

五个组件之间使用cluster IP互访，通过ingress配置域名发布web端访问页面。Ingress层配置https证书，实现web端加密访问。



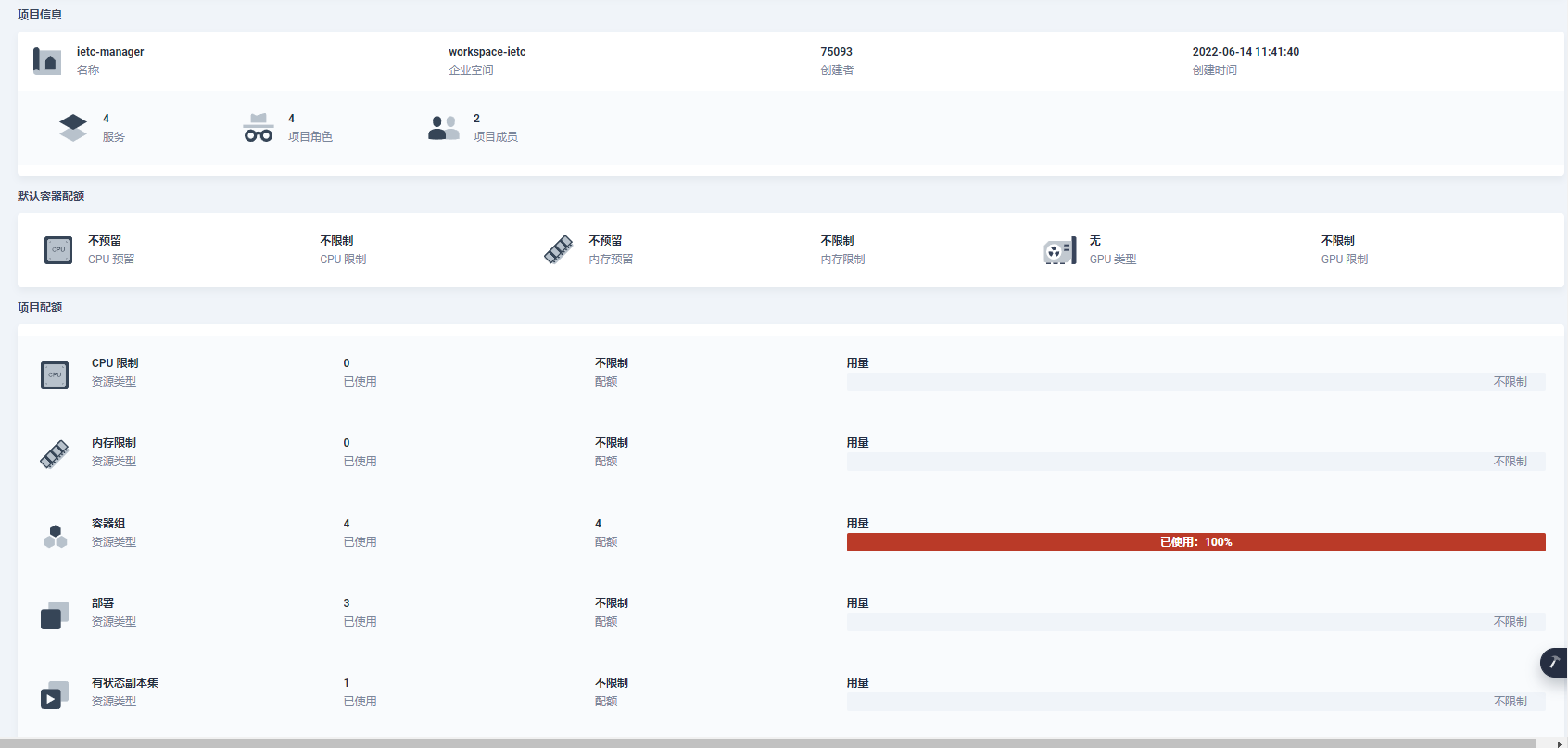
说明：

1. Web1 ingress域名：<https://web1.sicau.edu.cn>
2. Web2 ingress域名：<https://web2.sicau.edu.cn>
3. 由于用户访问前端使用https协议，若前端使用http协议访问后端，这种异构方式浏览器会认为不安全，浏览器会阻止前端向后端发起的访问；当用户使用域名访问前端应用，前端访问后端时使用cluster访问时，浏览器同样会认为不安全并阻止访问。因此两个前端访问后端时，需通过nginx 代理 解决上述问题。
4. 前后端分离项目中，若前后端分开部署，通过单独的nginx代理后端应用，会出现同源跨域问题。要解决同源跨域问题，只能通过前端自身代理后端应用。

# **常见问题**

1. 应用创建失败，提示pod数量已达到设定上限，请联系管理员老师调整pod配额。错误提示信息如下图：





1. 存储卷（pvc）无创建按钮，无法新增分配pvc，请联系管理员老师分配新的pvc。错误提示信息如下图：

